Master Thesis

University of Bremen, Alfred Wegener Insitute

February 2023

Sea Ice Modelling in Python

Jan Philipp Gärtner

jan.gaertner@awi.de

Supervisor:

Dr. Martin Losch

Examiner:

Prof. Thomas Jung

Prof. Markus Jochum

1 Abstract

Sea ice has a significant impact on the exchange of heat and momentum between ocean and atmosphere. For modelling the ocean in higher latitudes it is therefore important that the development of sea ice is included in the model. Climate models are usually written in Fortran due to their legacy and the good computational performance of Fortran. While this has computational benefits, there are also some downsides to using Fortran when compared to using Python code. The first one is its lower popularity among scientists who are not or just starting to work in climate modeling which makes it more difficult and time consuming to start researching with Fortran based models. The second one is it being a low-level language and as such more cumbersome to read, maintain, and develop than the high-level language Python. Häfner et al. (2018) presented the Python based ocean model Veros (https://veros.readthedocs.io/en/latest/) that is a translation of the Fortran backend of the ocean model PyOM2 (https://wiki.cen.uni-hamburg.de/ifm/TO/pyOM2). By using the Python library JAX (Bradbury et al., 2018) the performance of Veros was comparable to the original Fortran model (Häfner et al., 2021). Veros does not yet include a sea ice component and is thus not suited for modeling the ocean in higher latitudes. This work presents a model that is a Fortran to Python translation of the sea ice component of the MITgcm (Marshall et al. (1997), Losch et al. (2010), https://github.com/MITgcm/MITgcm, https://mitgcm.org/documentation/). The goal is to couple it to Veros to extend its range of usability. For the dynamics, only the elastic-viscous-plastic solver is implemented.

Contents

2 Introduction

Climate models are usually written in Fortran. On the one side, this is simply due to their legacy. Fortran has a good backwards compatibility and a lot of old code can be reused by newer models written in newer versions of Fortran. This saves a lot of time compared to rewriting legacy code in a new language. Another very important reason for the use of Fortran is its good computational performance as its main focus is on the speed of numerical calculations. Climate models usually execute a lot of calculations as they are iterated over many time steps and on large grids, therefore performance is especially crucial for this type of modeling. Despite the advantages of Fortran, it has a relatively low popularity in comparison to newer languages like Python among scientists who are not working in climate modeling. This makes it more difficult to start researching in this field as a lot of time needs to be invested into learning a new programming language. The higher popularity of Python also means that there are more libraries that can be imported and used. Fortran is a low-level language and as such not optimized for readability. Using a high-level language such as Python would make it much easier to make structural changes to the code like implementing new parameterizations and to debug the code after doing so.

The higher popularity and better readability of Python alone are not a good enough reason to switch from Fortran to Python though as the performance difference without any additional work or extensions is too large to justify the use of Python. To increase the performance of Python and to close this gap, JAX (Bradbury et al., 2018) is used. JAX is a Python library for array based computing. Its syntax is very similar to that of Numpy. A code written in Numpy therefore does not need to be fundamentally changed in order to work with JAX. JAX includes a set of useful transformations that can be used on functions like vectorization, parallelization, automatic differentiation and just-in-time (jit) compilation. jit comilation is the compilation of code at run time. This drastically reduces the runtime of iterative calculations and is therefore very well suited for climate models as they iterate over many time steps. Another big advantage of JAX besides being able to use these transformations is that it is accelerator agnostic, meaning the same code can be run on CPUs and GPUs. GPUs as compared to CPUs are due to the higher number of cores much more suitable for parallel computing. Using GPUs instead of CPUs can also reduce the power consumption and therefore the carbon footprint of

running climate models.

Häfner et al. (2018) presented a Python based ocean model that is publicly available as the Versatile Ocean Simulator (Veros, https://veros.readthedocs.io/en/latest/). It is a translation of the Fortran backend of the ocean model pyOM2 (v2.1.0, https://wiki.cen.uni-hamburg.de/ifm/TO/pyOM2). By using the jit compiler of JAX, the performance gap to the original Fortran model was closed (Häfner et al., 2021). Veros is fully parallelized and thus well suited for the use of GPUs, allowing it to be run with a much better energy efficiency compared to using a CPU based computing system. It is possible to switch between using Numpy and JAX as a backend when running Veros. This makes the development and implementation of new features easier, as debugging within compiled functions is much more difficult. The initial debugging can be done in Numpy before switching to JAX for performance. So far, Veros does not contain a sea ice model. It is therefore not suitable to be used in high latitudes as the presence of ice and snow on the ocean surface has a significant impact on the local climate (Roots, 1989). Some important aspects are the greatly increased surface albedo of ice or snow compared to open water, and the insulation effect as the heat transfer between ocean and atmosphere is drastically reduced if an ice cover is present. Furthermore, if energy is gained or lost at the surface, it is to be used for melting and freezing instead of a temperature change of the ocean surface. The ice cover also impedes the transport of moisture from the ocean to the atmosphere and has thus an impact on the cloud formation which also impacts the local climate. All of these effects also alter the ocean characteristics. It is thus obvious that an accurate model of the ocean in high latitudes needs to include sea ice.

The development of sea ice can be divided into two parts: the thermodynamics and the dynamics. The first model that contained both of these (the first model whose dynamics included the ice rheology and not just strongly simplified dynamics, respectively) was introduced by Hibler (1979). The model has been developed further, including the implementation of different numerical solvers, e.g. the elastic-viscous-plastic solver (EVP, Hunke and Dukowicz (1997)), the modified EVP solver (mEVP, Bouillon et al. (2013)), the adaptive EVP solver (aEVP, Kimmritz et al. (2016)), and the Jacobian-free Newton-Krylov solver (JNFK, Lemieux et al. (2010)). A comparison of results of different numerical solvers is given by Losch et al. (2010).

This work translates the sea ice component of the Massachusetts Institute of Technol-

ogy general circulation model (MITgcm, Marshall et al. (1997), Losch et al. (2010), https://github.com/MITgcm/MITgcm, https://mitgcm.org/documentation/) that is based on Hibler (1979) into Python. It is coupled to Veros to extend its range of usability.

A big motivation for the development of models in Python is the increased accessibility of the code. With a more structured and readable code, it is easier to implement and test new features, such as new parameterizations for specific processes or new dynamics solvers. After the coupling with Veros, a new parameterization of landfast ice is implemented. Landfast ice is sea ice that stays immobile or nearly immobile for a certain time span. It has multiple important roles. For local communities in the Arctic, it serves as a platform of traveling and hunting (Gearheard et al., 2006). In the Antarctica, it serves as a protection of outer ice shelf margins and reduces ice calving (Massom et al., 2018). It also has an impact on climate models as its development differs from regular sea ice: landfast ice is mostly immobile and thus changes are mostly due to thermodynamics. This will lead to differences in the ocean-atmosphere exchange of heat, momentum, and moisture. It is therefore important that landfast ice is accurately represented in models. There are two possibilities for the formation of landfast ice. The ice cover can be either fixed to the ground by forming down reaching ice keels or laterally to a coast. At the time of the Fortran to Python translation of the sea ice component of the MITgcm, a parameterization of landfast ice due the formation of ice keels was included in the MITgcm. The only form of coastal drag parameterization was the use of a no-slip boundary condition. The tangential velocity at a boundary is zero which can be seen as a very simple form of landfast ice parameterization. This especially underestimates the extent of landfast ice in regions of deeper water where no ice keels reach to the ground. Liu et al. (2022) developed a parameterization of coastal drag to simulate landfast ice to replace the no-slip drag. A new lateral stress term that is dependent on the coastline is added to the momentum equation. In this case, a free-slip boundary condition is used, i.e. there are no gradients of tangential velocity across a boundary. Using a free-slip boundary condition in combination with the coastal drag parameterization gives more control over the lateral drag exerted by the coastline.

In section 3 and 4 the thermodynamics and dynamics of the model are explained. Section 5 describes the technical aspects of the translation from Fortran to Python. The

challenges and results of the coupling with Veros are presented in section 6. After the coupling, the coastal drag parameterization to simulate landfast ice is implemented. The results are presented in section 7. A summary and discussion of this work are given in section 8 and 9. Section 10 contains an outlook to future work with the sea ice model.

3 Thermodynamics

The thermodynamics are implemented as in the subroutine seaice_growth_adx.F of the sea ice package of the MITgcm (https://github.com/MITgcm/MITgcm/blob/master/pkg/seaice/seaice_growth_adx.F).

Typically, a grid cell will not be fully covered with ice and will furthermore contain ice of different thicknesses. To account for this, the variables mean ice thickness h and sea ice concentration A are used. The mean ice thickness is the equivalent ice thickness if all of the ice in the cell were distributed equally over the whole cell area. The sea ice concentration is the fraction of the cell that is covered by ice. For the calculation of the growth rate of the mean ice thickness, a two level thickness configuration is assumed. If the mean ice thickness in a cell is h, then the mean thickness of the actual ice cover, i.e. the equivalent ice thickness if all of the ice in the cell were distributed equally over the fraction of the cell that is covered by ice, is $h_{\text{actual}} = \frac{h}{A}$. A grid cell is assumed to be covered with ice of thickness h_{actual} and open water, weighted with A and $1 - A$, respectively. The growth rate of the mean ice thickness is then calculated from the growth rate of the part of the cell that is covered with ice and the growth rate of the part that is open water (Hibler, 1979):

$$\frac{\Delta h}{\Delta t} = f(h_{\text{actual}})A + f(0)(1 - A) + f_{ml} \tag{1}$$

where $f(h)$ is the growth rate of ice of thickness h, and f_{ml} is an additional implementation of the growth rate due to the mixed layer temperature. For $A \to 0$ the actual ice thickness tends to infinity and is therefore calculated by using a regularized ice concentration that has a certain minimum value. The total change in the sea ice concentration is

$$\frac{\Delta A}{\Delta t} = \begin{cases} \frac{f(0)}{h_0}(1 - A) & \text{if } f(0) > 0 \\ 0 & \text{if } f(0) < 0 \end{cases} + \begin{cases} 0 & \text{if } \frac{\Delta h}{\Delta t} > 0 \\ \frac{A}{2h}\frac{\Delta h}{\Delta t} & \text{if } \frac{\Delta h}{\Delta t} < 0 \end{cases} \tag{2}$$

where h_0 is the lead closing parameter that describes the lateral growth. As A cannot exceed 100%, a cutoff function is used to ensure a maximum value of 1.

3.1 Ice Growth in Open Water

The ice growth in open water is given by the net heat flux out of the ocean. If heat is leaving the ocean, this is balanced by the release of latent heat, i.e. freezing. This only

occurs if the sea ice concentration of a grid cell is less than one (see equation (1)). The change in ice thickness due to open water growth is

$$f(0) = \frac{Q_{\text{net}}}{\rho_i L_f} \tag{3}$$

where Q_{net} is the net open water heat flux out of the ocean, ρ_i the density of ice, and L_f the latent heat of fusion.

3.2 Ice Growth of an Existing Ice Cover

For calculating the ice growth of an existing ice cover, first the heat fluxes through the ice (the ice-snow system to be more precise) need to be determined. For this, the heat flux divergence at the ice/snow-atmosphere interface and the corresponding temperature change is calculated. With this new surface temperature, the heat fluxes are recalculated (Parkinson and Washington, 1979). This process is iterated to get the correct surface temperature and heat fluxes. The steady state boundary condition at the ice/snow surface is

$$F_c = F_{ia} \tag{4}$$

where F_c is the conductive heat flux through the ice-snow system and F_{ia} the heat flux from the ice/snow surface to the atmosphere. The conductive heat flux is given by

$$F_c = k_{\text{eff}}(T_b - T_s) \tag{5}$$

where k_{eff} is the effective conductivity of the ice-snow system, T_b the temperature at the ice bottom which is set to the freezing temperature, and T_s the ice/snow surface temperature. The heat flux to the atmosphere is given by the outgoing fluxes of sensible heat F_{sens}, latent heat F_{lat}, and longwave radiation F_{lw}, and the absorbed heat fluxes of the incoming longwave and shortwave radiation $F_{\text{absorbedlw}}$ and $F_{\text{absorbedsw}}$, respectively.

$$F_{ia} = F_{\text{sens}} + F_{\text{lat}} + F_{\text{lw}} - F_{\text{absorbedlw}} - F_{\text{absorbedsw}} \tag{6}$$

$$F_{\text{sens}} = C_d c_p \rho_a v(T_s - T_a) \tag{7}$$

$$F_{\text{lat}} = C_d L_s \rho_a v(q_s - q_a) \tag{8}$$

$$F_{\text{lw}} = \sigma \epsilon_s T_s^4 \tag{9}$$

$$F_{\text{absorbedlw}} = \epsilon_s Q_{lw} \tag{10}$$

$$F_{\text{absorbedsw}} = (1 - a_s)(1 - p_{sw})Q_{sw} \tag{11}$$

C_d is the Dalton constant/ sensible and latent heat transfer coefficient, c_p the specific heat of air, ρ_a the density of air, v the surface wind speed, T_a the atmospheric temperature, L_s the latent heat of sublimation, q_s the specific humidity directly at the surface, q_a the specific humidity at 10m height, σ the Stefan-Boltzmann constant, ϵ_s the surface emissivity, a_s the surface albedo, p_{sw} the fraction of the shortwave radiation that passes the ice-snow system (and is not absorbed), and Q_{sw} and Q_{lw} the downward shortwave and longwave radiation, respectively.

If snow is present, no shortwave radiation can pass because of the high extinction coefficient of snow. The penetrating shortwave fraction p_{sw} is then set to 0. For ice with no snow cover Parkinson and Washington (1979) use a constant value for p_{sw}. In the MITgcm, it is calculated from the ice thickness according to:

$$p_{sw} = p_{sw,0}\, e^{-1.5 h_{\text{actual}}} \tag{12}$$

where $p_{sw,0}$ is a constant parameter.

Equation (4) is solved by using the Newton-Raphson method for determining the root of $F_c(T_s) - F_{ia}(T_s)$. If T_0 is a first guess of the surface temperature, then the first approximation to the real value is

$$T_1 = T_0 - (F_c(T_0) - F_{ia}(T_0))/\left(\frac{dF_c}{dT}(T_0) - \frac{dF_{ia}}{dT}(T_0)\right) \tag{13}$$

This surface temperature is then used to recalculate the fluxes. Using this method, only a few iterations are needed for the temperature and thus the fluxes to converge.

If the ocean looses heat to the atmosphere, i.e. the conductive heat flux through the ice-snow system is upward ($F_c > 0$), freezing at the ice bottom will occur to balance the heat loss. The change in ice thickness is then

$$f(h_{\text{actual}}) = \frac{F_c}{\rho_i L_f} \tag{14}$$

If the heat flux is from the atmosphere to the ocean, i.e. the conductive heat flux through the ice-snow system is downward ($F_c < 0$), melting at the surface will occur. In order for the ice to be melted at the surface, the snow cover has to be melted completely. The change in snow thickness is

$$f(h_{\text{s,actual}}) = \frac{F_{ia}}{\rho_s L_f} \tag{15}$$

where ρ_s is the density of snow and $h_{\text{s,actual}} = \frac{h_s}{A}$ the actual snow thickness. The change in ice thickness is then

$$f(h_{\text{actual}}) = \frac{F'_{ia}}{\rho_i L_f} \tag{16}$$

where F'_{ia} is the heat flux available for melting ice that is left after F_{ia} is reduced by the fraction that is used for melting snow.

The snow thickness can also be changed directly by snowfall or by precipitation if the surface temperature is below 0°C.

