

MITgcm SEAICE manual

Tim Hill

August 17, 2018

Contents

1	Introduction	2
2	Setting up the MITgcm	2
2.1	Basic tutorial	2
2.2	SEAICE tutorial	3
3	More details about running MITgcm	4
3.1	Available packages	4
3.1.1	DIAGNOSTICS	4
3.1.2	SEAICE package	6
3.1.3	EXF package	7
3.2	Core model parameters and recommendations	9
3.2.1	Equation of state	9
3.2.2	Checkpoint files	9
3.2.3	Time stepping	10
3.2.4	File input and output	10
3.2.5	Gridding parameters	11
4	What didn't work	12
4.1	Advection schemes	12
4.2	Surface winds	13
5	Results: cases studied	14
5.1	Ice advection	14
5.1.1	Results	16
5.1.2	What was learned	19
5.2	Ice melting	20
5.2.1	Results	22

5.2.2	What was learned	26
5.2.3	Next steps	28
5.3	Solar forcing	28
5.3.1	Results	30
5.3.2	What was learned	32
5.3.3	Next steps	33
5.4	Periodic forcing	33
5.4.1	Results	34
5.4.2	Run with constant temperature	35
5.4.3	What we learned	37
5.4.4	Next steps	37
5.5	1D comparison	38
5.5.1	Comparison case	38
5.5.2	Comparison results	39
5.5.3	Next steps	39
6	Recurring issues	41
6.1	NetCDF/HDF5 libraries	41
6.1.1	Disk space or quota	41
6.1.2	Modules	41
6.2	MITgcm python utils	42
7	Physics of the model	42
7.1	Radiative cooling	42
7.2	Water surface energy budget	43
7.3	Surface albedo	44
7.4	Thermodynamics	44
7.5	Total kinetic energy	45
7.6	Dynamical ice model	45
7.7	Radiation	46
7.8	Humidity	46
8	Visualization tools	48
9	Conclusions	48
10	Next steps	49

1 Introduction

This document aims to serve as a starting point for running the MITgcm in configurations with ice. I have the model at a functional point and hope to leave some resources so it is easy to pick up where I left off.

The first section serves as an introduction to MITgcm and should be a good resource to carry out the first few functional model runs, including running with the ice package.

The next few sections outline some specific things I learned about the model, and may be useful to someone trying to extend the model past what is outlined here. Some specific model configurations that did not work are explained, and problems I encountered with the model or with the Graham environment are discussed. Details about the model's physical parameterizations are included in the last section.

2 Setting up the MITgcm

The most useful references for the MITgcm are:

- The fluids wiki has a wealth of information about computing resources, running on the SHARCNET cluster, and has an MITgcm tutorial (discussed later).
- The MITgcm homepage
- The new MITgcm documentation is on read the docs, but is incomplete at the time of writing. The old documentation is still accessible but has some unfortunate qualities, including embedding tables as images so they are not searchable.
- The MITgcm github repo is where I usually look at and search through the model code if I need to.

2.1 Basic tutorial

The Fluids wiki tutorial is a good tutorial to start with the model. The tutorial was originally written by Emily Tyhurst, and I have corrected a few mistakes in the tutorial.

The tutorial is largely self-contained, but I want to say a word about directory structure. MITgcm expects your case files to be contained in the same directory as the model. For example, if you make a folder called `simulation1`

for your simulation, this folder might be in the same directory as the MITgcm directories `doc/`, `eesupp/`, `pkg/`, `model/`, etc., in which case the directory structure would look like

```
1 /home/user/MITgcm
2
3     simulation1/
4
5     doc/
6     eesupp
7     jobs/
8     lsopt/
9     model/
10    optim/
11    pkg/
12    tools/
13    utils/
14    verification/
15    ...
```

I had a folder I called `MITgcmdata/` which was a git repo that contained all my model runs. Then my directory tree was

```
1 /home/user/MITgcm
2
3     MITgcmdata/
4         simulation1
5         simulation2
6         ...
7
8     doc/
9     eesupp
10    jobs/
11    lsopt/
12    model/
13    optim/
14    pkg/
15    tools/
16    utils/
17    verification/
18    ...
```

Having a structure like this makes compiling the model much easier. You can in theory compile the model from anywhere as long as you point the `genmake2` script to the model directories, but in practice this has been harder than expected.

2.2 SEAICE tutorial

The SEAICE package (sec 3.1.2) provides a dynamic and thermodynamic sea-ice model for the MITgcm. This package is general purpose enough to model freshwater and ocean ice. The package does a good job (at least qualitatively) of recreating dynamics and ice quantities reasonably physically.

I have created a SEAICE tutorial that will be put on the fluids wiki. The tutorial walks you through a few cases, and has you design your own model run at the end. The data files from the tutorial should be a good starting point to base case files for more complicated experiments on.

3 More details about running MITgcm

This section tries to describe aspects of running the MITgcm with more useful information than the documentaion. Hopefully this will be a good resource when trying to extend the model past what is included in the tutorials.

3.1 Available packages

MITgcm is structured as a core model + packages. The only packages within my scope are the seaice, external forcing, calendar, and diagnostic packages. For each package, it must be enabled at both compile-time and at run-time. At compile-time, the package must be included in the `packages.conf` file. Then at run-time the package must be included in the packages namelist in `data.pkg`. For example, see listing 1.

```
1 # Packages
2 &PACKAGES
3 useEXF=.TRUE.,
4 useCAL=.TRUE.,
5 # useTHSICE=.TRUE.,
6 useSEAICE=.TRUE.,
7 useDIAGNOSTICS=.TRUE.
8 &
```

Listing 1: Sample `data.pkg` file. This would enable the EXF, CAL, SEAICE, and DIAGNOSTICS packages, and disable the THSICE package.

The rest of this section describes the additional packages available for MITgcm.

3.1.1 DIAGNOSTICS

The diagnostics package exists to make it easier to control the output files generated by the model. It is easy to choose which fields are outputted and how frequently files are saved. As usual, the package requires compile-time and run-time options.

Compile time options

At compile time (probably in your `code` directory), you should have the file `DIAGNOSTICS_SIZE.h` (example below). This can be copied from the `DIAGNOSTICS` package code, `pkg/diagnostics/DIAGNOSTICS_SIZE.h`.

```
1 C      Diagnostics Array Dimension
2 C
3 C      ndiagMax    :: maximum total number of available diagnostics
4 C      numlists    :: maximum number of diagnostics list (in data.diagnostics)
5 C      numperlist  :: maximum number of active diagnostics per list (data.
6 C          diagnostics)
7 C      numLevels   :: maximum number of levels to write     (data.diagnostics)
```

```

7 C      numdiags   :: maximum size of the storage array for active 2D/3D
8 C      diagnostics
9 C      nRegions   :: maximum number of regions (statistics-diagnostics)
10 C     sizRegMsk  :: maximum size of the regional-mask (statistics-diagnostics)
11 C     nStats      :: maximum number of statistics (e.g.: aver,min,max ...)
12 C     diagSt_size:: maximum size of the storage array for statistics-diagnostics
13 C Note : may need to increase "numdiags" when using several 2D/3D diagnostics ,
13 C and "diagSt_size" (statistics-diags) since values here are deliberately small.
14     INTEGER      ndiagMax
15     INTEGER      numlists , numperlist , numLevels
16     INTEGER      numdiags
17     INTEGER      nRegions , sizRegMsk , nStats
18     INTEGER      diagSt_size
19     PARAMETER( ndiagMax = 500 )
20     PARAMETER( numlists = 25, numperlist = 50, numLevels=2*Nr )
21     PARAMETER( numdiags = 25*Nr )
22     PARAMETER( nRegions = 0 , sizRegMsk = 1 , nStats = 4 )
23     PARAMETER( diagSt_size = 10*Nr )
24
25
26 CEH3 ;;; Local Variables: ***
27 CEH3 ;;; mode: fortran ***
28 CEH3 ;;; End: ***

```

Listing 2: Example DIAGNOSTICS_SIZE.h file

Parameter **numlists** (line 20) and **numdiags** (line 21) can be adjusted if you need more diagnostics.