3.3 Ice Melting due to the Mixed Layer Temperature

The ice thickness also changes due to melting at the bottom if the temperature of the ocean surface is higher than the freezing temperature. Following McPhee (1992), the ocean-ice heat flux is

$$F_{oi} = -St\, u_b^* \rho_w c_{p,w} (T_o - T_b) \tau_{ml} \tag{17}$$

where St is the Stanton number, u_b^* the typical friction velocity at the ice bottom, ρ_w the density of water, $c_{p,w}$ the heat capacity of water, T_o the ocean surface temperature, and τ_{ml} (additionally implemented in the MITgcm) the mixed layer turbulence factor that determines how strongly the heat flux depends on the temperature difference between ocean and ice:

$$\tau_{ml} = \frac{1 + (C_{tf} - 1)}{1 + exp(\frac{A - 0.4}{17.5})} \tag{18}$$

with the tapering factor C_{tf}. The change in ice thickness due to the temperature difference between ocean and ice is then

$$f_{ml} = \frac{F_{oi}}{\rho_i L_f} \tag{19}$$

3.4 Ice Growth due to the Formation of Snow-Ice

Following Archimedes' principle, the ice-snow interface is at the water surface if

$$\rho_s h_s + \rho_i h = \rho_w h \tag{20}$$

Snow is submerged in water if

$$h_{sub} = \frac{\rho_s h_s + \rho_i h}{\rho_w} > h \tag{21}$$

where h_{sub} is the height of the ice-snow system that is submerged. The submerged snow will then be transformed to snow-ice. For this process, three assumptions have been made. Firstly, all of the submerged snow is transformed to ice with the new ice thickness being the submerged height. Secondly, this transformation occurs without the release of latent heat. Thirdly, the mass of the ice-snow system is conserved, i.e. the snow height is not reduced by the height of snow that is submerged but by the height that is required in order for the submerged height to stay the same. Using these assumptions, the change in ice and snow thicknesses are calculated according to

$$\Delta h = h_{sub} - h \tag{22}$$

$$\Delta h_s = -(h_{sub} - h)\frac{\rho_i}{\rho_s} \tag{23}$$

4 Dynamics

4.1 Spatial Discretization

The variables are given on an Arakawa C-grid (Arakawa and Lamb, 1977). The c-point refers to the cell center, the u-point is at the middle of the western cell wall, the v-point at the middle of the southern cell wall, and the z-point at the south-west corner of the cell. The scalar values like h, A, and the heat fluxes defined in section 3 are stored on the c-points. In general, if not mentioned otherwise, all variables are stored at the c-points. The components of the velocity vector are stored on the respective velocity points. When solving the momentum equation, the sea ice mass is averaged to the velocity points, indicated with the notation $m^{u/v}$. When calculating the u-component of the velocity vector m^u is used, while for calculating the v-component m^v is used. They are calculated according to

$$m^u_{i,j} = \frac{1}{2}(m_{i,j} + m_{i-1,j}) \tag{24}$$

$$m^v_{i,j} = \frac{1}{2}(m_{i,j} + m_{i,j-1}) \tag{25}$$

4.2 The Viscous-Plastic Rheology

The horizontal momentum equation is

$$\rho h \frac{d\boldsymbol{u}}{dt} = \rho h f(v\boldsymbol{i} - u\boldsymbol{j}) + A^{u/v}(\boldsymbol{\tau_a} + \boldsymbol{\tau_w}) + \nabla\sigma - \rho h g \nabla H \tag{26}$$

where $\rho h = m$ is the mass of the ice-snow system, $\boldsymbol{u} = u\boldsymbol{i} + v\boldsymbol{j}$ the horizontal velocity vector, $\boldsymbol{i}$ and $\boldsymbol{j}$ the unit vectors corresponding to the x- and y-axis of the Cartesian coordinate system, respectively, f the Coriolis parameter, $\boldsymbol{\tau_a}$ the wind stress, $\boldsymbol{\tau_w}$ the water stress, σ the internal ice stress tensor, g the gravity, and H the sea surface elevation. The ice concentration centered around the velocity points $A^{u/v}$ is calculated analogously to $m^{u/v}$.

4.2.1 Stress Terms

The wind stress is calculated from the ice-wind velocity difference using

$$\tau_{a,u} = c_a\Big(\cos(\alpha_a)u_{rel,a} - \frac{f}{|f|}\sin(\alpha_a)v_{rel,a}\Big) \tag{27}$$

$$\tau_{a,v} = c_a\Big(\cos(\alpha_a)v_{rel,a} + \frac{f}{|f|}\sin(\alpha_a)u_{rel,a}\Big) \tag{28}$$

where c_a is the air-ice drag coefficient and $u_{rel,a}$ and $v_{rel,a}$ are the component-wise wind velocities relative to the ice velocities. The air turning angle α_a is the angle between the wind forcing and the resulting stress on the ice surface. It is a parameterization of the Ekman spiral that is not resolved in the model.

The water stress is calculated from the difference in ice and ocean surface velocity using

$$\tau_{w,u} = c_w\Big(\cos(\alpha_w)u_{rel,w} - \frac{f}{|f|}\sin(\alpha_w)v_{rel,w}\Big) \tag{29}$$

$$\tau_{w,v} = c_w\Big(\cos(\alpha_w)v_{rel,w} + \frac{f}{|f|}\sin(\alpha_w)u_{rel,w}\Big) \tag{30}$$

where c_w is the linear ice-water drag coefficient, $u_{rel,w}$ and $v_{rel,w}$ the component-wise ocean surface velocities relative to the ice velocities, and α_w the water turning angle.

The relative velocities $u_{rel,a/w}$ and $v_{rel,a/w}$ and the drag coefficients c_a and c_w are calculated at the c-point. After calculating the wind and water stress components using equation (27) - (30), they are interpolated to the velocity points.

4.2.2 Stress Tensor

The stress tensor $\sigma = \begin{pmatrix} \sigma_{11} & \sigma_{21} \\ \sigma_{12} & \sigma_{22} \end{pmatrix}$ defines the rheology, i.e. the relationship between the applied stresses and the resulting deformations. The eigenvalues of σ are the principal stress components $\sigma_{1,2}$:

$$\sigma_{1,2} = \frac{\sigma_{11} + \sigma_{22}}{2} \pm \sqrt{\left(\frac{\sigma_{11} - \sigma_{22}}{2}\right)^2 + \sigma_{21}\sigma_{12}} \tag{31}$$

The stress invariants $\sigma_{I,II}$ can then be defined as

$$\sigma_I = \frac{1}{2}(\sigma_1 + \sigma_2) = \frac{1}{2}(\sigma_{11} + \sigma_{22}) \tag{32}$$

$$\sigma_{II} = \frac{1}{2}(\sigma_1 - \sigma_2) = \frac{1}{2}\sqrt{(\sigma_{11} - \sigma_{22})^2 + 4\sigma_{21}\sigma_{12}} \tag{33}$$

where σ_I is the normal stress and σ_{II} is the shear stress. The stress state of the ice is then given as (σ_I, σ_{II}). The stress states of plastic deformation are given by the elliptical yield curve (Hibler (1979), Fig. 1) defined by

$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial \boldsymbol{u}_i}{\partial x_j} + \frac{\partial \boldsymbol{u}_j}{\partial x_i}\right) \tag{34}$$

$$\sigma_{ij} = 2\eta\dot{\epsilon_{ij}} + [(\zeta - \eta)\dot{\epsilon_{kk}} - \frac{P}{2}]\delta_{ij} \tag{35}$$

where $\dot{\epsilon_{ij}}$ are the components of the strain rate tensor with $\partial/\partial x_{i,j}$ the derivatives in x- and y-direction, respectively, $\zeta = \frac{P}{2\Delta}$ and $\eta = \frac{\zeta}{e^2}$ are the bulk and shear viscosity, respectively, P the ice strength, Δ a measure of the deformation rate, $\dot{\epsilon}_{kk}$ the trace of the strain rate tensor, δ_{ij} the Kronecker delta, and e the axes ratio of the elliptical yield curve. Δ is calculated from the normal strain rate $\dot{\epsilon}_I$ and the shear strain rate $\dot{\epsilon}_{II}$ according to

$$\Delta = \sqrt{\dot{\epsilon}_I^2 + \frac{1}{e^2}\dot{\epsilon}_{II}^2} \tag{36}$$

$$\dot{\epsilon}_I = (\dot{\epsilon}_{11} + \dot{\epsilon}_{22}) \tag{37}$$

$$\dot{\epsilon}_{II} = \sqrt{(\dot{\epsilon}_{11} - \dot{\epsilon}_{22})^2 + 4\dot{\epsilon}_{12}^2} \tag{38}$$

To avoid singularities of ζ for $\Delta \to 0$, a regularized value $\Delta_{\text{reg}} = \max(\Delta, \Delta_{\min})$ is used. This allows for stress states within the yield curve. In this region, the ice flow is viscous and the internal stress increases linearly with the strain rate. If a certain stress threshold is reached, the stress state lies on the yield curve and will deform in a plastic way so that the internal stress stays constant. The stress threshold for plastic deformation is so small that the ice is in its plastic limit most of the time. The viscous-plastic threshold is defined by $\Delta_{\min}$. If $\Delta \leq \Delta_{\min}$, the flow is viscous; if $\Delta > \Delta_{\min}$, the flow is plastic.

$\dot{\epsilon}_{11}$, $\dot{\epsilon}_{22}$, σ_{11}, and σ_{22} are defined on the c-points while $\dot{\epsilon}_{12}$, σ_{12} are defined on the z-points. The components of the divergence of the stress tensor are calculated on the respective velocity points.

The ice strength P is calculated from the ice thickness and cover fraction following Hibler (1979) using

$$P = P^* h e^{-C^*(1-A)} \tag{39}$$

where P^* and C^* are constant parameters. The ice strength is the maximum compressive stress of the ice.

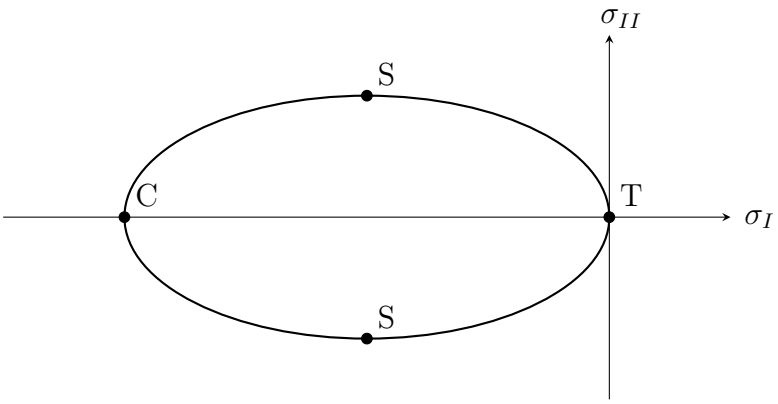

Figure 1: The stress states of the ice for plastic deformation are defined by the elliptical yield curve. They are given in terms of the stress invariants σ_I (normal stress) and σ_{II} (shear stress). The direction of deformation is given by the location on the yield curve. The points T and C are pure tensile and compressive deformation, respectively, and the points S are pure shearing deformation. The stress states on the yield curve that lie inbetween these points will lead to both shearing and tensile/ compressive deformation. For viscous deformation, the stress state lies within the ellipse.

Even without velocities and velocity gradients, i.e. when all the strain rate tensor components are zero, the stress tensor components σ_{11} and σ_{22} are non-zero due to the ice pressure term $p = \frac{P}{2}$ (equation (35)). This will result in an ice flow even though no forcing is present. To avoid this, a replacement pressure can be used (Hibler and Ip, 1995):

$$p_r = \zeta \Delta \tag{40}$$

The use of this, however, yields the problem that when the ice is not moving due to being directly at a boundary, no stress is present to resist further incoming ice. This can lead to an unphysical build up at the boundaries.
The elliptical yield curve is defined in such a way that the ice has resistance to compressive stresses but not to tensile stresses. To include some resistance to tensile stresses, the viscosities and ice pressure can be defined as

$$\zeta_t = s_t \frac{P}{2\Delta} \tag{41}$$

$$\eta_t = \frac{\zeta_t}{e^2} \tag{42}$$

$$p_t = s_t \frac{P}{2} \tag{43}$$

where s_t is the tensile stress factor (König Beatty and Holland, 2010). If both the tensile stress resistance and the replacement pressure are used, the pressure is given by

$$p_{r,t} = \zeta\Delta\frac{1-s_t}{1+s_t} \tag{44}$$

Implemented are the normal pressure term as well as the two modifications with the option to choose between them.

4.3 The Elastic-Viscous-Plastic Method

Integrating the momentum equation explicitly using the viscous-plastic rheology requires very small time steps. Hunke and Dukowicz (1997) showed that for a divergence-free velocity field the time step would need to be of the order of one hundredth of a second for an explicit time stepping scheme to be stable if the grid cell spacing is about 10km. Only being able to use small time steps results in a computationally expensive model. To overcome this, an elastic behaviour is introduced, allowing for larger time steps and thus increasing the numerical efficiency.

The momentum equation is solved using a modified elastic-viscous-plastic (mEVP) solver. To advance from time step n to time step $n+1$, first a number of subcycling iterations indicated with indices p are done (Bouillon et al., 2013):

$$\sigma^{p+1} = \sigma^p + (\sigma(\boldsymbol{u}^p) - \sigma^p)\frac{1}{\alpha} \tag{45}$$

$$\boldsymbol{u}^{p+1} = \boldsymbol{u}^p + \left(\frac{\Delta t}{m^{u/v}}(\nabla\sigma^{p+1} + \boldsymbol{R}^{p+\frac{1}{2}}) + \boldsymbol{u}_n - \boldsymbol{u}^p\right)\frac{1}{\beta} \tag{46}$$

where α and β are numerical relaxation parameters and $\boldsymbol{R}$ contains all the forcing terms from the momentum equation. $\sigma(\boldsymbol{u^p})$ is directly calculated from $\boldsymbol{u^p}$ while σ^p is calculated using equation (45), i.e. from the previous iteration step.

In convergence, when $\sigma^{p+1} \to \sigma^p$ and $\boldsymbol{u}^{p+1} \to \boldsymbol{u}^p$, the equations (45) and (46) become

$$\sigma^p = \sigma(\boldsymbol{u}^p) \tag{47}$$

$$\boldsymbol{u}^p = \frac{\Delta t}{m^{u/v}}(\nabla\sigma^{p+1} + \boldsymbol{R}^{p+\frac{1}{2}}) + \boldsymbol{u}_n \tag{48}$$

$$= \boldsymbol{u}^{n+1} \tag{49}$$

$$\Leftrightarrow \qquad m^{u/v}\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} = \nabla\sigma^{p+1} + \boldsymbol{R}^{p+\frac{1}{2}} \tag{50}$$

and the full viscous-plastic equations are recovered. Convergence is assumed after a fixed number of subcycling iterations. The starting values $\sigma^{p=0}$ and $\boldsymbol{u}^{p=0}$ are 0 and $\boldsymbol{u}^n$,

respectively.