Run time options

The diagnostics you want to output, and how frequently to write files are specified at runtime in file **data.diagnostics**. Specify the field names in **field** and **filename** lines, and set the frequency. See the following example. The file needs namelists **&DIAGNOSTICS_LIST** (for writing an entire field to file) and **DIAG_STATIS_PARMS** (for writing per-level statistics), even if one of them is empty.

```

1 # Diagnostic Package Choices
2 #
3 # for each output-stream:
4 #   filename(n) : prefix of the output file name (only 8.c long) for outp.stream n
5 #   frequency(n):< 0 : write snap-shot output every |frequency| seconds
6 #           > 0 : write time-average output every frequency seconds
7 #   timePhase(n) : write at time = timePhase + multiple of |frequency|
8 #   averagingFreq(n) : frequency (in s) for periodic averaging interval
9 #   averagingPhase(n): phase      (in s) for periodic averaging interval
10 #  repeatCycle(n) : number of averaging intervals in 1 cycle
11 #  levels(:,n) : list of levels to write to file (Notes: declared as REAL)
12 #                  when this entry is missing, select all common levels of this
13 #                  list
13 #  fields(:,n) : list of diagnostics fields (8.c) (see "available_diagnostics.log"
14 #                  file for the list of all available diag. in this particular
14 #                  config)
15 #
16 #&DIAGNOSTICS_LIST
17 # diag_mnc    = .FALSE. ,
18 # dumpAtLast = .TRUE. ,
19 #
20 #   frequency(1) = -3600.0,
21 #   timePhase(1) = 0,
22 #   fields(1, 1) = 'THETA',
23 #   filename( 1) = 'T',
24 #
25 #   frequency(2) = -3600.0,

```

```

26   timePhase(2) = 0,
27   fields(1, 2) = 'UVEL',
28   filename( 2) = 'U',
29 #
30 ...
31 &
32
33 &DIAG_STATIS_PARMS
34 &

```

Listing 3: Example `data.diagnostics` file

For a real model run we would likely want to include more diagnostics by continuing to add to the namelist `DIAGNOSTICS_LIST` in the same way. The full list of diagnostics is listed in the documentation for each package, as well as in the `available_diagnostics.log` file created when you run with the diagnostics package. See the examples in the tutorials for a full working example.

3.1.2 SEAICE package

The SEAICE package includes physical parameterizations for ice dynamics and simple thermodynamics. The dynamical model is described in more detail in section 7.6. This section focuses on how to set up and use the SEAICE package.

SEAICE has both compile-time and run-time options, like other packages.

Compile-time options

The compile-time options come from the two header files `SEAICE_OPTIONS.h` and `SEAICE_SIZE.h`. See the files in the SEAICE tutorial or in the package code `pkg/seoice/SEAICE_OPTIONS.h` and modify as needed.

Run-time options

Run-time options are specified in the file `data.seoice`. The SEAICE package has some questionable default values (see this GitHub issue). The simplest file that fixes the default values is listing 4. This file would be suitable for starting a simulation with no ice, since the initial ice area fraction, thickness, and snow files are not specified.

```

1 # SEAICE parameters
2 &SEAICE_PARM01
3 # There are some problems with SEAICE pkg defaults, see https://github.com/MITgcm
   /MITgcm/issues/107
4 SEAICEuseDYNAMICS = .TRUE.,
5 SEAICEScaleSurfStress = .TRUE.,
6 SEAICE_useMultDimSnow = .TRUE.,
7 SEAICEetaZmethod = 3,
8 SEAICE_drag = 0.001,

```

```

9   MIN_LWDOWN          = 0.0 ,
10  SEAICE_area_reg     = 1.0E-5,
11  SEAICE_hice_reg     = 1.0E-5,
12 # These three settings let ice be advected physically by the wind/surface
   currents
13  SEAICEadvHEFF       = .TRUE.,
14  SEAICEadvAREA        = .TRUE.,
15 # Always either use 33 or 77!
16  SEAICEadvScheme      = 33,
17 &
18
19 &SEAICE_PARM03
20 &

```

Listing 4: Recommended parameters for data.seaice file

Initial ice conditions can be specified as a constant value, or with an input file. To specify an initial constant value, set the keywords `SEAICE_initialHEFF`, `SEAICE_initialAREA`. To set with a file, set the input files `HeffFile`, `AreaFile`, `HsnowFile`, `HsaltFile` to initialize the initial ice thickness, fractional ice covered area, snow thickness, and salinity. Keywords should be in namelist `SEAICE_PARM01`. See the SEAICE tutorial for an example with initial ice configuration (Case 2).

3.1.3 EXF package

The external forcing (EXF) package allows the model to be forced with real (or simulated) observations of meteorological data. This package has temporal interpolation abilities, which would be convenient when working with real observations.

As with all packages, EXF requires compile-time and run-time options.

Compile-time options

The compile-time options are specified in file `EXF_OPTIONS.h`. See the file in the SEAICE tutorial, or copy from the exf package, `pkg/exf/EXF_OPTIONS.h`.

Run-time options

The run-time options are specified in file `data.exf`, see example in listing 5. Fields that should be included to force the model are in table 1

Initial state files are set in `EXF_NML_02` in `data.exf` with the keyword `[fieldname]file`. For example, `uwindfile = [filename]` to set the initial zonal winds.

The EXF package has a built in method to handle different time intervals between meteorological records and model time steps. The time interval between meteorological records in seconds is specified for each field with the namelist entry `[field]period` in `EXF_NML_02`, for any of the above fields.

Field name	Description	Typical range
uwind	Surface (10-m) zonal winds, m s^{-1}	$ \text{uwind} \leq 10$
vwind	Surface (10-m) meridional winds, m s^{-1}	$ \text{vwind} \leq 10$
atemp	Surface (2-m) air temperature, K	$200 \leq \text{atemp} \leq 300$
aqh	Surface (2-m) specific humidity, kg kg^{-1}	$0 \leq \text{aqh} \leq 0.02$
swdown	Downward shortwave radiation, W m^{-2}	$50 \leq \text{swdown} \leq 450$
lwdown	Downward longwave radiation, W m^{-2}	$50 \leq \text{lwdown} \leq 450$

Table 1: Required fields to force MITgcm

The time for the meteorological records to repeat themselves is set with the keyword `repeatPeriod` in `EXF_NML_01`.

```

1 # ****
2 # External Forcing Data
3 # ****
4 &EXF_NML_01
5 #
6 useExfCheckRange = .TRUE. ,
7 useExfCheckRange = .TRUE. ,
8 repeatPeriod      = 86400.0, # time until met. records repeat themselves
9 exf_iprec        = 64,
10 exf_yftype       = 'RL',
11 &
12 # ****
13 &EXF_NML_02
14 uwindstartdate1 = 20050401,
15 uwindstartdate2 = 00000,
16 uwindperiod     = 3600.0, # time between consecutive entries in uwindfile
17
18 vwindstartdate1 = 20050401,
19 vwindstartdate2 = 00000,
20 vwindperiod     = 3600.0
21
22 atempstartdate1 = 20050401,
23 atempstartdate2 = 000000,
24 atempperiod     = 3600.0,
25
26 aqhstartdate1   = 20050401,
27 aqhstartdate2   = 000000,
28 aqhperiod       = 3600.0,
29
30 swdownstartdate1 = 20050401,
31 swdownstartdate2 = 000000,
32 swdownperiod    = 3600.0,
33
34 lwdownstartdate1 = 20050401,
35 lwdownstartdate2 = 000000,
36 lwdownperiod    = 3600.0,
37
38 uwindfile       ='x_wind.bin',
39 vwindfile        ='y_wind.bin',
40 atempfile        ='airtemp.bin',
41 aqhfile          ='aqh.bin',
42 swdownfile       ='swdown.bin',
43 lwdownfile       ='lwdown.bin',
44 &
```

```

45 &EXF_NML_03
46 &
47
48 &EXF_NML_04
49 &
50 &
51 &EXF_NML_OBCS
52 &
53

```

Listing 5: Sample `data.exf` file. This instance has meteorological records every 3600 s (1 h), and the forcing is repeated every day.

The EXF package can also handle the case when the meteorological records start from a different time than your model run. In this case the `startDate_1` and `startDate_2` fields become important. In `data.cal`, set the model start time using these fields as follows:

- `startDate_1`: Format `YYYYMMDD` specifies the date to start
- `startDate_2`: Format `hhmmss` specifies the time to start

Then in `data.exf`, specify the start time of the meteorological records in the same way. You can set this globally, or for each individual field.

3.2 Core model parameters and recommendations

This section discusses choosing an equation of state, using checkpoint files, time-stepping, binary input/output, and other parameters for the core model.

3.2.1 Equation of state

The equation of state relates temperature, density, and salinity of water. For freshwater, the maximum density of is at 4 °C, so a linear equation of state is a poor choice with temperatures near 4 °C. The equation of state of Jackett and McDougall (1995, [2]) is a good compromise between simplicity (doesn't add an extra pickup file) and accuracy. Use this equation of state by including `eosType='JMD95Z'`, in the `PARM01` namelist of `data`. See the discussion in the documentation about EOS type.

3.2.2 Checkpoint files

The model will inevitably fail sometimes. This can lose a lot of computation time especially when running large jobs on Graham. Therefore, it is best practice to save checkpoint files periodically. MITgcm has rolling checkpoint files and permanent checkpoint files. Rolling checkpoint files save the model

current state with one of two file names ("ckptA" and "ckptB"), overwriting the oldest file to save new checkpoints. Permanent checkpoint files are saved with the model iteration number and therefore do not get overwritten. The frequency at which to save checkpoint files is controlled by the keywords `chkptFreq` for rolling checkpoints and `pchkptFreq` for permanent checkpoints in the PARM03 namelist of the `data` file.

3.2.3 Time stepping

The time step and simulation length parameters are all in the PARM03 namelist of the `data` file. Specify the start and end time with the `startTime` and `endTime` keywords. The model time step is given by the `deltaT` keyword. Alternatively, different time steps can be specified for the momentum and tracer equations by setting `deltaTmom` and `deltaTtracer` individually. If not using the diagnostics package, the file output frequency should be specified as `dumpFreq` here too. If using the diagnostics package, set this to 0.

3.2.4 File input and output

Input and output files to the model are binary files in big-endian format. I usually use double-precision floats for input, by specifying `readBinaryPrec=64`, in PARM01. This corresponds to data type ">f8" in python's numpy package. See listing 6 for a pythonic way to set input data for the model.

```

1 import numpy as np
2
3 T = np.zeros((100, 100, 50), dtype = '>f8', order = 'F')
4 for z in range(50):
5     T[:, :, z] = 4 * (z / 50)
6
7 with open('initial_temperature', "wb") as fid:
8     T.tofile(fid)
```

Listing 6: Python code for making input files to MITgcm

Model output is by default a 32-bit float data type. You shouldn't have to do much with the binary output data if you convert the data to netCDF. But sometimes it can be nice to check a field's values interactively as the model runs. Listing 7 has python code to open the output data.

```

1 import numpy as np
2 T = np.fromfile("T.0000001080.data", ">f4")
3 # ... do something with T
```

Listing 7: Python code for opening MITgcm binary output files

3.2.5 Gridding parameters

The size of the model grid must be specified at compile-time and at run-time. This includes setting the number of processors. Each compiled executable must always be run with the same number of grid points and processors, but can be run for different resolutions on the same grid.

Compile-time size options

The model grid first must be specified in file `SIZE.h`. See listing 8 for an example file. The overall grid is segmented into a few different subgrids. The full domain is split into tiles in X and Y. Then you specify the number of tiles per processor in X and Y (`nSx`, `nSy`), and the number of X, Y points per tile (`sNx`, `sNy`), and the number of processors in X and Y (`nPx`, `nPy`). Therefore, you need to satisfy $Nx = sNx * nSx * nPx$ and $Ny = sNy * nSy * nPy$. The amount each tile overlaps is given by `OLx` and `OLy`. The values you need here depend on which packages you use, and sometimes the advection schemes. The values in the example file are suitable for the SEAICE package with advection scheme 33. You can find the default file in `model/inc/SIZE.h`, and copy this into your experiment directory and modify as needed.

```
1 C $Header: /u/gcmpack/MITgcm/verification/exp0/code/SIZE.h_mpi,v 1.1 2003/09/10
2 C          02:25:46 edhill Exp $
3 C
4 C   =====
5 C   | SIZE.h Declare size of underlying computational grid.
6 C   *=====
7 C   | The design here supports a three-dimensional model grid
8 C   | with indices I,J and K. The three-dimensional domain
9 C   | is comprised of nPx*nSx blocks (or tiles) of size sNx
10 C   | along the first (left-most index) axis, nPy*nSy blocks
11 C   | of size sNy along the second axis and one block of size
12 C   | Nr along the vertical (third) axis.
13 C   | Blocks/tiles have overlap regions of size OLx and OLy
14 C   | along the dimensions that are subdivided.
15 C   *=====
16 C   \ev
17 C
18 C Voodoo numbers controlling data layout:
19 C   sNx :: Number of X points in tile.
20 C   sNy :: Number of Y points in tile.
21 C   OLx :: Tile overlap extent in X.
22 C   OLy :: Tile overlap extent in Y.
23 C   nSx :: Number of tiles per process in X.
24 C   nSy :: Number of tiles per process in Y.
25 C   nPx :: Number of processes to use in X.
26 C   nPy :: Number of processes to use in Y.
27 C   Nx :: Number of points in X for the full domain.
28 C   Ny :: Number of points in Y for the full domain.
29 C   Nr :: Number of points in vertical direction.
30   INTEGER sNy
31   INTEGER OLx
32   INTEGER OLy
33   INTEGER nSx
34   INTEGER nSy
35   INTEGER nPx
36   INTEGER nPy
```

```

37      INTEGER Nx
38      INTEGER Ny
39      INTEGER Nr
40      PARAMETER (
41      &          sNx = 40,
42      &          sNy = 40,
43      &          OLx = 3,
44      &          OLy = 3,
45      &          nSx = 1,
46      &          nSy = 1,
47      &          nPx = 10,
48      &          nPy = 10,
49      &          Nx = 400,
50      &          Ny = 400,
51      &          Nr = 50)
52
53 C      MAX OLX - Set to the maximum overlap region size of any array
54 C      MAX_OLY that will be exchanged. Controls the sizing of exch
55 C      routine bufers.
56      INTEGER MAX_Olx
57      INTEGER MAX_Oly
58      PARAMETER ( MAX_Olx = OLx,
59      &          MAX_Oly = OLy )

```

Listing 8: Sample SIZE.h configuration file

Run-time options

The resolution is specified at run time in the `data` file, namelist `PARM04`. Depending on your preference for using `r` or `z` for your vertical coordinate, this namelist might look like listing 9. The grids are specified as `(number of points) * (distance between points)`. Alternatively, you can specify a list of distances. This might be convenient for the vertical coordinate to space grid points closer to the surface.

```

1  &PARM04
2  usingCartesianGrid=.TRUE.,
3  usingSphericalPolarGrid=.FALSE.,
4  delX=400*12.50,
5  delY=400*12.50,
6  delZ=50*0.2,

```

Listing 9: Sample grid namelist data

4 What didn't work

This section describes problems with advection schemes and wind forcing that were barriers to getting the model running.

4.1 Advection schemes

MITgcm has many choices for advection schemes for various parts of the model. The SEAICE package is particularly sensitive to the choice of advection scheme. Choosing the wrong advection scheme can cause an unphysical

amount of ice to form, cause ice to form above 0 C, or fail to move ice under wind or surface current forcing. The recommended schemes are 33 and 77, with all available schemes listed in table 2 and described in the documentation.

Token	Advection Scheme
1	1st-order upwind
2	2nd-order centered difference
3	3rd-order upwind
4	3th-order centered difference
7	7th-order one step method with monotonicity preserving limiter
20	2nd-order direct space and time (Lax-Wendroff)
30	3rd-order direct space and time
33	3rd-order flux-limited direct space and time
40	Piecewise parabolic method with "null" limiter
41	Piecewise parabolic method with "mono" limiter
42	Piecewise parabolic method with "weno" limiter
50	Piecewise quartic method with "null" limiter
51	Piecewise quartic method with "mono" limiter
52	Piecewise quartic method with "weno" limiter
77	Non-linear flux limiter
80	2nd-order moment advection scheme (Prather, 1986)
81	2nd-order moment advection scheme, Prather Limiter

Table 2: Standard model parameters for case 1: Ice formation

The SEAICE advection scheme is specified by including the line (for example) `SEAICEadvScheme = 33` in the `SEAICE_PARM01` namelist of `data.seaice`.

4.2 Surface winds

At some point we tried running the model with no wind forcing. I specified a binary file with all points identically 0, and input this into the model by including `uwindfile = 'const0.bin'` and `vwindfile = 'const0.bin'` in `EXF_NML_02` of `data.exf`. In this case I started with an ice covered lake, and was trying to melt the ice with downward solar radiation and warm air temperature. There was no heat transfer between the atmosphere and ice, so the ice never melted. My theory about why involves the possible configurations you can use the EXF package in. The package needs certain combinations of fields to be specified to compute the heat fluxes and other fields that it

couples to the core model. Of course this isn't said in the documentation, but the following table is included in the header file `EXF_OPTIONS.h`. The last 4 configurations work with the SEAICE package. I typically have used configuration 3: specify air temperature, humidity, downward radiation, precipitation and runoff. The model uses bulk formulae to compute net fluxes.