The superscript $^{p+\frac{1}{2}}$ of the forcing term refers to an intermediate calculation after $\boldsymbol{R}^p$ has been calculated. The ice-ocean stress can be treated explicitly or implicitly. In the forcing term $\boldsymbol{R}$, the stress term from equations (29) and (30) is calculated from the ocean velocities and not the ocean velocities relative to the ice with the following differentiation between explicit and implicit treatment. If the ice-ocean stress is treated explicitly, then

$$\boldsymbol{R}^{p+\frac{1}{2}} = \boldsymbol{R}^p - c_d^{u/v}\boldsymbol{u}^p \tag{51}$$

If it is treated implicitly, then

$$\boldsymbol{R}^{p+\frac{1}{2}} = \boldsymbol{R}^p \tag{52}$$

and an additional term is added to the velocity step:

$$\boldsymbol{u}^{p+1} = \boldsymbol{u}^p + (\frac{\Delta t}{m^{u/v}}(\nabla\sigma^{p+1} + \boldsymbol{R}^p) + \boldsymbol{u}_n - \boldsymbol{u}^p)\frac{1}{\beta}\frac{1}{1 + c_d^{u/v}\Delta t\frac{m^{u/v}}{\beta}} \tag{53}$$

c_d^u and c_d^v are the ice-water drag coefficients at the velocity points (explained in more detail in section 4.4).

For the elastic-viscous-plastic method to be stable, the parameters α and β need to be sufficiently large. Kimmritz et al. (2015) performed a 1D stability analysis of the mEVP solver and found the stability criterion to be

$$\alpha\beta >> \gamma \tag{54}$$

where $\gamma = \frac{k^2 P\Delta t}{2m\Delta}$ with $k^2 \leq \frac{\pi^2}{\Delta x^2}$ (note that Δ in the denominator of γ is the one from equation (36) while the other Δ refer to the time step and the grid spacing), and α and β are of similar size. If they are too large, however, the process is slowed down unnecessarily as with larger α and β more subcycling iteration steps are needed. This motivated Kimmritz et al. (2016) to develop an adaptive EVP method (aEVP) with variable values for α and β. As they only need to be large when the grid cell size is small or the viscosities ζ and η are large, they are set to

$$\alpha = \beta = \sqrt{\zeta\frac{c\Delta t}{A_c m}} \tag{55}$$

where A_c is the area of the grid cell and c is an empirical constant. Using these parameters, the method is stable without being too computationally expensive. They are first

calculated at the c-point using equation (55) and then averaged to the u-, v- and z-points. For equation (45) $\alpha^{c/z}$ is used with α^c being used for σ_{11} and σ_{22} and α^z for σ_{12}. The averaging to the z-point is

$$\alpha^z_{i,j} = \frac{1}{4}(\alpha_{i,j} + \alpha_{i,j-1} + \alpha_{i-1,j} + \alpha_{i-1,j-1}) \tag{56}$$

Implemented are both methods with the option to choose between the mEVP solver with constant values of α and β, and the aEVP solver with α and β that are dependent on the ice field and the grid cell size.

4.4 Parameterizing Landfast Ice

The total drag coefficient c_d (used in the momentum equation following equations (51) - (53)) is calculated according to

$$c_d = c_w + c_b \tag{57}$$

where c_w is the linear ice-water drag coefficient (see equation (29) and (30)) and c_b the basal drag coefficient. The basal drag coefficient parameterizes landfast ice. Landfast ice is sea ice that stays immobile or nearly immobile for a certain time span. There are two processes that lead to the formation of landfast ice. Firstly, the ice cover can form ice keels that extend to the ground and build up normal stress which can keep the ice cover at a fixed location or drastically slow down its movement. Secondly, the ice can be fixed to a nearby coast.

The basal drag parameterization covers landfast ice due to the formation of ice keels (Lemieux et al., 2015). The basal drag coefficient is

$$c_b = \begin{cases} \frac{k_2}{\sqrt{u^2+v^2+u^2_{b,0}}}(h - h_c)e^{-C^*_b(1-A)} & \text{if } A > 0.01 \\ 0 & \text{else} \end{cases} \tag{58}$$

where $h_c = \frac{h_w A}{k_1}$ is the critical ice height that allows for the formation of landfast ice, h_w the water depth, k_1, k_2, and C^*_b fixed basal drag parameters, and $u_{b,0}$ a velocity parameter to avoid singularities. In the MITgcm, an implementation of a smooth maximum of $h - h_c$

and 0 has been used to avoid unphysical non-differentiable behaviour:

$$c_b = \begin{cases} \frac{k_2}{\sqrt{u^2+v^2+u_{b,0}^2}} \frac{1}{10} \ln(e^{10(h-h_c)}+1) e^{-C^*(1-A)} & \text{if } A > 0.01 \\ 0 & \text{else} \end{cases} \tag{59}$$

The basal drag coefficient depends strongly on the ice concentration as the mean ice cover of a grid cell can only be regarded as landfast ice if A is close to 1. For small values of A, some fraction of the cell could be landfast ice while other separate areas of sea ice could still be moving.

The formation of landfast ice due the ice being fixed to a boundary can be parameterized by the use of a coastal drag term. At the time of the Fortran to Python translation of the sea ice component of the MITgcm, the only form of coastal drag parameterization was the use of a no-slip boundary condition. In this case, the tangential velocity at a boundary is zero. Liu et al. (2022) developed a parameterization of coastal drag to simulate landfast ice to replace the no-slip drag. By the use of a new drag coefficient that is dependent on the coastline, a new lateral stress term is added to the momentum equation. This lateral drag coefficient is used in combination with a free-slip boundary condition. The total drag coefficient is then

$$c_d = c_w + c_b + c_l \tag{60}$$

where c_l is the lateral drag coefficient. It is calculated according to

$$c_l = \rho_i C_l F \frac{1}{|\boldsymbol{u}| + u_{l,0}} \tag{61}$$

where C_l is a fixed lateral drag coefficient, F the form factor and $u_{l,0}$ a small residual velocity of 0.01 m/s. The form factor is dependent on the coastline.

There are two form factors F_1 and F_2. The first form factor is given by the model grid. Its value in a specific grid cell is given by how many neighbouring grid cells are land cells:

$$F_1^{u/v} = \begin{cases} 0, & \text{no neighbouring land cells in x/y direction} \\ 1, & \text{one neighbouring land cell in x/y direction} \\ 2, & \text{two neighbouring land cells in x/y direction} \end{cases} \tag{62}$$

F^u is to be used for the u-component of the momentum equation, while F^v is used for the v-component.

The second form factor is calculated from a sub-grid scale coastline data set. The data is taken from Natural Earth 10 m Physical Vectors (https://www.naturalearthdata.com/downloads/10m-physical-vectors/). The 10 m coastline vector is projected on the x- and y-axis and normalized by the grid cell size. The normalized integrals of the coastline are

$$F_2^u = \frac{\sum_{n=1}^{N} |l_n \cos\theta_n|}{\Delta x} \tag{63}$$

$$F_2^v = \frac{\sum_{n=1}^{N} |l_n \sin\theta_n|}{\Delta y} \tag{64}$$

where N is the number of coastline points within one grid cell, l_n the length of the nth 10 m coastline vector, θ_n the angle between the nth 10 m coastline vector and the x-axis of the model grid, and Δx and Δy the grid cells lengths in x- and y-direction, respectively. F_2^u and F_2^v are calculated on the z-points according to equation (63) and (64) and then averaged to the u- and v-points to be used in the momentum equation. They are normalized to a maximum value of 3 so they can be used in equation (61) with the same lateral drag coefficient as F_1.

Implemented are both ways of calculating the total drag coefficient: without coastal drag (equation (57)) in combination with a no-slip boundary condition and with coastal drag (equation (60)) in combination with a free-slip boundary condition. When using the coastal drag parameterization, it is possible to switch between the two form factors.

4.5 Advection Scheme

After the ice velocity has been calculated, the advection is done using a second order flux limiter scheme. The advective flux F is formulated as a combination of a first order upwind flux F_1 and a second order Lax-Wendroff flux F_{LW}. The following calculations are for the advection in u-direction. The advection in v-direction is done analogously.

$$F = (1 - \psi(r))F_1 + \psi(r)F_{LW} \tag{65}$$

$$F_1 = U\overline{\theta} - \frac{1}{2}|u|\delta\theta \tag{66}$$

$$F_{LW} = U\overline{\theta} - \frac{1}{2}c|u|\delta\theta \tag{67}$$

where $U = u\Delta y$ with u the calculated velocity in u direction and Δy the grid cell width along the western cell wall, θ is the advected sea ice field, i.e. the ice or snow thickness or the ice concentration, $c = \frac{|u|\Delta t}{\Delta x}$ the CFL number, ψ the limiter function, and r the slope ratio. The used abbreviations are

$$\overline{\theta}_{i,j} = \frac{1}{2}(\theta_{i,j} + \theta_{i-1,j}) \tag{68}$$

$$\delta\theta_{i,j} = \theta_{i,j} - \theta_{i-1,j} \tag{69}$$

The slope ratio and the limiter function are

$$r = \begin{cases} \dfrac{\theta_{i-1,j} - \theta_{i-2,j}}{\theta_{i,j} - \theta_{i-1,j}} & \text{if } u_{i,j} > 0 \\ \dfrac{\theta_{i+1,j} - \theta_{i,j}}{\theta_{i,j} - \theta_{i-1,j}} & \text{if } u_{i,j} < 0 \end{cases} \tag{70}$$

$$\psi(r) = \max(0, \max(\min(1, 2r), \min(2, r))) \tag{71}$$

The change in the sea ice field θ is then

$$\Delta\theta_{i,j} = \frac{\Delta t}{A_c}(F_{i,j} - F_{i+1,j}) \tag{72}$$

After the advection, if the ice concentration is larger than 1, it is set to 1 to account for ridging.

5 Fortran to Python Translation

The MITgcm is written in Fortran 77. In the following sections, the for this work relevant differences between Fortran 77 and Python as well as the translation to Python is described. Also presented are structural changes that have been made to the code after the translation that work towards the coupling with Veros.

5.1 Translation to Python

When working with arrays, the difference in the indexation needs to be considered. In Fortran, the range of indices is manually defined when the array is initialized which also allows for negative indices. In Python, only the length of an array can be defined with the first index always being `0`. If in Fortran an array is defined with indices `1 - L` to `N + L`, this corresponds to an array with indices `0` to `N + 2L` in Python. The Fortran indices `1` and `N` then correspond to the Python indices `L` and `N + L`, respectively. Another difference when working with arrays is the use of vector operations in Python. While in Fortran 77 calculations need to be done for each element of an array individually using loops, in Python it is possible to directly use the array or slices of the array. If for example the Fortran code for the interpolation between two velocities

```
DO j=1,Ny
 DO i=1,Nx
  uAtC(i,j) = 0.5 * ( u(i+1,j) + u(i,j) )
 ENDDO
ENDDO
```

is translated to Python, the result is

```
uAtC[Ly:Ny+Ly, Lx:Nx+Lx] = 0.5 * ( u[Ly:Ny+Ly, Lx+1:Nx+Lx+1]
                                 + u[Ly:Ny+Ly, Lx  :Nx+Lx  ] )
```

Note that the order of indexation has been switched. This is because in Fortran multi-dimensional arrays are stored on memory is such a way that the most rapidly changing index comes first (F order). Arrays in Python are stored in C memory order, i.e. the most rapidly changing index comes last. In other words, in Fortran the first index (`i`) corresponds to the x-axis of a matrix and the second index (`j`) corresponds to the y-axis.

In Python, this is the other way round. This code can be made much more compact by using the `roll` function of NumPy which rolls the elements of an array backwards along a given axis. Elements that roll beyond the last position are reintroduced at the first one. The code is then

```
uAtC = 0.5 * ( roll(u,-1,axis=1) + u )
```

Using this method the whole array `u` is used. This has the advantage that `uAtC` does not need to be initialized separately.
Array operations that are dependent on `if` statements can be simplified by using NumPy's `where` function. If for example in the MITgcm the emissivity of the ice surface is set based on whether or not snow is present, the code is

```
_RL Emiss = (1:Nx, 1:Ny)
DO j=1,Ny
 DO i=1,Nx
  IF (hSnow(i,j) .GT. 0.0)
   Emiss(i,j) = SnowEmiss
  ELSE
   Emiss(i,j) = IceEmiss
 ENDDO
ENDDO
```

In Python this can be done in a single line:

```
Emiss = where(hSnow > 0, SnowEmiss, IceEmiss)
```

While in Fortran the array `Emiss` needs to be initialized first with its elements then individually set, the `where` function used in the Python implementation returns an array of the same shape as `hSnow` and fills it with `SnowEmiss` where `hSnow > 0` and with `IceEmiss` everywhere else.

5.2 Use of Object Oriented Programming

In the MITgcm and the first version of the model of this work, variables are passed individually between functions. This can get very confusing for more complex functions as some of them require more than 15 input variables and have similarly many outputs.

When such functions are nested within each other, it is difficult to keep track of variables and also to make changes in the code, e.g. if one wants to add or remove something as input or output. For this reason, Veros uses object oriented programming. The functions do not take in or give out many individual variables but work on a state object that contains all the variables. This drastically increases the readability of the code and therefore also the accessibility for new programmers. If functions are nested or reused with different inputs, e.g. in an iterative loop, it is still possible to have individual variables as input and output.

5.3 Testing the Model Against the MITgcm Reference

The testing of the model was split up in two parts. The thermodynamics and the dynamics were tested separately. For the thermodynamics, one dimensional benchmarks were used that only consisted of one grid cell. There were several benchmarks created with different constant forcings in both melting and freezing conditions. A direct comparison of the Python model and the MITgcm showed differences in the range of 10^{-7}. The modeling time was one year. The differences were random and did not accumulate over the modeling time. They were assigned to the differences in numerical precision.

After ensuring that the model works in the same way as the original one, some minor changes have been made. While in the original model only h_{actual} and not $h_{\text{s,actual}}$ has been regularized (see section 3), the model of this work uses the regularization for both the ice and the snow thickness.

The dynamics were compared on a grid with 65×65 cells with a grid cell width of 8000 m. For the forcing, wind and ocean surface velocities have been used separately and combined. Both the mEVP and the aEVP solver have been compared. It showed that for the aEVP solver, the differences between the results were one order of magnitude smaller than for the mEVP solver. In convergence, the aEVP solver yields deviations of the ice thickness, ice concentration, and ice velocities in the order of 10^{-3} when compared to the MITgcm.

6 Coupling With Veros

6.1 Order of Operations

The necessary order of operations for a coupled ocean-sea ice model is displayed in Fig. 2. At the beginning of the time step, the ocean surface forcing is calculated. This refers to the fluxes of heat, salt, and momentum into and out of the ocean, respectively. Afterwards the sea ice model (from now on called Veris for versatile ice simulation) is called. The sea ice growth is calculated based on the net ocean surface heat flux in regions of open water and on the ocean temperature and atmospheric state where there is ice present (see section 3). Alongside with the sea ice variables like the ice thickness and concentration, Veris also updates the heat, salt and momentum fluxes acting on the ocean surface. The heat flux is modified due to the release and absorption of latent heat during freezing and melting, respectively. The salt flux is altered so that the surface salinity increases (decreases) if sea ice is growing (melting). In regions of ice cover, the salt flux is also modified by precipitation and snowfall accumulating as snow on the ice surface instead of going into the ocean, as well as the evaporation under an ice cover being set to zero. The momentum flux on the ocean surface is calculated as the wind stress in regions of open water and the ice-ocean stress in regions of ice cover. After these changes in the ocean surface forcing, the ocean model is to be called. Because of this sequence a first attempt to include Veris as a plug-in to Veros failed because plug-ins of Veros are called at the end of the time step. Therefore all modifications of the surface forcing calculated by Veris would be overwritten by the surface forcing being calculated at the beginning of the next time step.