	C	#	TEMP	DOWN	BULK	EVAP	TURB	actions
1	C	(1)	-	-	-	-	-	Read-in hflux, swflux and sflux.
2	C	(2)	-	def	-	-	-	Read-in hflux, swdown and sflux. Compute swflux.
3	C	(3)	def	def	def	-	-	Read-in atemp, aqh, swdown, lwdown, precip, and runoff. Compute hflux, swflux and sflux.
4	C	(4)	def	-	def	-	-	Read-in atemp, aqh, swflux, lwflux, precip, and runoff. Compute hflux and sflux.
5	C	(5)	def	def	-	def	def	Read-in hs, hl, swdown, lwdown, evap, precip and runoff. Compute hflux, swflux and sflux.
6	C	(6)	def	-	-	def	def	Read-in hs, hl, swflux, lwflux, evap, precip and runoff. Compute hflux and sflux.

Listing 10: Available configurations for running the EXF package

Having truly zero wind speed may interfere with the above configurations, and the package might not compute all the required fields to couple to the core model. A work around is specifying ε -winds. By specifying a very small wind speed $\mathcal{O}(10^{-16}) \text{ m s}^{-1}$, the model behaves properly and computes all heat fluxes.

5 Results: cases studied

5.1 Ice advection

This case was designed to verify:

- Ice will be moved by wind and surface currents
- Water will be cooled by a low air temperature
- Ice will melt in warm water and form in cold water

The setup for this experiment is specified in table 3. A square lake (6 km x 6 km x 10 m) is initialized with constant temperature 2°C and air temperature -5°C. There is 2.5 m s^{-1} wind in the positive x direction in the "South" half of the lake. Two small pieces of ice are initialized. One is directly in the

wind, and the other is out of the wind. The chunk in the wind should be advected quickly by the wind, and the other should be slower since it needs to wait for surface currents to build up and push it.

Paramter Name	Parameter Value	Comment
Nx	60	15 x 4 processors
<td>60</td> <td>15 x 4 processors</td>	60	15 x 4 processors
tempAdvScheme	33	High order advection scheme
eosType	Linear	Linear EOS with incorrent slope: caused cold water at 0 °C to drop to the bottom, below warmer water. This EOS is not recommended.
delX	60*100.0	6 km lake, 100 m resolution
delY	60*100.0	6 km lake, 100 m resolution
delZ	20*0.5	10 m deep lake, 0.5 m resolution

uwind

$$\begin{cases} 2.5 \text{ m/s} & y \leq 3\text{km} \\ 0 \text{ m/s} & y > 3\text{km} \end{cases}$$

vwind	0	
atemp	-5°C	Constant cold air temperature
aqh	0.01 kg kg ⁻¹	Specific humidity.
swdown	0	Minimum shortwave downward radiation
lwdown	50 W m ⁻²	Minimum longwave downward radiation
SEAICEuseDYNAMICS	. TRUE.	Allow SEAICE dynamics
SEAICE_area_reg	1.0E-5	Ice fraction less than this will be treated as zero ice for next time step
SEAICE_hice_reg	1.0E-5	Height cutoff
SEAICEadvHEFF	. TRUE.	Advect effective height
SEAICEadvAREA	. TRUE.	Advect seaice area fraction
Temperature	2 °C	Temperature stratified lake
Initial SEAICE area fraction	0.7	Two small chunks of ice

Initial SEAICE thickness	0.7 m	Two small chunks of ice
--------------------------	-------	-------------------------

Table 3: Parameter values for case 1: ice advection

5.1.1 Results

This case was first run with the default ice advection scheme (`SEAICEadvScheme=2`), which gives unphysical results - ice forms when the surface temperature is above 0 °C (figure 1). I ran the same case with advection schemes 7, 33, and 77 (figures 2, 3). Scheme 7 is also clearly unphysical. In this case the overlap extent in X and Y was not enough, causing the gridding pattern.

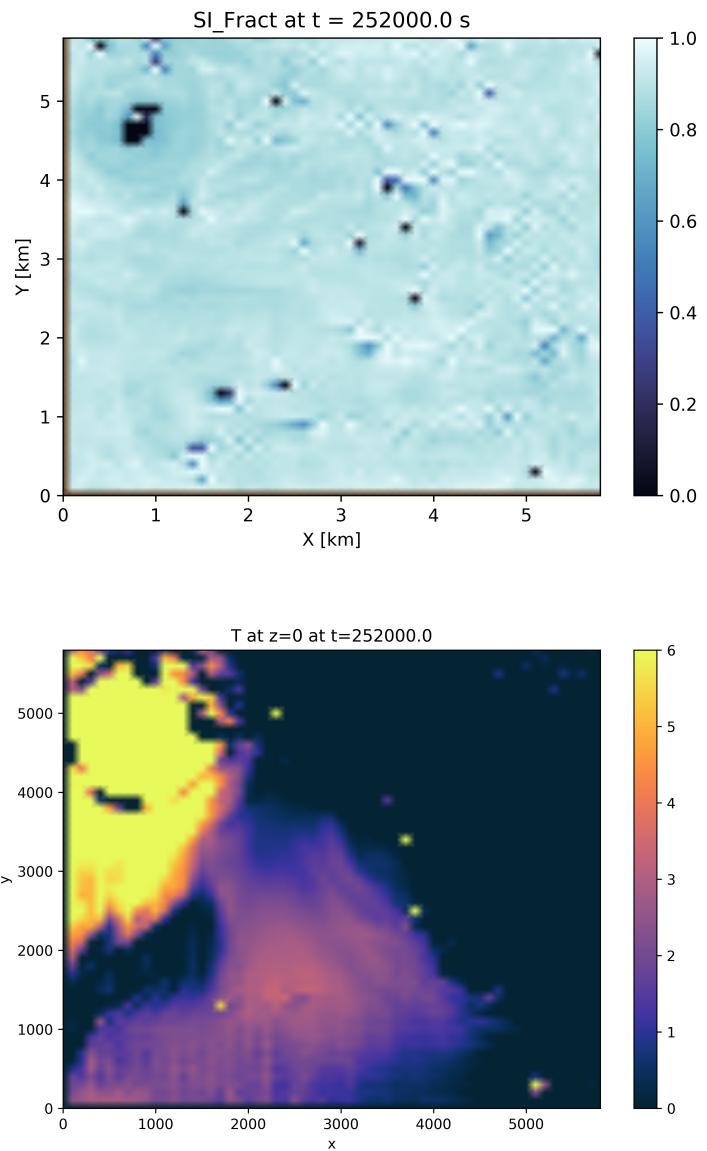


Figure 1: Ice area fraction and temperature fields for advection sceheme 2 (second order centered difference scheme). Note surface temperature much above freezing where ice has newly formed.

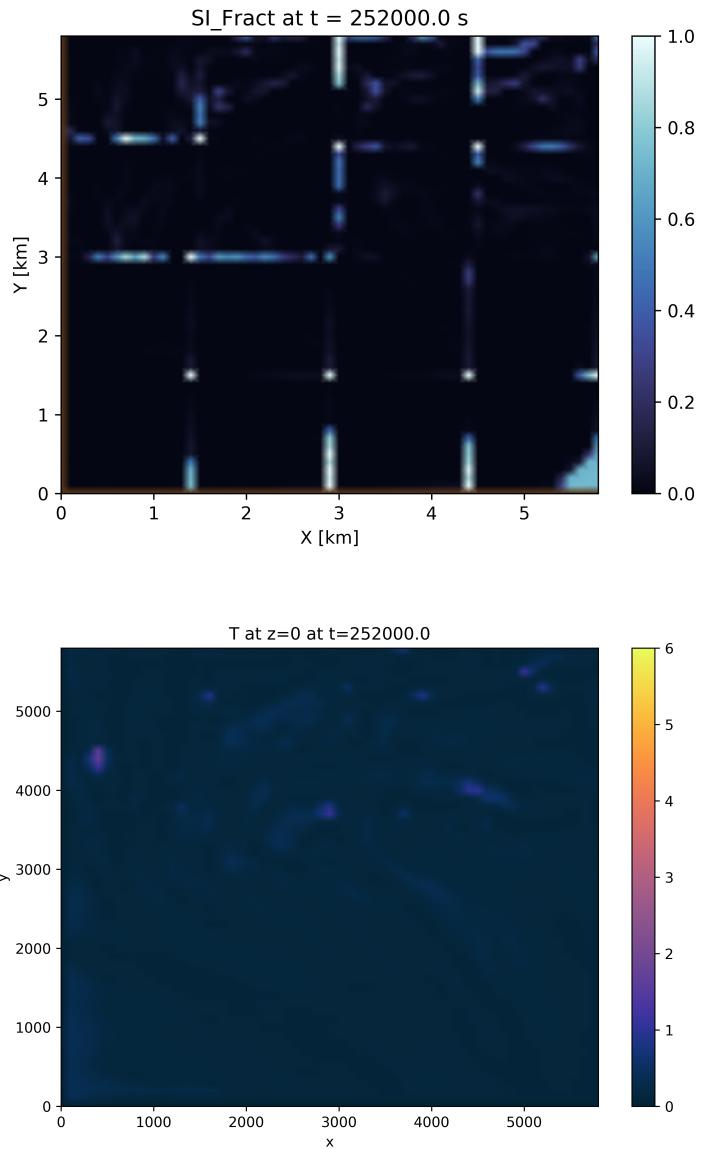


Figure 2: Ice area fraction and temperature fields for advection sceheme 7. The obvious gridding pattern comes from the wrong overlap extent in X and Y.

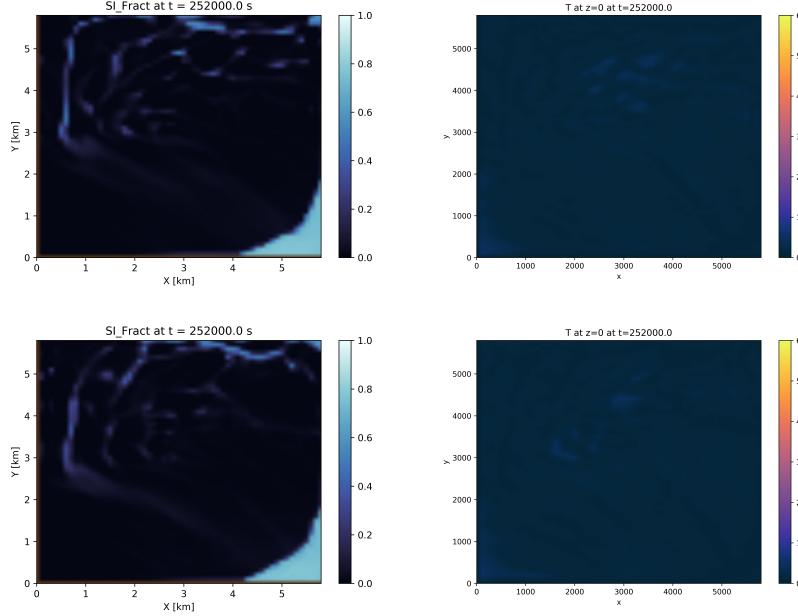


Figure 3: Ice area fraction and temperature fields for advection scheme 33 (top row) and 77 (bottom row). These agree qualitatively very well, and agree with physical intuition.

The difference between the top and bottom of each panel in figure 3 is due to the wind. Ice forms in the top half where the water is calmer (because there is no wind there), gets pushed by the surface currents, and then in the bottom half is pushed quickly by the wind. It collects in the bottom right corner.

5.1.2 What was learned

From this simulation, we learned and see the following points

- With a good advection scheme (33 or 77, high dimensional with flux limiter) the ice is advected physically by the wind and surface currents.
- The water was cooled by the cold air
- Ice melted at first because of the warm water. Ice forms later on when the surface temperature is at or below 0 (assuming we have a good advection scheme).

- The Linear equation of state is a poor choice. Looking at temperature cross-sections the cold water falls to the bottom, and water closer to 4°C rises to the top. This is opposite from what should happen. A more accurate EOS would form ice faster since cold water would stay at the top.

5.2 Ice melting

We wanted to design a scenario which would isolate how the model melts ice. The lake is initialized with constant ice cover, with warm air and moderate downward solar radiation (see complete details in table 4. There are Eastward winds in the South half of the lake. We expect the ice to melt and break up. In particular, we want to see:

- Ice to melt until it starts to break up
- The ice to start being moved around once it has space to move (once there is some open water)
- Ice to completely melt, water to be heated by the radiation and air

Paramter Name	Parameter Value	Comment
Nx	600	50 x 12 processors
Ny	600	50 x 12 processors
tempAdvScheme	33	High order advection scheme
nonHydrostatic	.FALSE.	Use hydrostatic configuration
eosType	JMD95z	Polynomial approximation to true equation of state. This EOS respects the max in density at 4°C
delX	600*10.0	6 km lake, 100 m resolution
delY	600*10.0	6 km lake, 100 m resolution
delZ	100*0.1	10 m deep lake, 0.5 m resolution
uwind		

$$\begin{cases} 1.0 \text{ m/s} & y \leq 3\text{km} \\ 0 \text{ m/s} & y > 3\text{km} \end{cases}$$

<code>vwind</code>	0	
<code>atemp</code>	10°C	Constant cold air temperature
<code>aqh</code>	0.01	Specific humidity
<code>swdown</code>	300 W m ⁻²	Minimum shortwave downward radiation
<code>lwdown</code>	300 W m ⁻²	Minimum longwave downward radiation
<code>SEAICEuseDYNAMICS</code>	.TRUE.	Allow SEAICE dynamics
<code>SEAICE_area_reg</code>	1.0E-5	Ice fraction less than this will be treated as zero ice for next time step
<code>SEAICE_hice_reg</code>	1.0E-5	Height cutoff
<code>SEAICEadvHEFF</code>	.TRUE.	Advect effective height
<code>SEAICEadvAREA</code>	.TRUE.	Advect seaice area fraction
<code>SEAICEadvScheme</code>	33	Advection scheme 33: Flux limited direct space and time advection scheme
Temperature		Temperature stratified lake
		$\begin{cases} 0^\circ\text{C} & z = 0 \text{ m} \\ 4^\circ\text{C} & z = -10 \text{ m} \end{cases}$
Initial SEAICE area fraction	0.9	Constant ice covered fraction
Initial SEAICE thickness	0.1	Constant ice thickness

Table 4: Parameter values for case 2: ice melt

The model was run with these parameters, as well as:

- Change `nonHydrostatic` to .TRUE. to test sensitivity to hydrostatic/non-hydrostatic configuration
- Double the number of grid points in X and Y to test resolution sensitivity
- Set surface winds to 0 to test effect of wind

5.2.1 Results

The first results are for the case outlined in table 4. This is hydrostatic with 12.5 m horizontal resolution and 0.5 m s^{-1} winds in the South half. See pseudocolour plots of the ice thickness and temperature in figure 5, and the ice mass time-series in figure 4. Psuedocolour plots aren't presented for other cases with wind, since they are qualitatively similar.

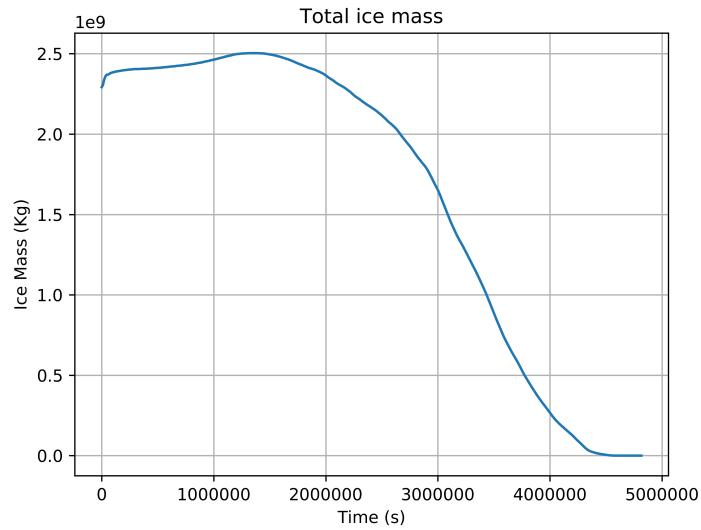


Figure 4: Total ice mass with parameters from table 4. The ice initially grows in the cold water, and later is melted by the warm air temperature and solar radiation.