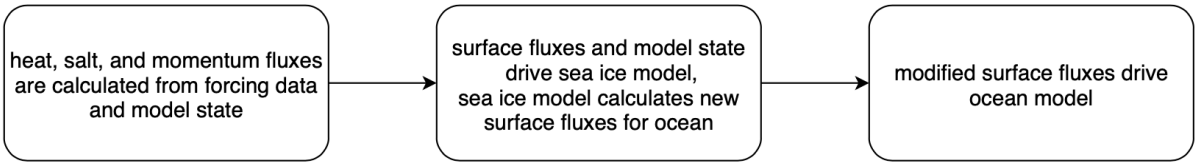

Figure 2: Required order of operations for coupling a coupled ocean-sea ice model.

6.2 Using the Original Surface Heat Forcing

In the original version of Veros, all of the surface heat forcing is contained in the single variable Q_{net} (net surface heat flux) that is imported as a prescribed field. With this approach, several problems arise when coupling the models.

Firstly, this surface heat flux is not suitable to drive a sea ice model. This is because it is an air-ocean heat flux that already assumes a sea ice cover in the respective regions. The presence of sea ice reduces the surface heat flux which can be seen in Fig. 3. Displayed is Q_{net} in February and in July. In general, if a variable is shown at a specific month and it is not stated otherwise, this always refers to the first day of that month. The heat flux is close to zero in regions where ice is expected, like the Baffin and Hudson Bay and the Nordic Seas in February and along the coast of the Antarctica in July. For the calculation of thermodynamic sea ice growth, an open water heat flux field would be needed (see section 3.1). This prescribed heat flux is too low and does not lead to accurate sea ice growth.

The sea ice growth is not only determined by Q_{net} but also by other variables. Many of these are not available in the original version of Veros, e.g. the atmospheric temperature and humidity. These additionally needed fields are taken from the ECWMF ERA5 reanalysis (Hersbach et al., 2020). With all of the necessary variables now being available, the problem is that Q_{net} as a prescribed field is independent of the current state of the model while it should be dependent on e.g. ocean surface and atmospheric temperature.

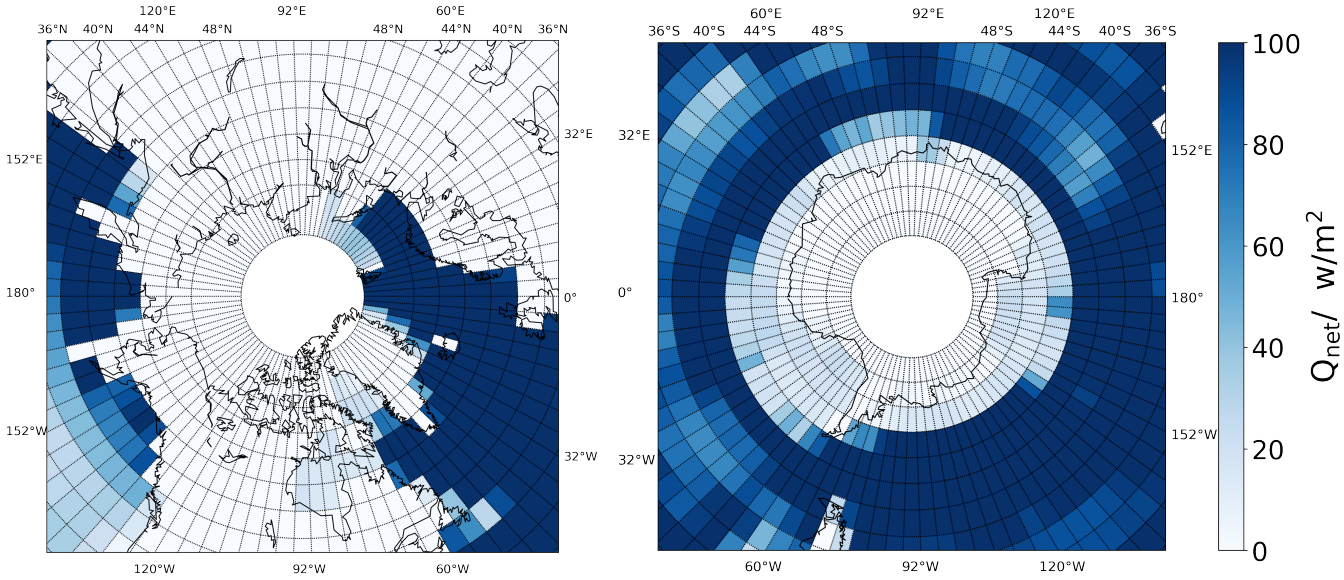

Figure 3: Prescribed net ocean surface heat flux in the northern hemisphere in February (left image) and in the southern hemisphere in July (right image). A positive sign means heat flux out of the ocean. In this data set an ice cover is assumed, therefore the heat flux is close to zero in the regions where sea ice is expected. The arctic region above 80°N is not part of the model region.

6.3 Implementing the Heat Flux Bulk Formula of the MITgcm

To overcome both of these issues, the heat flux bulk formula from the MITgcm is implemented that calculates Q_{net} from both the atmospheric state and the current state of the ocean model (Large and Yeager, 2009). The net heat flux is calculated as the sum of shortwave and longwave radiation and the sensible and latent heat flux:

$$Q_{\text{net}} = Q_{\text{sw,net}} + Q_{\text{lw,net}} + Q_{\text{lat}} + Q_{\text{sen}} \tag{73}$$

The net shortwave radiation $Q_{\text{sw,net}}$ also accounts for the ocean surface albedo. The net longwave radiation $Q_{\text{lw,net}}$ is the sum of the downward and upward longwave radiation. The upward longwave radiation is calculated from Stefan-Boltzmann's law according to

$$Q_{\text{lw,upward}} = \epsilon_{\text{ocean}} \sigma T_{\text{ocean}}^4 \tag{74}$$

where ϵ_{ocean} is the emissivity of the ocean, σ the Stefan-Boltzmann constant and T_{ocean} the ocean surface temperature. The latent heat flux Q_{lat} is calculated according to

$$Q_{\text{lat}} = \rho_a L_e v_w c_{\text{lat}} \Delta q \tag{75}$$

where L_e is the latent heat of evaporation, v_w the wind speed, c_{lat} the transfer coefficient of latent heat, and Δq the difference in specific humidity of the atmosphere and directly at the ocean surface. The sensible heat flux Q_{sen} is calculated according to

$$Q_{\text{sen}} = \rho_a c_p v_w c_{\text{sen}} \Delta T \tag{76}$$

where c_p is the specific heat of air, c_{sen} the transfer coefficient of sensible heat, and ΔT the difference in atmospheric temperature and ocean surface temperature.
From the latent heat flux the evaporation rate can be calculated. If this is not done, a prescribed evaporation field is used which is independent of the current model state. Calculating the evaporation rate from the latent heat flux and thus from the model state is therefore more consistent. The evaporation rate E is

$$E = \frac{Q_{\text{lat}}}{L_e \rho_w} \tag{77}$$

Using this heat flux and evaporation rate yields an ice distribution shown in Fig. 4 and 5. The ice distribution in February looks realistic with the Baffin and Hudson Bay being covered with ice and also some ice being present in the Nordic Seas. In July, however, the ice distribution is no longer physical with large gaps in front of the coast of the Antarctica.

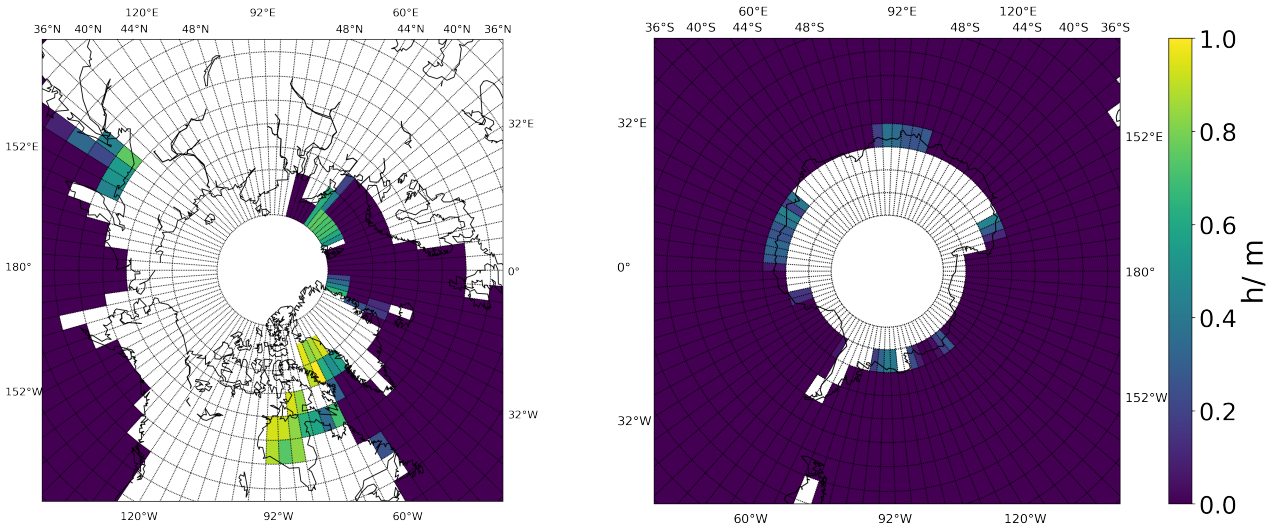

Figure 4: Ice thickness distribution in February with the surface heat flux calculated using the same bulk formula as the MITgcm.

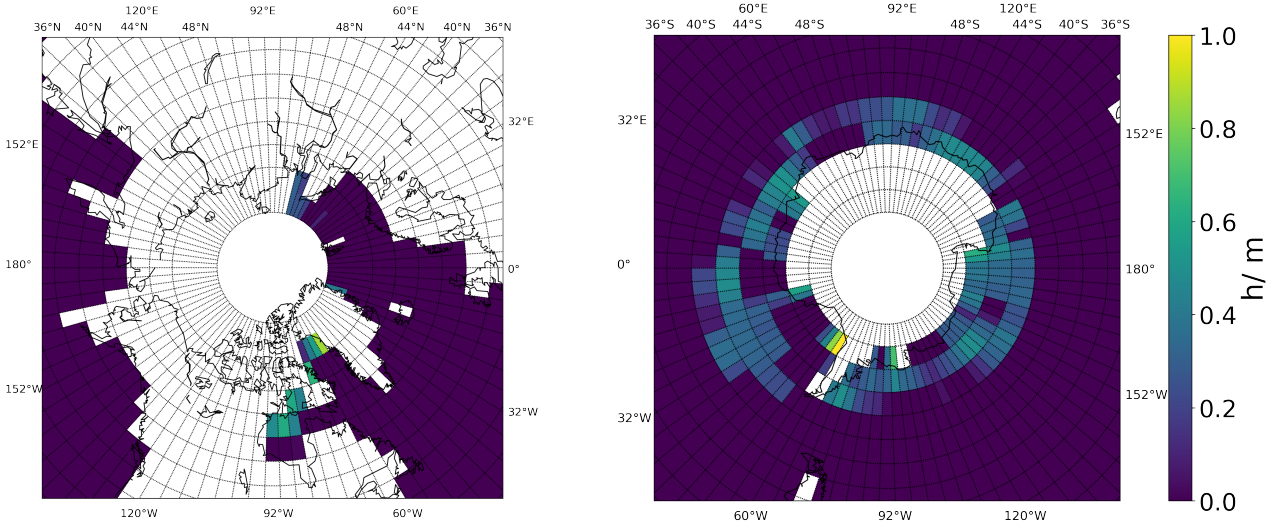

Figure 5: Ice thickness distribution in July with the surface heat flux calculated using the same bulk formula as the MITgcm.

The ice cover is expected to be continuous. As a reference, the MITgcm was run on the same grid with the same forcing data. The results are displayed in Fig. 6 and 7. There are no large gaps in the ice cover.

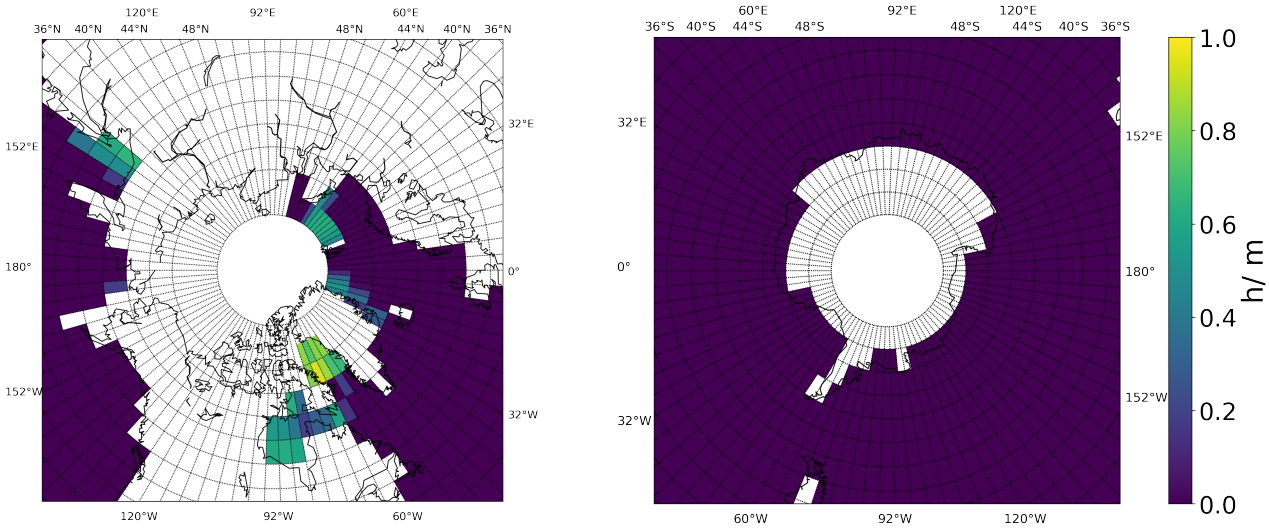

Figure 6: Reference ice thickness distribution in February calculated by the MITgcm using the same forcing data as the Veros run shown in Fig. 4 and 5.

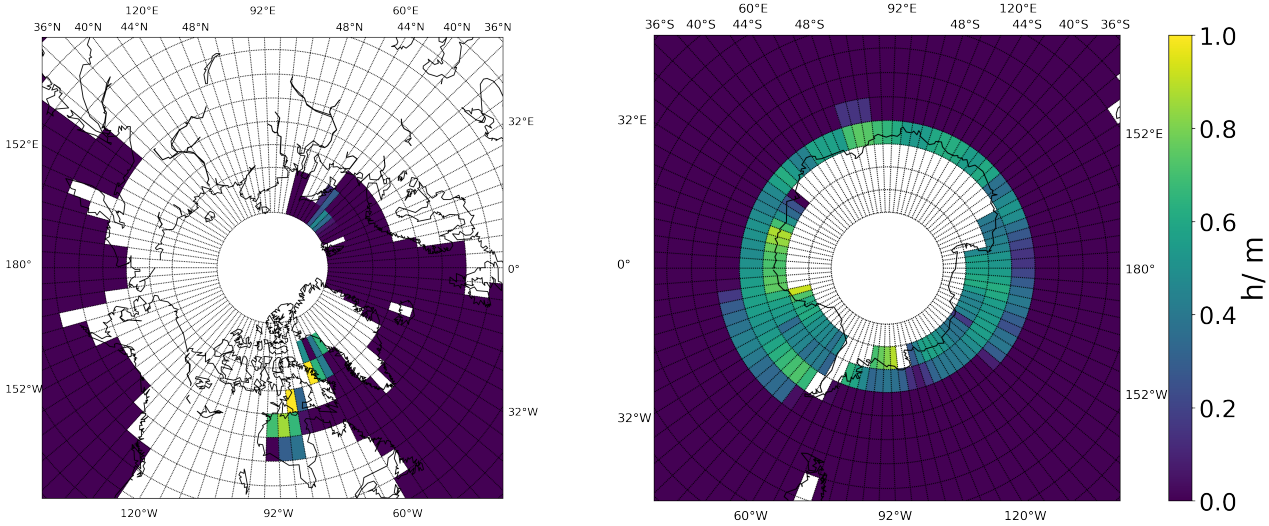

Figure 7: Reference ice thickness distribution in July calculated by the MITgcm using the same forcing data as the Veros run shown in Fig. 4 and 5.

The gaps in the ice distribution in Veros are due to an increased ocean surface temperature in these regions. This can easily be verified by setting the ocean surface temperature to a constant value which then allows for a continuous ice cover to be formed. The other forcing fields like the air temperature do not show a pattern with local anomalies whereas the ocean surface temperature (displayed in Fig. 8) shows the same pattern as the ice distribution with regions close to the coast of the Antarctica where the temperature is higher. In these regions, the ocean surface temperature is too high for sea ice to be formed, independently of the other forcing fields.

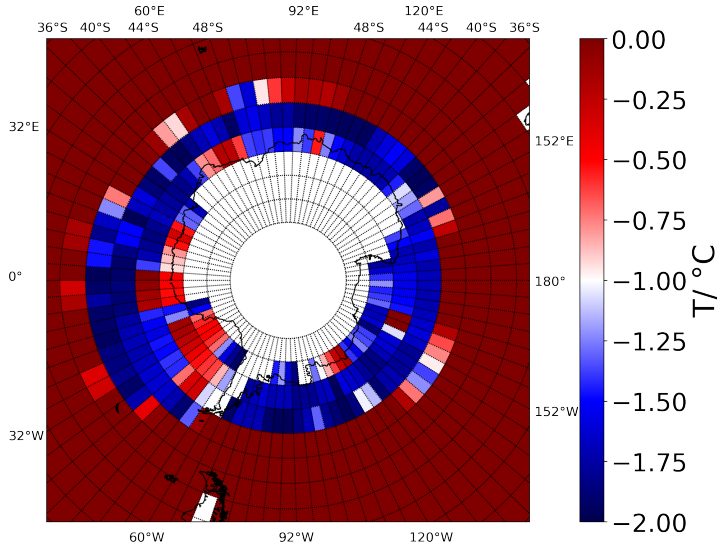

Figure 8: Ocean surface temperature in July with the surface heat flux calculated using the same bulk formula as the MITgcm.

6.4 Analysing Veros Using the Simple Sea Ice Mask

There is a simple sea ice model in the original version of Veros. When the ocean surface temperature is at or below the freezing point and the heat flux is negative, i.e. heat is leaving the ocean, Veros assumes an ice cover that inhibits any further cooling and sets the surface forcing to zero. This is realized by

```
mask = logical_and(temp < -2, forc_temp_surface < 0.0)
forc_temp_surface = where(mask, 0.0, forc_temp_surface)
forc_salt_surface = where(mask, 0.0, forc_salt_surface)
```

where `temp` is the ocean surface temperature, `forc_temp_surface` the heat flux and `forc_salt_surface` the salt flux. The ocean surface temperature in July using this sea

ice mask instead of Veris is displayed in Fig. 9. The ocean is at the freezing around the coast of the Antarctica with no isolated regions of higher temperature.

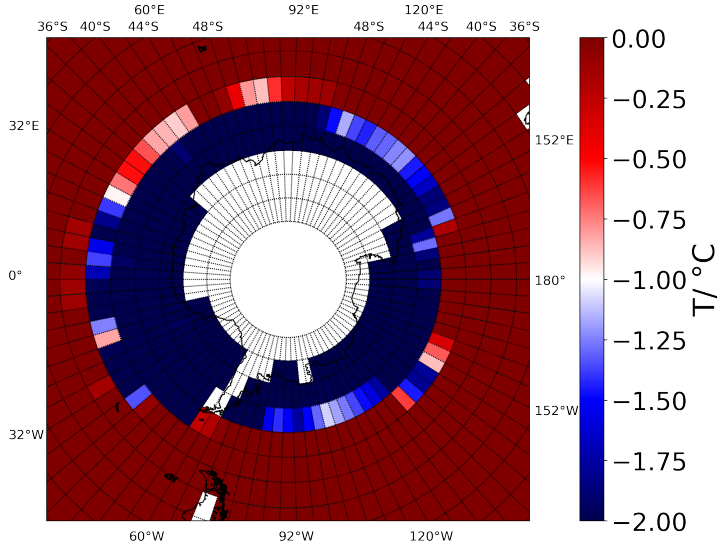

Figure 9: Ocean surface temperature in July using the simple sea ice model of the original version of Veros.

In this situation, the surface salt flux `forc_salt_surface` $= \frac{dS}{dt}$ is not calculated from forcing data like precipitation but is purely a restoring term that is calculated as a nudging towards a prescribed field of surface salinity:

$$\frac{dS}{dt} = \frac{1}{t_{res}}(S_{ref} - S) \tag{78}$$

where t_{res} is the restoring time scale, S_{ref} the given reference surface salinity, and S the actual surface salinity.

If the model is run with no surface salt flux, i.e. $\frac{dS}{dt} = 0$, the resulting surface temperature shows regions of larger surface temperature, displayed in Fig. 10. Fig. 11 shows the development of the surface temperature and salinity and the salt flux of one of the grid cells with a too high temperature. The left image is with salt restoring, the right without. The left image shows the expected development of the surface temperature as it is going down to and staying at the freezing point during southern winter. The salt restoring is zero or negative throughout the whole year which represent a freshwater flux lowering the salinity. Without this salt restoring, the salinity is higher and the temperature has a minimum at around -0.3°C. A possible reason for this could be vertical mixing due to the higher density of more saline water. After around 110 days, the temperature increases by

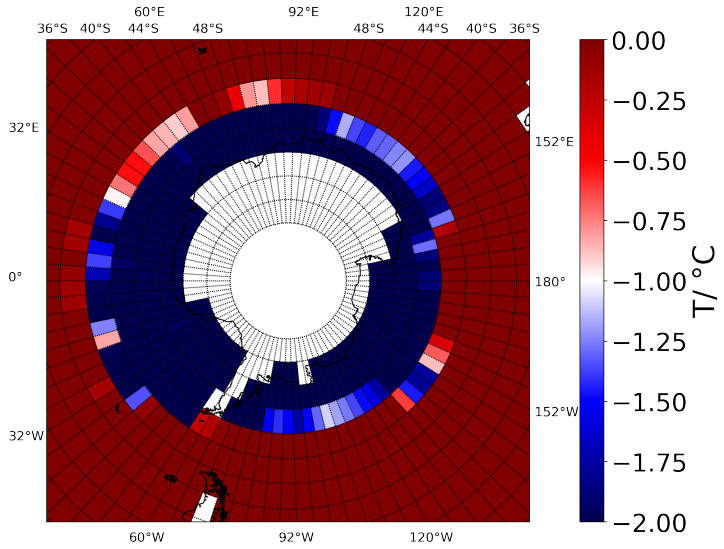

Figure 10: Ocean surface temperature in July using the simple sea ice model of the original version of Veros with the surface salt flux being set to zero.

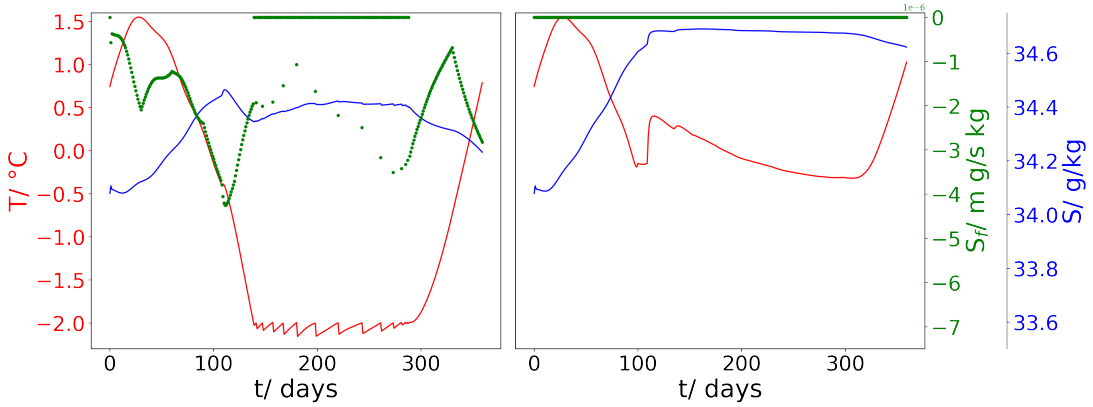

Figure 11: Ocean surface temperature T and salinity S and surface salt flux S_f for a specific grid cell. The y-axis for the left and the right plot are the same. During these model runs, the simple sea ice model of the original version of Veros was used, preventing the temperature from dropping below the freezing point. The left image is from a model run with salt restoring and the right image from a model run without.

0.5°C over a few days.

If this was due to the surface heat flux, this increase would also be present in the model run with salt restoring. It is thus most likely that this grid cell needs a negative salt flux in order for the temperature to reach the freezing point and remain there. The same analysis is done for other grid cells that have a too high temperature for sea ice formation. They show the same behaviour: with salt restoring, the salt flux is negative throughout

most of the year and the temperature is at the freezing point during winter. Without salt restoring, the salinity and temperature are higher. In some grid cells, a large sudden increase in temperature can be observed. An example of this behaviour is displayed in Fig. 12. This could be due to a density threshold that is reached at a certain salinity, leading to upward mixing of warm water. In fact, most grid cells with too high temperatures for ice formation show this sudden increase in temperature after staying at the freezing point for some time instead of the temperature always being too high as in the grid cell analysed in Fig. 11.

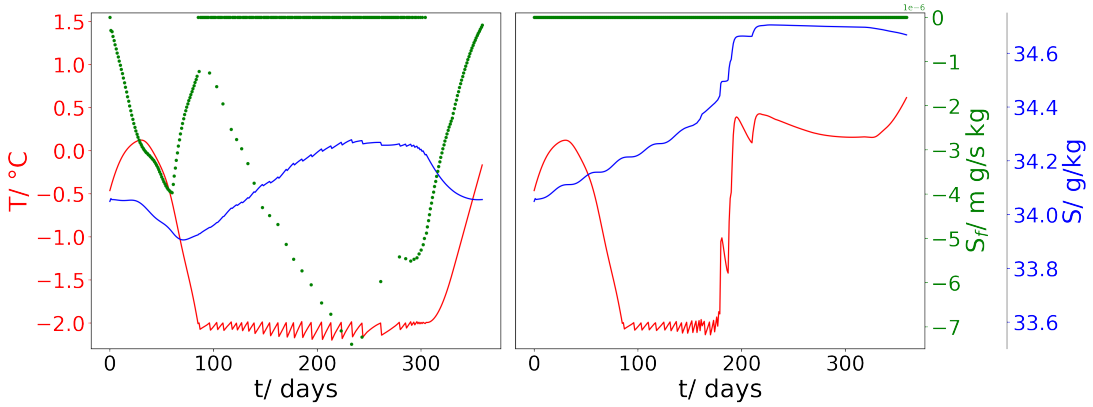

Figure 12: Analysis of another grid cell with a too high temperature, analogous to Fig. 11 (left: with salt restoring, right: without salt restoring).

6.5 Running Veris Combined With Salinity Restoring

In Veris, the salt flux into the ocean is calculated from the model state. In regions of no ice cover, it is determined by precipitation, evaporation and runoff. In regions of ice cover, the salt flux is determined by ice growth. For ice growth or melting, and melting of snow, the salt flux is proportional to the change in ice and snow thickness:

$$\frac{dS}{dt}_{\text{Veris}} = S\left((E-P)(1-A) - R - P_i A + \frac{\Delta h}{\Delta t}\frac{\rho_i}{\rho_w} + \frac{\Delta h_s}{\Delta t}\frac{\rho_s}{\rho_w}\right) \tag{79}$$

P is the precipitation rate, R the runoff rate, P_i the precipitation rate that flows through an ice cover, and $\frac{\Delta h}{\Delta t}$ and $\frac{\Delta h_s}{\Delta t}$ the changes in mean ice and snow thickness per time step. If there is an ice cover at some coast, the runoff is assumed to still flow into the ocean therefore it is not weighted with the ice concentration A. The term P_i is only non-zero in regions of precipitation and an ice surface temperature larger than the freezing point.

Note that the changes in ice and snow thickness are the changes in mean values and are already implicitly weighted by the ice concentration.

If Veris is run with this calculated salt flux, the surface temperature is too large in some specific grid cells leading to gaps in the ice cover as shown in section 6.3. In regions of ice growth, the salt flux is positive due to the $\frac{\Delta h}{\Delta t}\frac{\rho_i}{\rho_w}$ term. The analysis in section 6.4, however, led to the conclusion that some grid cells need a negative salt flux/ sufficiently low salinity for the surface temperature to reach the freezing point and remain there. The salt flux is therefore modified by adding the restoring term to it:

$$\frac{dS}{dt} = \frac{dS}{dt}_{Veris} + \frac{dS}{dt}_{res} \tag{80}$$

$\frac{dS}{dt}$ is the total salt flux that is used by Veros, $\frac{dS}{dt}_{Veris}$ the salt flux calculated by Veris according to equation (79), and $\frac{dS}{dt}_{res}$ the restoring salt flux calculated according to equation (78). The ice cover from a model run using this configuration is displayed in Fig. 13 and 14. It shows a continuous ice cover. Both in February and in July, there is more sea ice than in the MITgcm reference run. In regions of sea ice, the ice cover is larger, both in thickness and extent. In February, Veros has in contrast to the MITgcm also developed sea ice in the Southern Ocean.

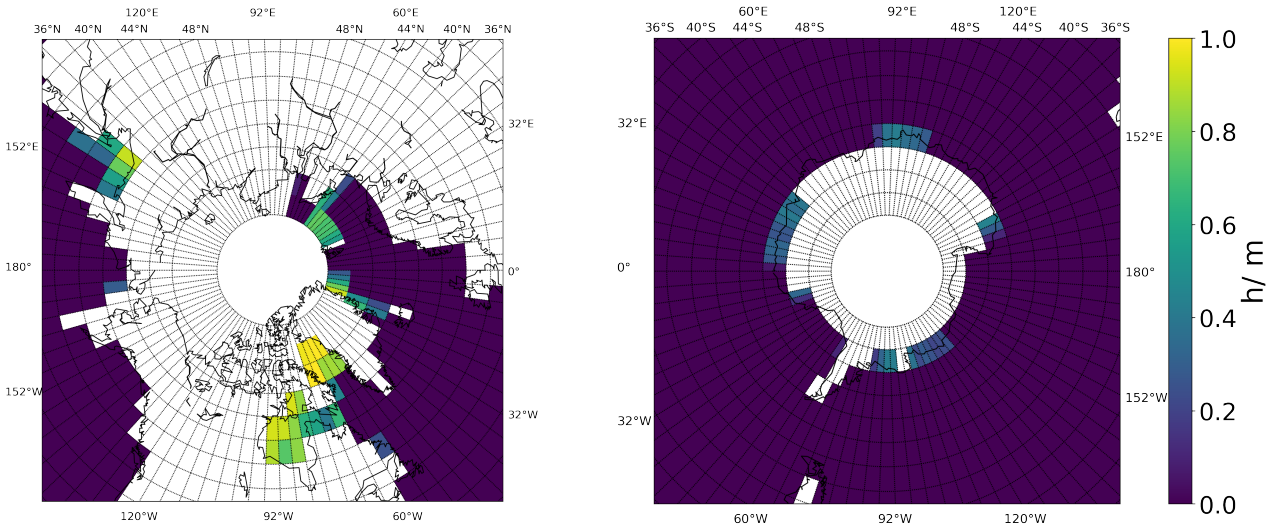

Figure 13: Sea ice distribution in February with the salt flux being calculated as the sum of the Veris salt flux and the restoring salt flux.