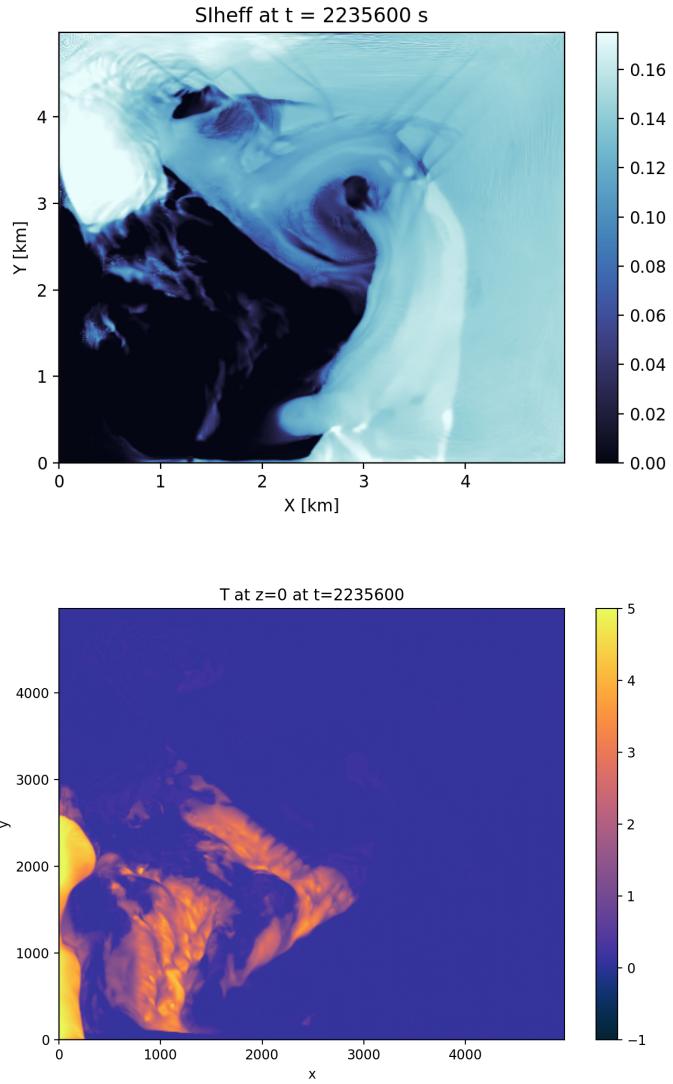


Figure 5: Effective ice thickness (top) and surface temperature (bottom) with parameters as in table 4. The ice has melted in the bottom left corner from the wind. Notice the ridge built up to the right of the open water, and the cracking in the ice above the open water.

Parameter comparisons

We want to see how sensitive the results are to changes in some parameters.
We compare

- Hydrostatic and non-hydrostatic model configurations
- Higher resolution
- Removing surface winds

The model runs much faster in hydrostatic configuration than non-hydrostatic, and eventually we will need to use the hydrostatic configuration to run a full lake-scale model. Therefore, we'd like to know how much of a difference this approximation makes for our cases. Similarly, we want to understand what impact resolution will have in our simulations. Therefore, we run 4 cases: high and low resolution for both hydrostatic and non-hydrostatic configurations. We start with 12.5 m horizontal resolution, and increase to 6.25 m resolution. Total kinetic energy is compared in figure 6. The case with no surface wind is not shown since the kinetic energy is significantly lower.

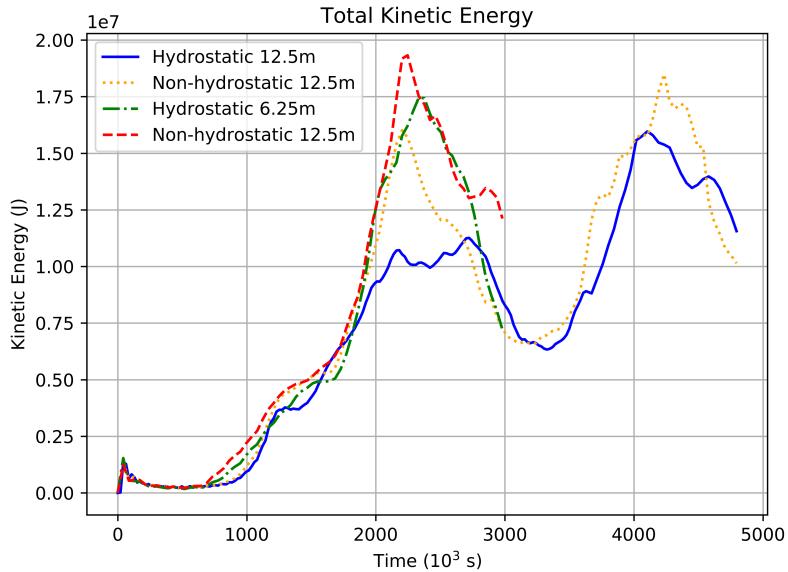


Figure 6: Comparison of total kinetic energy for different model configurations.

The high-resolution simulations have good agreement between hydrostatic and non-hydrostatic configurations, and agree well with the lower resolution non-hydrostatic configuration. The low resolution hydrostatic configuration underestimates the first maximum of kinetic energy compared to the other simulations.

We can compare the mean ice thickness and total internal energy in the water between model configurations. We compare the above simulations, as well as the simulation with no surface winds. Intuitively removing the winds should change the rate of heat transfer and ice break-up, so we investigate this. The results are in figure 7.

In each case, the ice melts most rapidly at about 5 weeks in to the simulation. Call this the break-up time. All simulations have relatively good agreement on ice thickness until the break-up time. The case with no wind takes longer to start melting, and then melts faster, than the other cases. At break-up time the case with no wind seems to be able to transfer heat from the water into the ice to melt it more efficiently than the other cases. This might explain the dip in internal energy compared to the other cases.

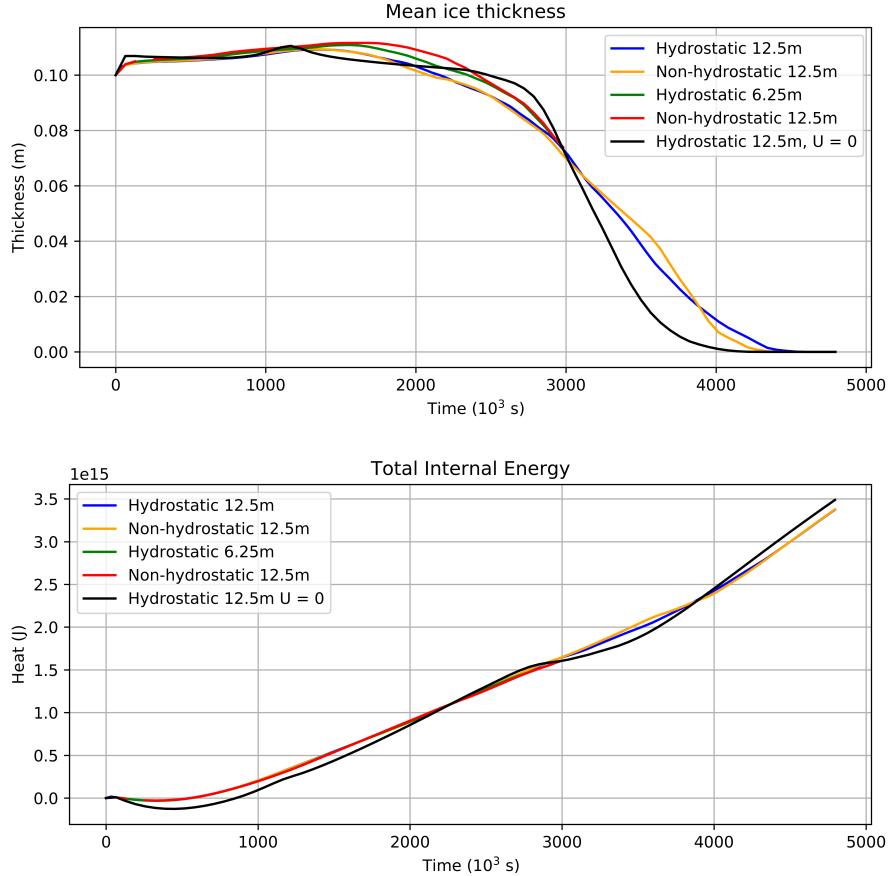


Figure 7: Comparison of mean ice thickness (top) and total internal energy of the water (bottom) between model configurations.

The case with no surface wind has very different qualitative ice distributions when it melts than the case with surface wind (fig 8). The ice mostly melts from the edges, but still forms coherent clumps. It maintains a more even distribution throughout the domain, which may contribute to melting faster.