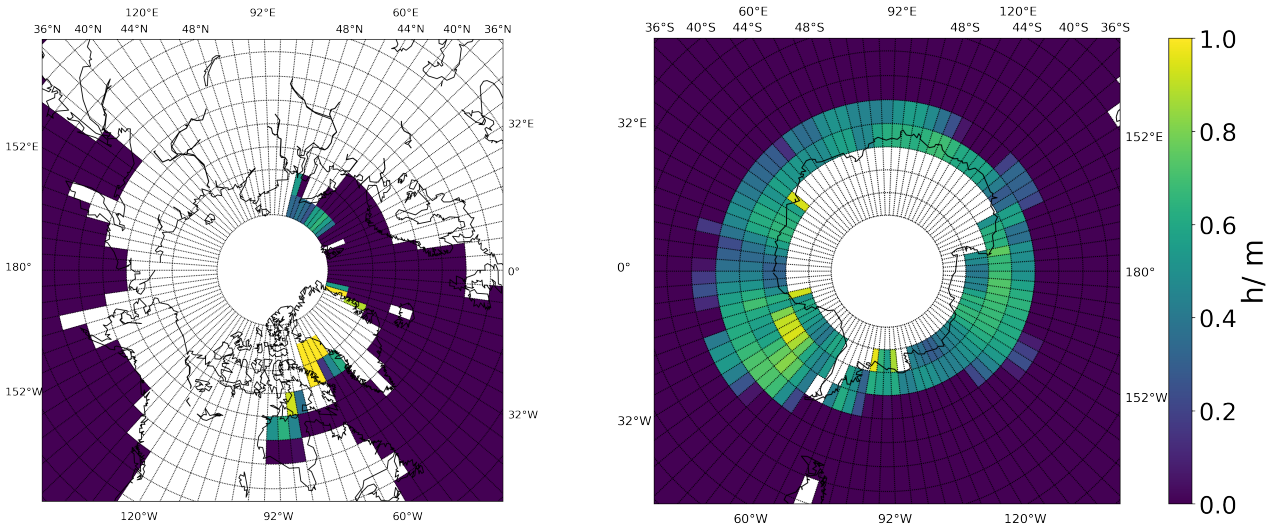

Figure 14: Sea ice distribution in July with the salt flux being calculated as the sum of the Veris salt flux and the restoring salt flux.

Ideally, the restoring term would only be needed in regions of no ice cover. The salt and freshwater fluxes between atmosphere and ocean in these regions are not based on conservation as there is no coupled atmosphere model. The atmospheric variables like temperature and humidity are imported from prescribed fields. In regions of ice cover, however, the salt fluxes should be modelled correctly. Here, the salt flux is determined by the interaction of ice and ocean which are interactively coupled and both part of the model. The salt flux here is thus based on conservation. Based on this, the salt flux is calculated according to

$$\frac{dS}{dt} = \frac{dS}{dt}_{Veris} + \frac{dS}{dt}_{res}(1 - A) \tag{81}$$

Note that $\frac{dS}{dt}_{Veris}$ is also given in regions of no ice cover where it is calculated from precipitation and evaporation (see equation (79)). The ice concentration weighting is already done in Veris, therefore $\frac{dS}{dt}_{Veris}$ does not need to be multiplied with A. The results of a model run using this configuration are displayed in Fig. 15, 16 and 17. The ice thickness is lower than in the configuration with the salt flux just being the restoring term but there are no large gaps in the ice cover in July. Later in the year, a large region of open water is formed in the Weddell Sea.

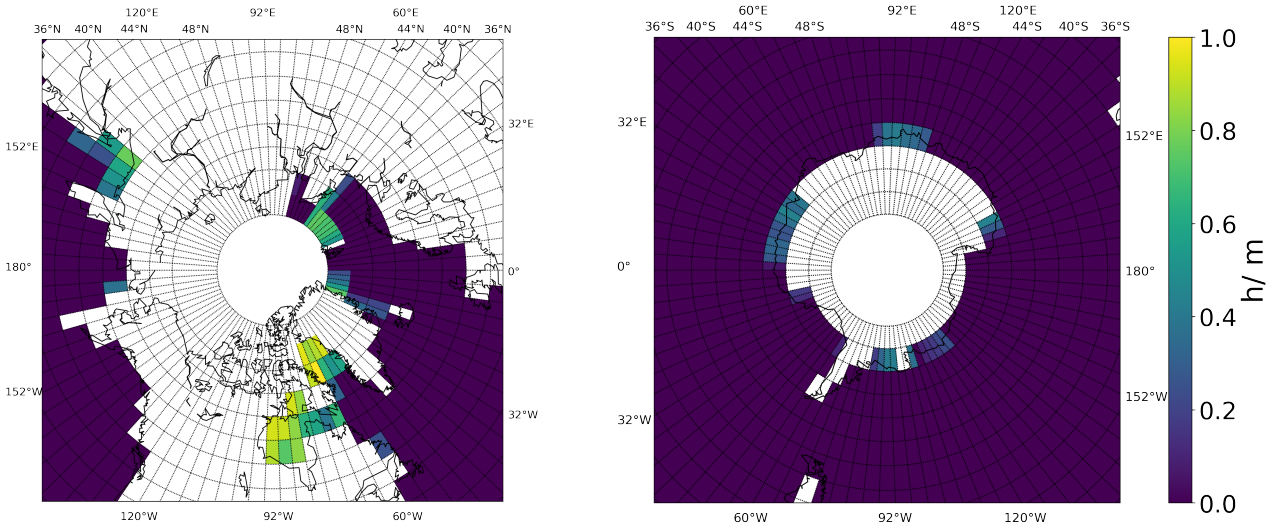

Figure 15: Sea ice distribution in February with the salt flux being weighted by the ice concentration: In regions of sea ice, the Veris salt flux is used. In regions of open water, the sum of the Veris salt flux and the restoring salt flux is used.

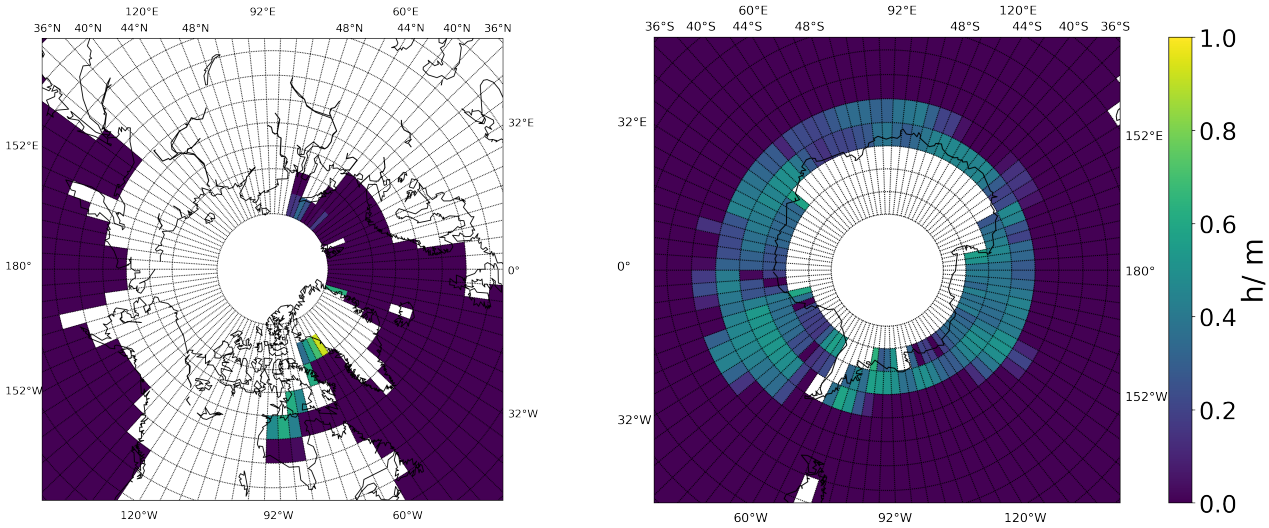

Figure 16: Sea ice distribution in July with the salt flux being weighted by the ice concentration: In regions of sea ice, the Veris salt flux is used. In regions of open water, the sum of the Veris salt flux and the restoring salt flux is used.

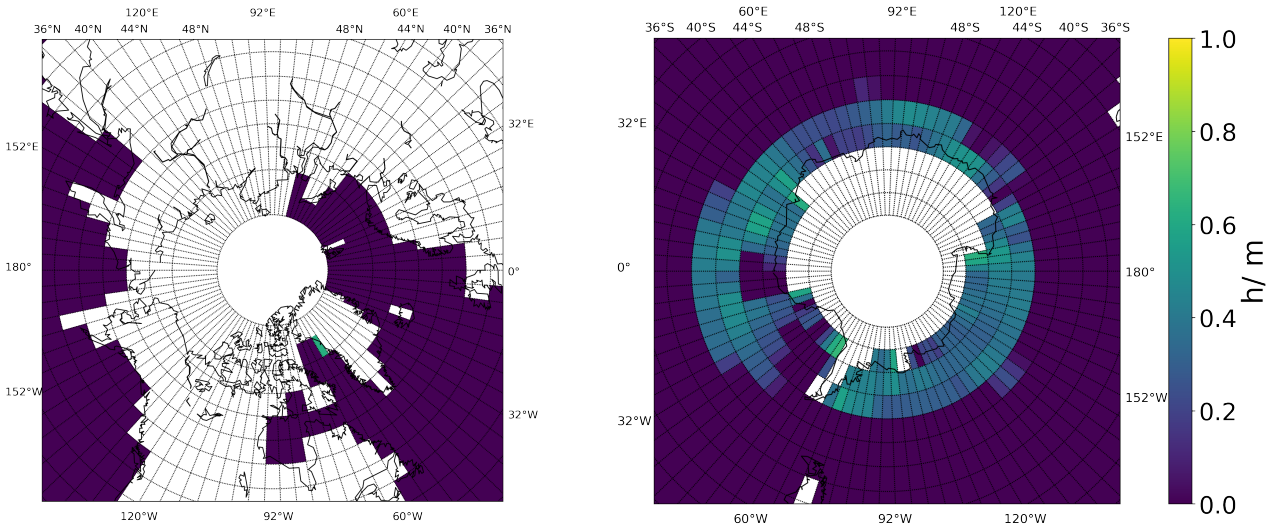

Figure 17: Sea ice distribution in September with the salt flux being weighted by the ice concentration: In regions of sea ice, the Veris salt flux is used. In regions of open water, the sum of the Veris salt flux and the restoring salt flux is used.

6.6 Comparison With ERA5 Data

When analyzing the output of Veros in this section, model runs with the salt flux into the ocean being calculated as the sum of the Veris salt flux and the restoring term according to equation (80) have been used. As a simple evaluation of the modelled sea ice distribution, the ice concentration is compared to the values taken from ERA5. The monthly mean values are displayed in Fig. 18 and 19. The forcing data for Veris was taken from ERA5 with the only variables imported from Veros being the ocean surface temperature, salinity and velocity. The ice thickness is not available in ERA5. The ice concentration in Veros has values close to one in the whole ice covered region with an abrupt drop to zero at the ice edge. In ERA5, the ice concentration steadily declines when approaching the ice edge with values around 0.4 in the outermost grid cells of the ice cover.

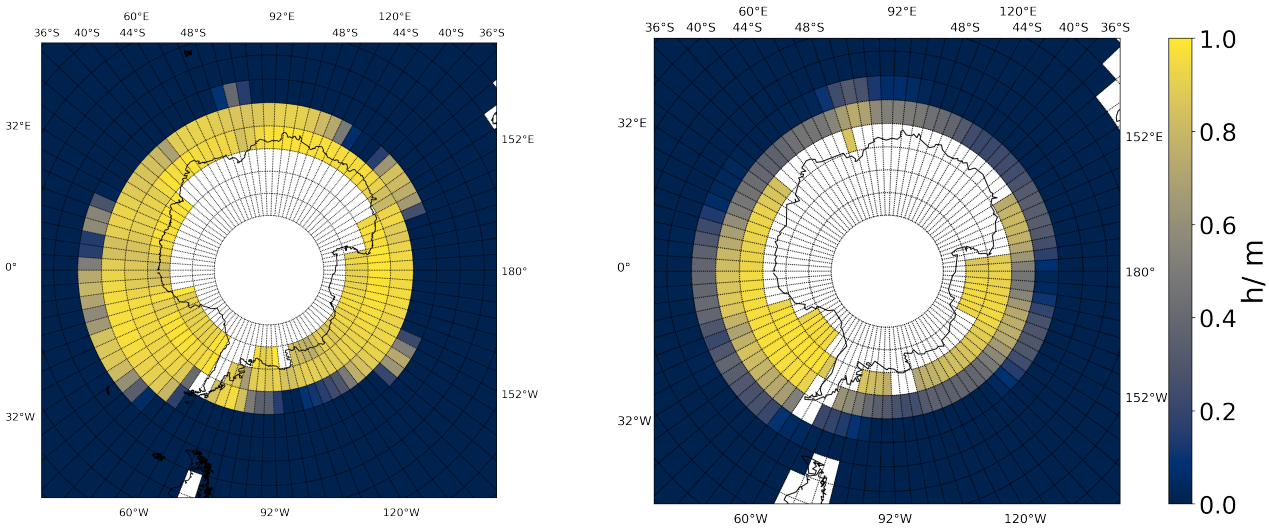

Figure 18: Monthly mean ice concentration modelled by Veros (left image) and taken from ERA5 (right image) around the Antarctica in July.

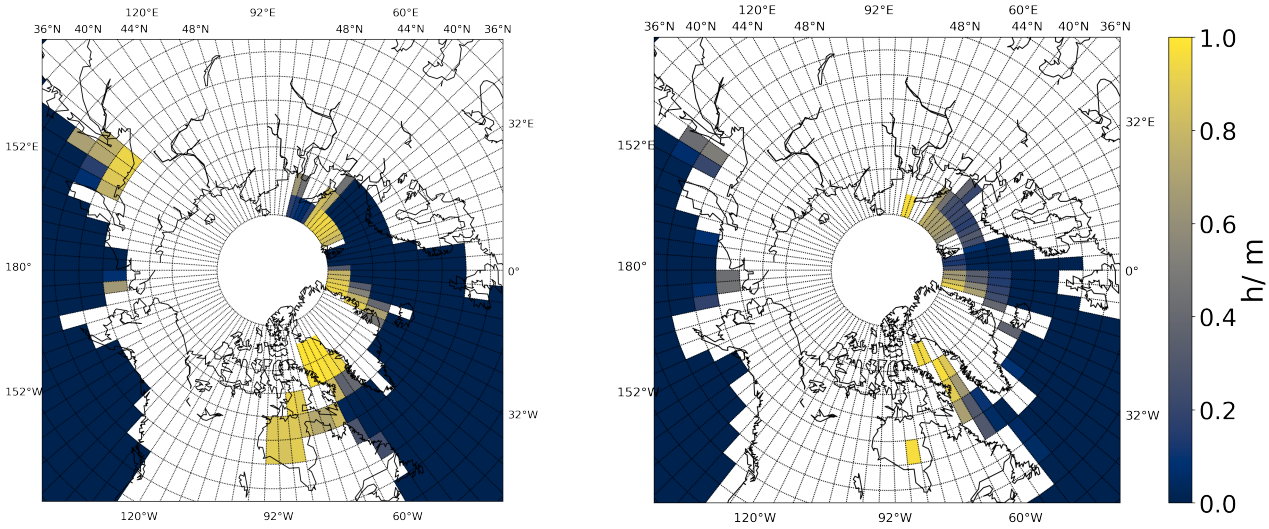

Figure 19: Monthly mean ice concentration modelled by Veros (left image) and taken from ERA5 (right image) in the northern hemisphere in February.

6.7 Multi-Year Runs

In the previous sections, the first modelling year was analyzed. The atmospheric forcing data of Veros is prescribed for one year. If Veros is run for longer, the forcing data of the first year is reused periodically. The sea ice is expected to reach an equilibrium at some point. Fig. 20 shows the mean ice thickness in the northern and southern hemisphere during the respective winter over a modelling time of 200 years. This refers to the mean ice thickness of the grid cells that are covered with sea ice. The mean ice thickness in the southern hemisphere is increasing in the first few years and then decreases to 49 cm after

100 years. The mean ice thickness in the northern hemisphere is continually increasing until it reaches an equilibrium at 3.39 m after 100 years. Analyzing the mean ice thickness only during the winter of the respective hemisphere yields the same result. The sea ice is expected to reach equilibrium much faster than 100 years. This would require an ocean in equilibrium which is not the initial condition of this model. Therefore the sea ice takes a longer time to reach equilibrium as the ocean model takes a long time period to spin up. This can be seen in Fig. 21 where the mean global ocean surface temperature is displayed. The ice thickness distribution after 100 years is displayed in Fig. 22 and 23.

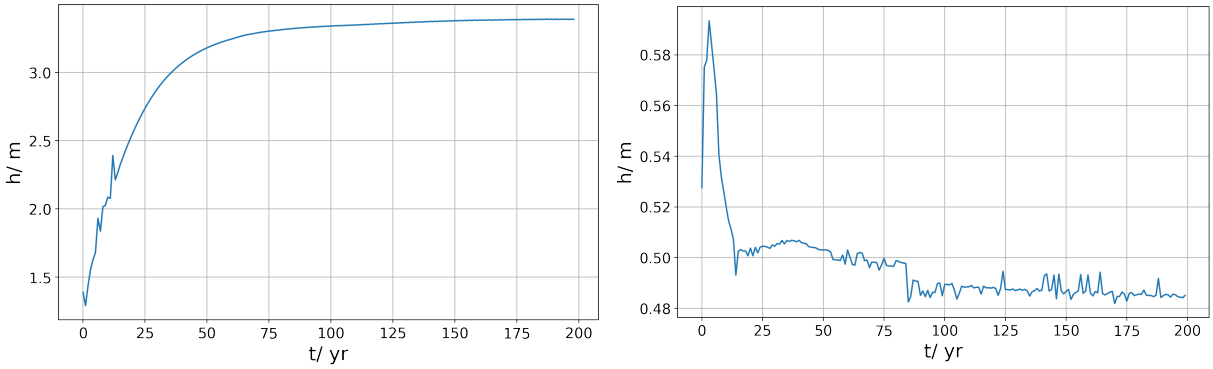

Figure 20: Mean ice thickness in the northern (left image) and southern (right image) hemisphere over a modelling time of 200 years.

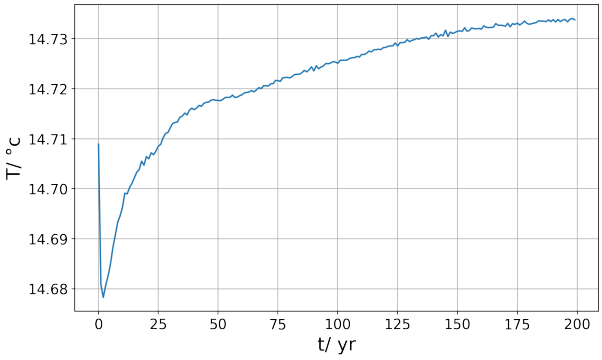

Figure 21: Mean global ocean surface temperature over a modelling time of 200 years.

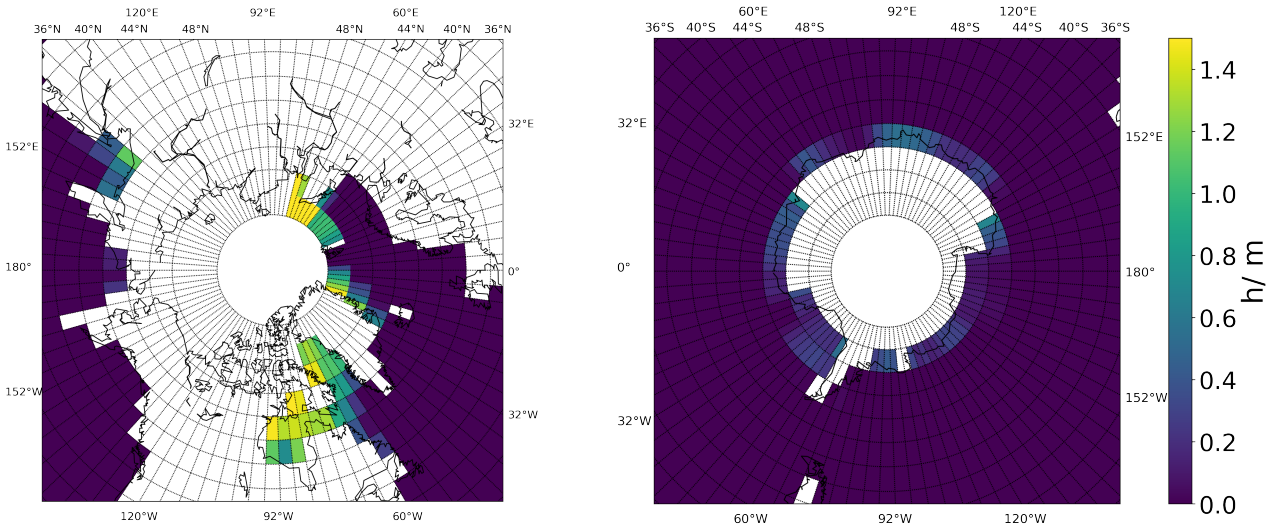

Figure 22: Mean ice thickness distribution in northern winter after 100 years.

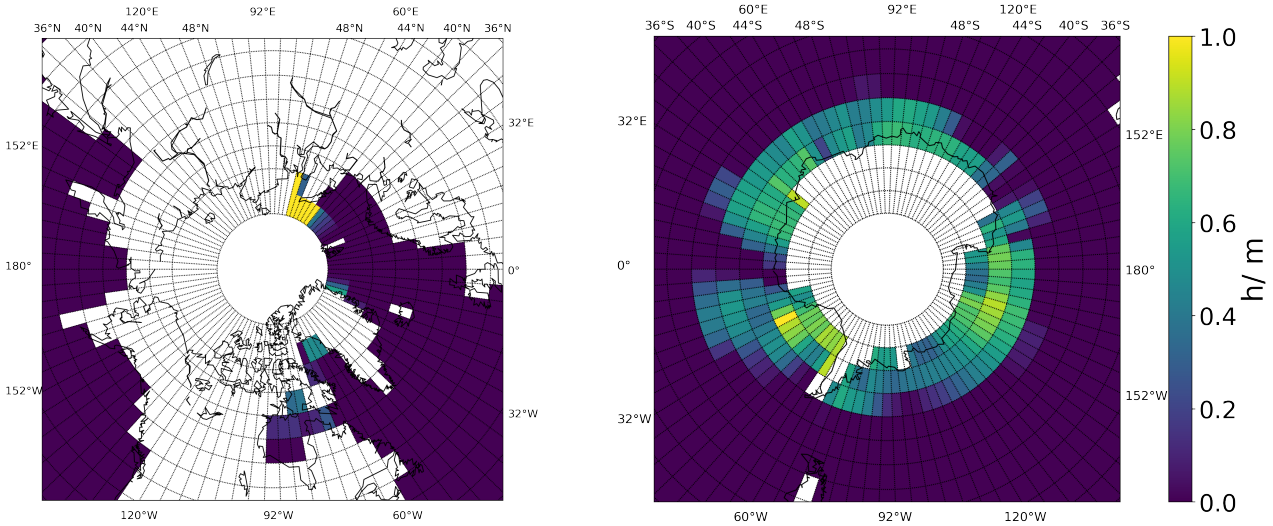

Figure 23: Mean ice thickness distribution in southern winter after 100 years.

There is a significant build up of ice in the northern hemisphere north of the Kara Sea. In this region the sea ice reaches a thickness of 8 m both in summer and in winter. This is due to the missing arctic region. Without an Arctic, the region north of Kara Sea is directly at a boundary. There are no ocean currents from the Arctic that could transport the sea ice to other regions, leading to a build up. The region also has a width of only one grid cell in latitude direction, the sea ice is therefore stuck between two boundaries.

7 Analyzing The Effects Of The Coastal Drag Parameterization To Simulate Landfast Ice

In Veris, two possibilities for the simulation of landfast ice are implemented. The formation of landfast ice due to ice keels reaching to the ground is parameterized by the basal drag coefficient c_b. The formation of landfast ice due to the ice being fixed to a nearby coast is either parameterized by a lateral drag coefficient c_l in combination with a free-slip boundary condition or by just using a no-slip boundary condition with no additional drag coefficient. The total drag coefficient is

$$c_d = \begin{cases} c_w + c_b & \text{using the no-slip boundary condition} \\ c_w + c_b + c_l & \text{using the free-slip boundary condition} \end{cases} \tag{82}$$

A comparison of the zonal ice velocities of a model run with the coastal drag parameterization to one without is shown in Fig. 24 and 25. In these model runs, the salt flux was calculated as the sum of the Veris salt flux and the restoring term (equation (80)). The ice velocities at a coast are drastically reduced. The reduction in velocity is larger when using the second form factor as it yields larger values of the drag coefficient (see section 4.4). A comparison of the resulting sea ice distribution to the one calculated without coastal drag is displayed in Fig. 26. The changes in the ice field are the smallest in the Ross Sea where the ice speed directly at the coast is very small in the configuration without coastal drag already. Adding the coastal drag parameterization does therefore not lead to large changes in the ice velocity here. The ice speed without coastal drag is largest south of the Indian Ocean, leading to the largest reduction in ice speed and ice thickness when adding the coastal drag parameterization. The ice thickness does not only change directly at the coast but also further out in the ocean. This is because if the velocity of the grid cells at the coast changes, the advection into their neighbouring cells further away from the coast also changes. In general, the ice thickness in the configuration using coastal drag is larger. Without coastal drag, the ice drifts away from the coast forming regions of open water. In open water the sea ice growth rate is the largest but as the ice drifts away from the coast, the mean ice thickness in these regions decreases. There is also ice drifts northward into regions of warmer water where it melts. This is why the increased ice production due to more open water does not lead to an accumulation of more sea ice. With coastal drag, the ice drift into regions where it would melt is reduced and the ice is

reaching larger thicknesses.

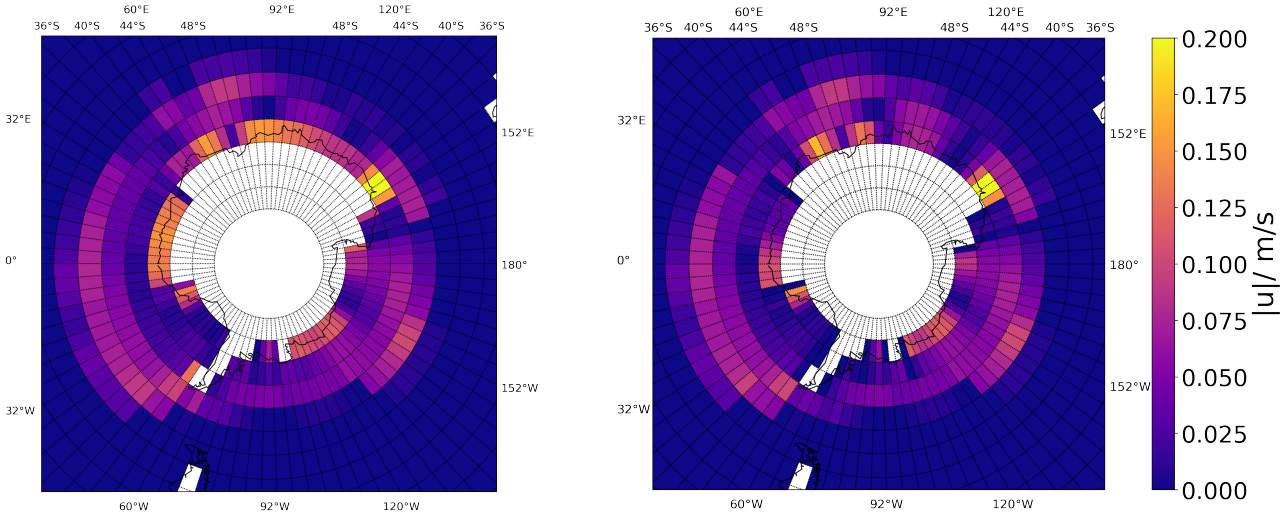

Figure 24: Left image: Mean ice speed from March to November without coastal drag. Right image: Mean ice speed with coastal drag using form factor F_1.

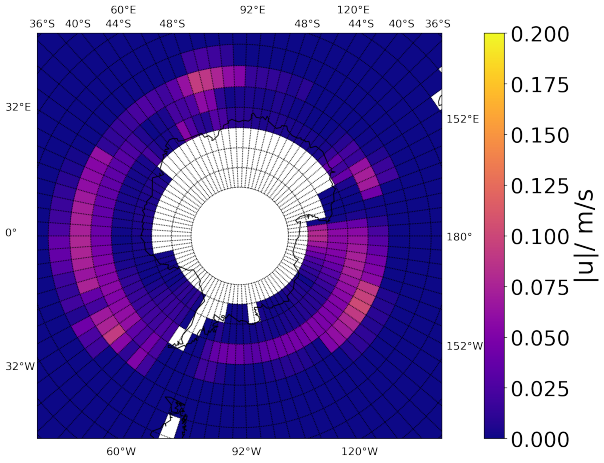

Figure 25: Mean ice speed from March to November with coastal drag using form factor F_2.