5.2.2 What was learned

From running the main case, we can say the model melts ice reasonably physically. The ice melts until it gets thin enough to break up, at which point it cracks and starts being pushed around by the wind and surface

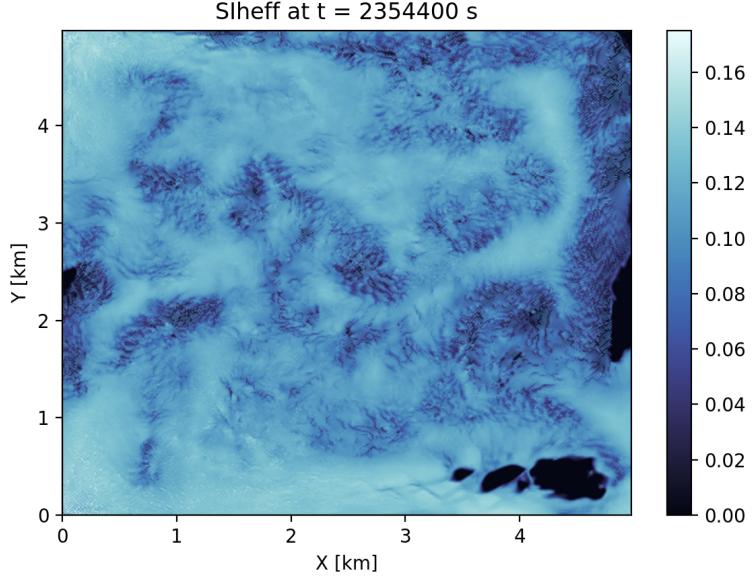


Figure 8: Surface ice thickness for case with no surface winds. The ice largely melts from the edges, with some spatial variability

currents. The ice continues to move and melt, eventually melting entirely. The remaining water is heated by the downward radiation and warm air temperature.

The total kinetic energy varies significantly between different parameter sets. In particular, the kinetic energy is significantly different between hydrostatic and non-hydrostatic configurations at lower resolutions. This could be a problem for lake scale simulations, where the resolution would be relatively large. However, we expect that for large resolutions the difference between hydrostatic and non-hydrostatic configurations should be small. Nevertheless, even with the large difference in kinetic energy, the total internal energy and average ice thickness are relatively insensitive to changing the hydrostatic flag. They are more sensitive to the inclusion of surface winds, which may be expected.

Regarding running the model, I had to include a very small wind speed instead of truly 0 winds. This is discussed in more detail in section 4.2. These cases were also run without all the recommended modifications to the default parameters given in listing 4. These results should be checked with

those improved parameters and flags.

5.2.3 Next steps

This case with the same initial conditions has been studied thoroughly. It could be worthwhile to compare the melting rate to initial conditions with some small variability in the initial ice distribution, which is more realistic. This could also be a good case to introduce more interesting bathymetry and see what effect lake shape has on the dynamics.

5.3 Solar forcing

We are interested in how the model simulates solar forcing over open water and ice covered water. This is clearly important to lake dynamics, and likely needs to be studied in more detail than I have done so far.

To study the different solar forcing for ice-covered and open water, I initialize a lake to be completely covered in 1 m thick ice in the North half, and open water in the South half. This is an idealized interface between an ice sheet and open water in a real lake. I set the air temperature to be 0 °C, and send down moderate solar radiation. I continue to use Eastward winds in only the South half of the lake. See more details on the setup in table 5.

From this case, I want to see

- More warming in the open water than under ice covered water, since the ice should reflect most of the incoming radiation and absorb heat from the water in melting
- A layer of cold water staying under the ice since the water at 0 °C is less dense than sun-heated water closer to 4 °C

Parameter Name	Parameter Value	Comment
Nx	600	50 x 12 processors
<td>600</td> <td>50 x 12 processors</td>	600	50 x 12 processors
tempAdvScheme	33	High order advection scheme
nonHydrostatic	.FALSE.	Hydrostatic model configuration
eosType	JMD95z	Polynomial approximation to true equation of state. This EOS respects the max in density at 4 °C

<code>delX</code>	600*10.0	6 km lake, 100 m resolution		
<code>delY</code>	600*10.0	6 km lake, 100 m resolution		
<code>delZ</code>	100*0.1	10 m deep lake, 0.5 m resolution		
<hr/>				
<code>uwind</code>	$\begin{cases} 0.5 \text{ m/s} & y \leq 3\text{km} \\ 0 \text{ m/s} & y > 3\text{km} \end{cases}$			
<hr/>				
<code>vwind</code>	0			
<code>atemp</code>	0°C	Constant cold air temperature		
<code>aqh</code>	0.01	Specific humidity		
<code>swdown</code>	300 W m ⁻²	Minimum shortwave downward radiation		
<code>lwdown</code>	300 W m ⁻²	Minimum longwave downward radiation		
<hr/>				
<code>SEAICEuseDYNAMICS</code>	.TRUE.	Allow SEAICE dynamics		
<code>SEAICE_area_reg</code>	1.0E-5	Ice fraction less than this will be treated as zero ice for next time step		
<hr/>				
<code>SEAICE_hice_reg</code>	1.0E-5	Height cutoff		
<code>SEAICEadvHEFF</code>	.TRUE.	Advect effective height		
<code>SEAICEadvAREA</code>	.TRUE.	Advect seaice area fraction		
<code>SEAICEadvScheme</code>	33	Advection scheme 33: Flux limited direct space and time advection scheme		
<hr/>				
Temperature	Temperature stratified lake			
<hr/>				
$\begin{cases} 0^\circ\text{C} & z = 0 \text{ m} \\ 4^\circ\text{C} & z = -10 \text{ m} \end{cases}$				
<hr/>				
Initial SEAICE area fraction	1.0, $y \geq 3 \text{ km}$	Constant ice in North half		
Initial SEAICE thickness	1.0, $y \geq 3 \text{ km}$	Constant ice in North half		
<hr/>				

Table 5: Parameter values for case 3: solar forcing

This case was run in hydrostatic and non-hydrostatic configuration.

5.3.1 Results

A plot of the surface temperature is shown in figure 9. Figure 10 shows constant temperature contours. Note the warm water pushing the 0°C contour up under the ice in the East end of the lake. The ripples in the contours are caused by velocity shear. Since the wind has a smooth transition (hyperbolic tangent function), the wind at the ice edge is half the magnitude of the maximum wind in the South half. The surface currents are then in the negative X direction against the ice and positive X in the bulk of the open water, creating shear.

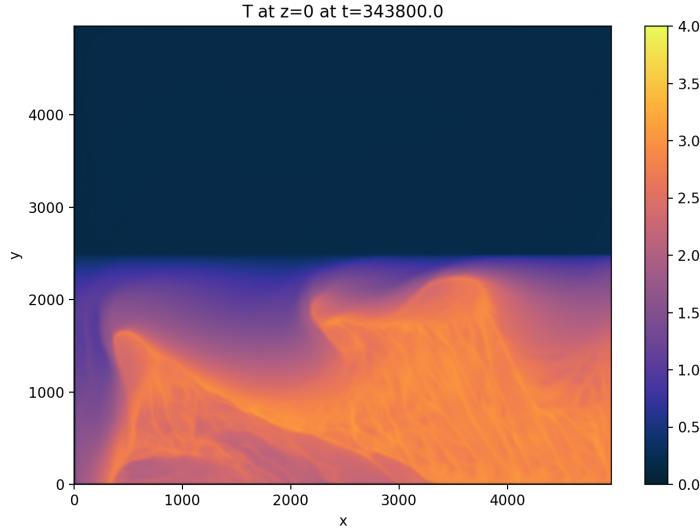


Figure 9: Surface temperature at the end of the model run. The surface layer under the ice remains at exactly 0°C , while the open water is warmed by the sun.