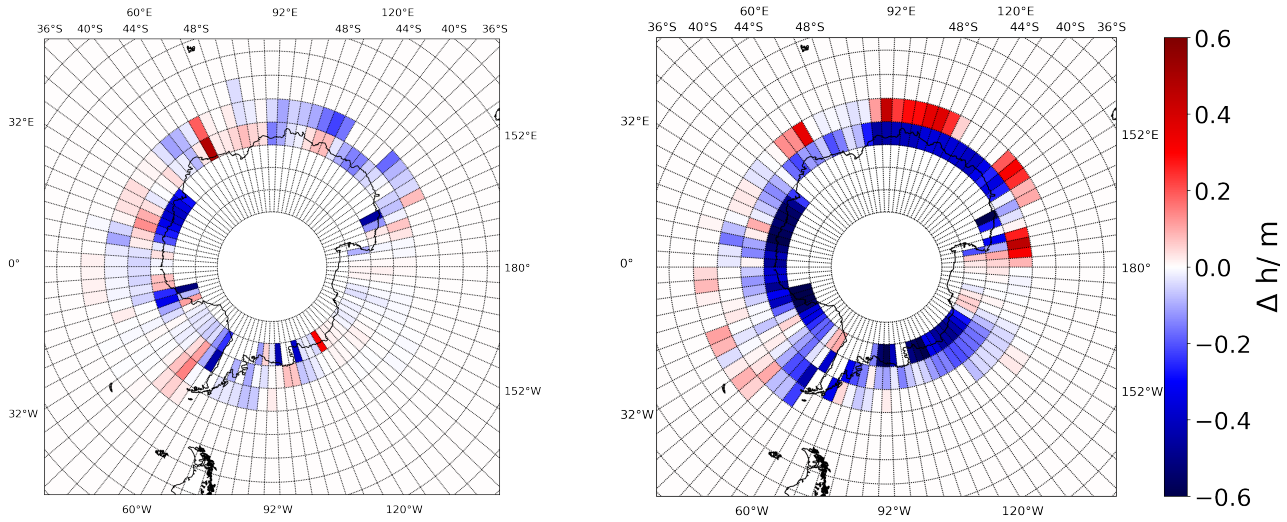

Figure 26: Comparison of the ice thickness in July calculated with coastal drag to the ice thickness calculated without coastal drag ($\Delta h = h - h_{\text{coastal drag}}$). Left image: use of form factor F_1. Right image: use of form factor F_2.

The criterion used by Liu et al. (2022) to categorize landfast ice is an ice velocity below $5 \cdot 10^{-4}$ m/s (corresponding to a drift of 600 m in two weeks) and an ice concentration larger than 95%. This criterion leads to the landfast frequency displayed in Fig. 27. It is important to note that the grid cell size used by Liu et al. (2022) is 36km whereas the grid cell size used in this work is 4°. With smaller grid cells, some cells close to the coast can be landfast ice while cells further away from the coast are still moving. If they are represented by a single larger grid cell, its mean velocity can be larger than the landfast ice criterion. This leads to an underestimation of landfast ice when using lower spatial resolutions. As a reference the landfast ice frequency obtained from a NASA Moderate Resolution Imaging Spectroradiometer (MODIS) derived data set with a spatial resolution of 1km x 1km (Fraser et al., 2020) is shown in Fig. 28. The landfast ice extent is small compared to the grid cell size of Veros (4° x 4°). Landfast ice is therefore not expected to be modelled correctly.

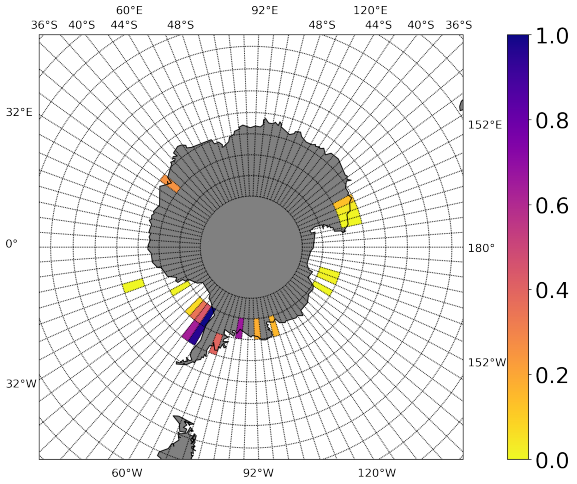

Figure 27: Landfast ice frequency between March and November (1 corresponds to the presence of landfast ice at every day).

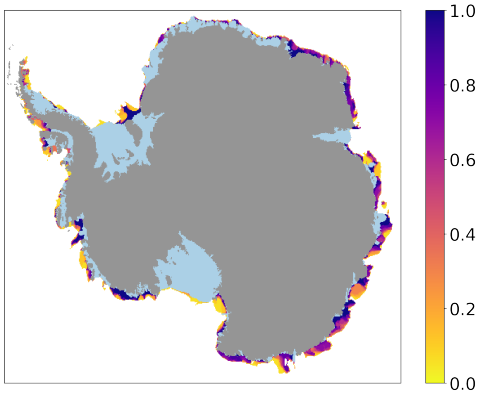

Figure 28: Landfast ice frequency between March and November (1 corresponds to the presence of landfast ice at every day). The areas in light blue are shelf ice. This data set is obtained by satellite remote sensing and has a spatial resolution of 1km x 1km (Fraser et al., 2020).

8 Summary

This work presented the underlying physics and its numerical implementation of a sea ice model as well as the translation of the model from Fortran to Python. The model was tested against the original in idealized scenarios with constant forcing before coupling it to the ocean model Veros. The sea ice distribution calculated from the coupled models was compared to the MITgcm which was run using the same forcing data. The ice cover of Veros without any additional tuning showed large gaps in the ice cover. This was due to a too high ocean surface temperature preventing sea ice growth. By analyzing the surface temperature and salinity of the original version of Veros using the simple sea ice mask as sea ice model, it was concluded that the ocean surface temperature is heavily dependent on the ocean surface salinity. Larger salinities are linked to larger temperatures in general and often to a sudden increase in temperature in a very short time period. This is most likely because due the higher density of more saline water, vertical mixing takes place which brings up warmer water from lower levels. To suppress this, a salinity restoring can be used that nudges the surface salinity towards a prescribes value. This salinity restoring is zero or negative in regions of sea ice growth leading to a lower surface salinity and no vertical mixing. This allows for the surface temperature to stay at the freezing point during winter. If Veros is run with Veris and this salinity restoring, the sea ice distribution is physical and does not show gaps of open water in the first modelling years. In longer runs the ice distribution does show gaps. It is possible to only use the salinity restoring in regions of open water. Veros does not contain a modelled atmosphere, therefore the atmosphere-ocean salt flux is not based on conservation. The salt flux is based on conservation in regions of ice cover where the salt flux is between the ice model and the ocean model. It is thus expected that these regions do not need nudging. This configuration showed much better results than the one with no nudging at all in the sense that the ice cover shows much less gaps. It is much thinner than expected though and still shows large gaps in the Weddell Sea.

A parameterization of coastal drag to simulate landfast ice is additionally implemented in Veris. It was shown that this has a large impact on the ice velocities as they were drastically reduced directly at a coast. This is the expected behaviour of landfast ice. The overall development of the ice distribution was strongly influenced by this with the ice thickness being generally larger when using coastal drag, especially directly at the

coast. The criterion for landfast ice used by Liu et al. (2022) was not suitable to diagnose landfast ice in the model runs of this work due to the larger grid cell size used by Veros.

9 Discussion

The goal of this work was to translate the sea ice component of the MITgcm into Python and to couple it to Veros. The coupling was successful in the sense that Veros is now able to calculate an ice cover in the expected regions. As a reference, the ice distribution is also calculated by the MITgcm using the same forcing data as Veros. The ice thickness calculated by Veros was larger than the one calculated by the MITgcm. For the ice cover to be continuous, a nudging in the salt flux is needed. If the salt flux is directly calculated from precipitation, evaporation and sea ice growth, the ocean surface temperature shows regions of higher temperature that are too high for sea ice to form. For Veros to yield physically sound results with a directly calculated salt flux, an in-depth analysis and probably modification of its dynamics of vertical mixing is required.

The grid used by Veros only extends to 80°N and does not cover the Arctic region. For the ocean in the Arctic to be modelled correctly, a coupled sea ice model is crucial as large parts of the Arctic Ocean are covered by sea ice. As Veros did not have a sea ice model yet, the Arctic could therefore not be modelled in a realistic form anyway. If a working sea ice model is implemented, a polar grid can be developed which then allows for ocean and sea ice modelling in the Arctic. For a comprehensive analysis of the Arctic Ocean and especially on how the dynamics of the Arctic Ocean change with respect to sea ice growth and changes in the sea ice field, Veros should be modified to be working without the salinity restoring. Veros still being dependent on the salinity restoring has two implications. The sea surface salinity of the Arctic has to be known throughout the modelling time. This is not directly a problem, as it can be imported from e.g. ERA5, similar to the atmospheric variables used in this work. If the surface salinity is calculated by a nudging term, however, the influence of sea ice growth and other processes on it over longer time scales can not be analyzed. The salt flux due to ice growth in a single time step can be calculated but the nudging term will prevent the salinity from deviating from its prescribed value. The dynamics of the ocean are dependent on the surface salinity by e.g. vertical mixing. If the surface salinity is prescribed and not modelled, the effects of sea ice on the ocean dynamics can therefore not be comprehensively analyzed. For this to be possible, the ocean surface fluxes of heat, salt, and momentum need to be calculated by the sea ice model from the model state. Still, using a configuration with a prescribed salinity allows for the analysis of the sea ice distribution.

Part of the motivation of developing a Python based model was that it is easier to implement new features than in a low-level language like Fortran. Implementing an additional landfast ice parameterization after the translation and coupling could be done very quickly. The new parameterization lead to the expected reduction in ice speed close to the coast. For a more comprehensive analysis a higher spatial resolution is required. This is generally true for evaluating model outputs and for comparing it to observational data but it is especially required when analyzing landfast ice. With too large grid cells the ice speed of a cell will most likely be larger than the landfast ice criterion, leading to an underestimation of landfast ice. It is possible to use a setup configuration of Veros that has a higher spatial resolution. This was not done in this work because in these setups the surface heat flux is treated in a different way, allowing for solar radiation to penetrate through the upper ocean layer. The calculation of the surface heat forcing of Veros was changed for this work as the original heat forcing was not suitable to drive a sea ice model. These changes would also need to be implemented in the configurations with a higher spatial resolution which requires more extensive work with Veros.

10 Outlook

The aspects of possible future work discussed in the last section were all about Veros: analysing and modifying the dynamics, including the Arctic Ocean in the model region, and using a higher spatial resolution. There is also scope for development in Veris, briefly presented in this section.

With Python being a very popular programming language in the scientific community, there are many libraries that can be used. In future work with Veris, other dynamics solvers can be imported and implemented.

There are multiple thermodynamics routines in the sea ice component of the MITgcm. The one translated for this work assumes that the sea ice has both a salinity and heat capacity of zero. There is another thermodynamics routine in the MITgcm that can be translated and implemented that includes both a non-zero heat capacity and salinity of sea ice. The salinity of the sea ice has an impact on the salt flux into the ocean that is created by freezing and melting. With a heat capacity of zero, no energy is needed for the warming and cooling of the ice leading to an instantaneous adjustment of the ice temperature. This leads to the sea ice immediately starting to melt in melting conditions and also to a faster freezing in freezing conditions. In reality some energy would be needed for the warming and cooling of the ice. This would lead to a delay between changes in the surrounding conditions and the respective reaction of the sea ice.

References

Arakawa, A. and Lamb, V. R. (1977). Computational design of the basic dynamical processes of the ucla general circulation model. *General circulation models of the atmosphere*, 17(Supplement C):173–265.

Bouillon, S., Fichefet, T., Legat, V., and Madec, G. (2013). The elastic–viscous–plastic method revisited. *Ocean Modelling*, 71:2–12.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Fraser, A., Massom, R., Ohshima, K., Willmes, S., Kappes, P., Cartwright, J., and Porter-Smith, R. (2020). Circum-antarctic landfast sea ice extent, 2000–2018. *Australian Antarctic Data Centre*, 10.

Gearheard, S., Matumeak, W., Angutikjuaq, I., Maslanik, J., Huntington, H. P., Leavitt, J., Kagak, D. M., Tigullaraq, G., and Barry, R. G. (2006). "it's not that simple": a collaborative comparison of sea ice environments, their uses, observed changes, and adaptations in barrow, alaska, usa, and clyde river, nunavut, canada. *AMBIO: A Journal of the Human Environment*, 35(4):203–211.

Häfner, D., Jacobsen, R. L., Eden, C., Kristensen, M. R., Jochum, M., Nuterman, R., and Vinter, B. (2018). Veros v0. 1–a fast and versatile ocean simulator in pure python. *Geoscientific Model Development*, 11(8):3299–3312.

Häfner, D., Nuterman, R., and Jochum, M. (2021). Fast, cheap, and turbulent—global ocean modeling with gpu acceleration in python. *Journal of Advances in Modeling Earth Systems*, 13(12):e2021MS002717.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., et al. (2020). The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049.

Hibler, W. and Ip, C. (1995). The effect of sea ice iheology on arctic buoy drift. *ASME APPLIED MECHANICS DIVISION-PUBLICATIONS-AMD*, 207:255–255.

Hibler, W. D. (1979). A dynamic thermodynamic sea ice model. *Journal of physical oceanography*, 9(4):815–846.

Hunke, E. C. and Dukowicz, J. K. (1997). An elastic–viscous–plastic model for sea ice dynamics. *Journal of Physical Oceanography*, 27(9):1849–1867.

Kimmritz, M., Danilov, S., and Losch, M. (2015). On the convergence of the modified elastic–viscous–plastic method for solving the sea ice momentum equation. *Journal of Computational Physics*, 296:90–100.

Kimmritz, M., Danilov, S., and Losch, M. (2016). The adaptive evp method for solving the sea ice momentum equation. *Ocean Modelling*, 101:59–67.

König Beatty, C. and Holland, D. M. (2010). Modeling landfast sea ice by adding tensile strength. *Journal of Physical Oceanography*, 40(1):185–198.

Large, W. and Yeager, S. (2009). The global climatology of an interannually varying air–sea flux data set. *Climate dynamics*, 33:341–364.

Lemieux, J.-F., Tremblay, B., Sedláček, J., Tupper, P., Thomas, S., Huard, D., and Auclair, J.-P. (2010). Improving the numerical convergence of viscous-plastic sea ice models with the jacobian-free newton–krylov method. *Journal of Computational Physics*, 229(8):2840–2852.

Lemieux, J.-F., Tremblay, L. B., Dupont, F., Plante, M., Smith, G. C., and Dumont, D. (2015). A basal stress parameterization for modeling landfast ice. *Journal of Geophysical Research: Oceans*, 120(4):3157–3173.

Liu, Y., Losch, M., Hutter, N., and Mu, L. (2022). A new parameterization of coastal drag to simulate landfast ice in deep marginal seas in the arctic. *Journal of Geophysical Research: Oceans*, 127(6):e2022JC018413.

Losch, M., Menemenlis, D., Campin, J.-M., Heimbach, P., and Hill, C. (2010). On the formulation of sea-ice models. part 1: Effects of different solver implementations and parameterizations. *Ocean Modelling*, 33(1-2):129–144.

Marshall, J., Adcroft, A., Hill, C., Perelman, L., and Heisey, C. (1997). A finite-volume, incompressible navier stokes model for studies of the ocean on parallel computers. *Journal of Geophysical Research: Oceans*, 102(C3):5753–5766.

Massom, R. A., Scambos, T. A., Bennetts, L. G., Reid, P., Squire, V. A., and Stammerjohn, S. E. (2018). Antarctic ice shelf disintegration triggered by sea ice loss and ocean swell. *Nature*, 558(7710):383–389.

McPhee, M. G. (1992). Turbulent heat flux in the upper ocean under sea ice. *Journal of Geophysical Research: Oceans*, 97(C4):5365–5379.

Parkinson, C. L. and Washington, W. M. (1979). A large-scale numerical model of sea ice. *Journal of Geophysical Research: Oceans*, 84(C1):311–337.

Roots, E. (1989). Climate change: high-latitude regions. *Climatic Change*, 15(1):223–253.