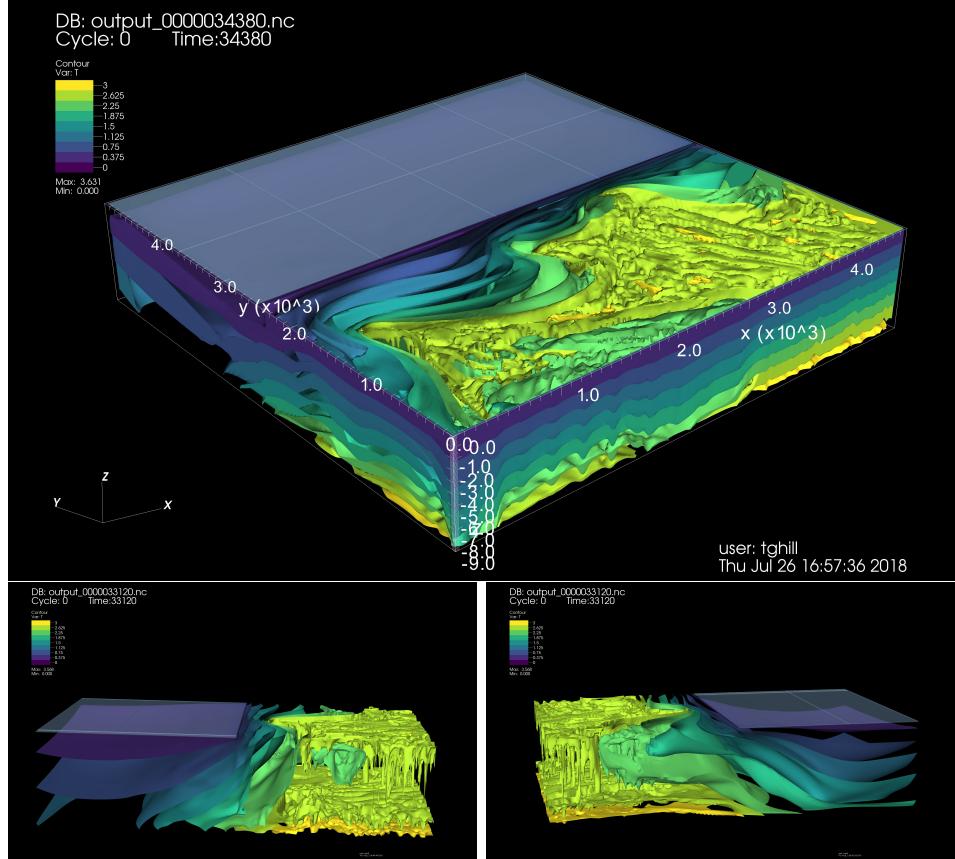


Figure 10: Constant temperature surfaces for hydrostatic configuration. The top panel shows the entire domain. The bottom panels show two perspectives of a cut of the Eastern half of the domain. The forward face in the bottom left is the center of the domain. The forward face in the bottom right is the Eastern edge of the domain.

This case was also run in non-hydrostatic configuration. A comparison of the kinetic energy, internal energy, and ice mass are shown in figure ???. The kinetic energies agree very well, and the internal energy and ice mass are even closer. This might not be the case for higher resolutions, but this was not studied. This case has a horizontal resolution of 20 m and a vertical resolution of 20 cm. If the resolution is increased to closer to 5 m as in the ice melting high resolution cases the curves might diverge.

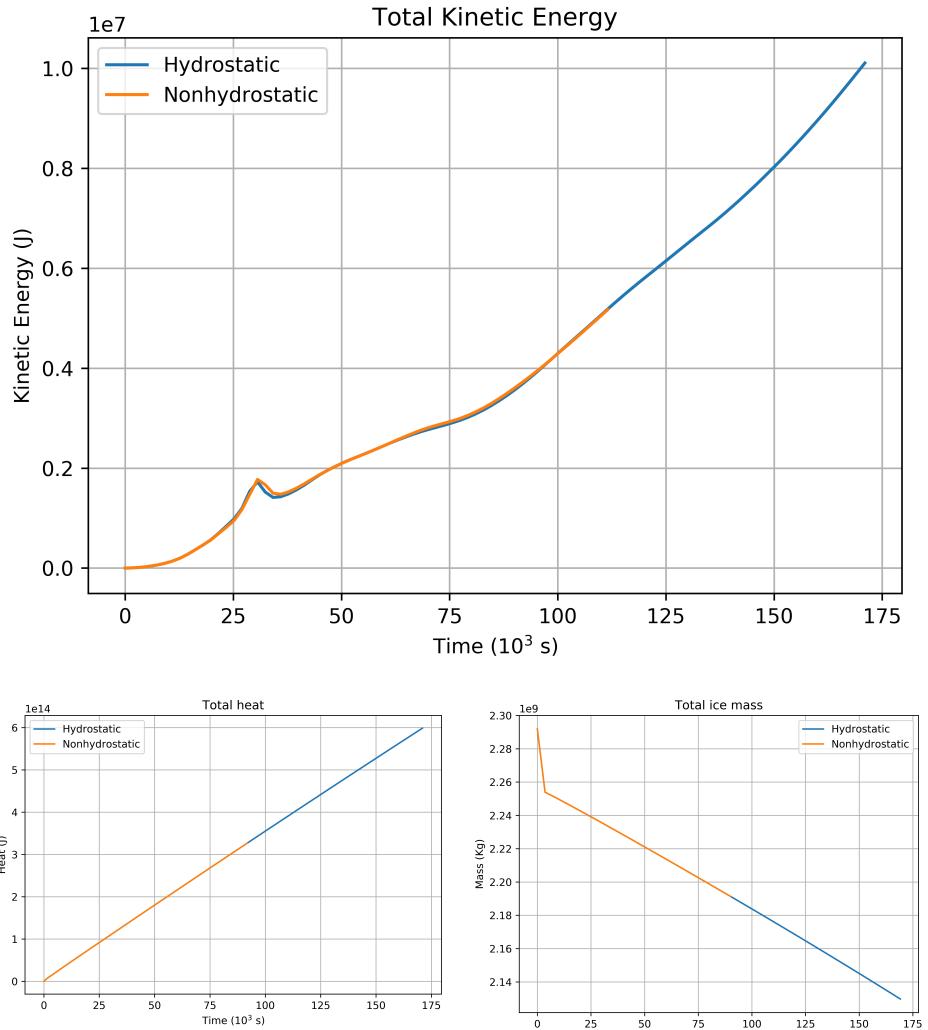


Figure 11: Comparison of total kinetic energy (top), internal energy in the water (bottom left) and ice mass (bottom right) for hydrostatic and nonhydrostatic configurations.

5.3.2 What was learned

From this case, we see that the treatment of solar radiation is consistent with our intuition. At the resolution used, the difference between hydrostatic and non-hydrostatic configurations is very small. This was also a good test case

for 3D visualization techniques (figure 10).

5.3.3 Next steps

It might be worthwhile to run this case at higher resolution on Graham to see if the non-hydrostatic configuration has more kinetic energy from convection than the hydrostatic case. It would also be interesting to try to resolve the convection currents better and really study how they interact with the ice. This might require scaling the domain down to have much finer resolution.

5.4 Periodic forcing

So far, all cases have used a similar domain, and have had constant forcing. We try running the model with a much larger domain and daily cycles in the forcing fields. We use a sinusoidal cycle as a rough approximation of the daily cycle of meteorological forcing (see figure 12).

The domain is a 25 km x 25 km square lake, modelled at 250 m resolution. The lake is 50 m deep, with 2 m vertical resolution. The sides of the lake are sloped with a constant 2% slope (drops 50 m over 2.5 km). The meteorological forcing is shown in figure 12. Some more parameter details are given in table 6.

Paramter Name	Parameter Value	Comment
Nx	100	25 x 4 processors
<td>100</td> <td>25 x 4 processors</td>	100	25 x 4 processors
tempAdvScheme	33	High order advection scheme
nonHydrostatic	.FALSE.	Non-hydrostatic model configuration
eosType	JMD95z	Polynomial approximation to true equation of state. This EOS respects the max in density at 4°C
delX	100*250.0	25 km lake, 250 m resolution
delY	100*250.0	25 km lake, 250 m resolution
delZ	25*2.0	50 m deep lake, 2 m resolution
uwind	Sinusoidal between $\pm 1 \text{ m/s}$	Daily cycle with average of 0
vwind	0	
atemp	Sinusoidal between -5 and 15 °C	Daily cycle with average of +5 °C

<code>aqh</code>	0.01	Specific humidity
<code>swdown</code>	Sinusoidal between 0 and 500 W m^{-2}	Daily cycle of SW radiation
<code>lwdown</code>	Sinusoidal between 50 and 550 W m^{-2}	Daily cycle of LW radiation, respect minimum LW radiation
<code>SEAICEuseDYNAMICS</code>	.TRUE.	Allow SEAICE dynamics
<code>SEAICE_area_reg</code>	1.0E-5	Ice fraction less than this will be treated as zero ice for next time step
<code>SEAICE_hice_reg</code>	1.0E-5	Height cutoff
<code>SEAICEadvHEFF</code>	.TRUE.	Advect effective height
<code>SEAICEadvAREA</code>	.TRUE.	Advect seaice area fraction
<code>SEAICEadvScheme</code>	33	Advection scheme 33: Flux limited direct space and time advection scheme
Temperature	$\begin{cases} 0^\circ\text{C} & z = 0 \text{ m} \\ 4^\circ\text{C} & z = -10 \text{ m} \end{cases}$	
Initial SEAICE area fraction	Large chunks, $0.2 \leq \text{SIarea} \leq 1.0$	Variable ice cover
Initial SEAICE thickness	Large chunks, $0.5 \leq \text{SIheff} \leq 1.0$	Variable ice cover

Table 6: Parameter values for case 3: solar forcing

5.4.1 Results

We expect a roughly sinusoidal signal in the output fields in response to the input forcing (fig 12). The mean value of the output fields is shown in figure 13. Notice the strange shape of the wind stress curve, and that the peak positive U value occurs before midday. We would expect the peak positive surface currents to occur at or slightly after the peak positive wind speed, but the currents appear to lead the wind.

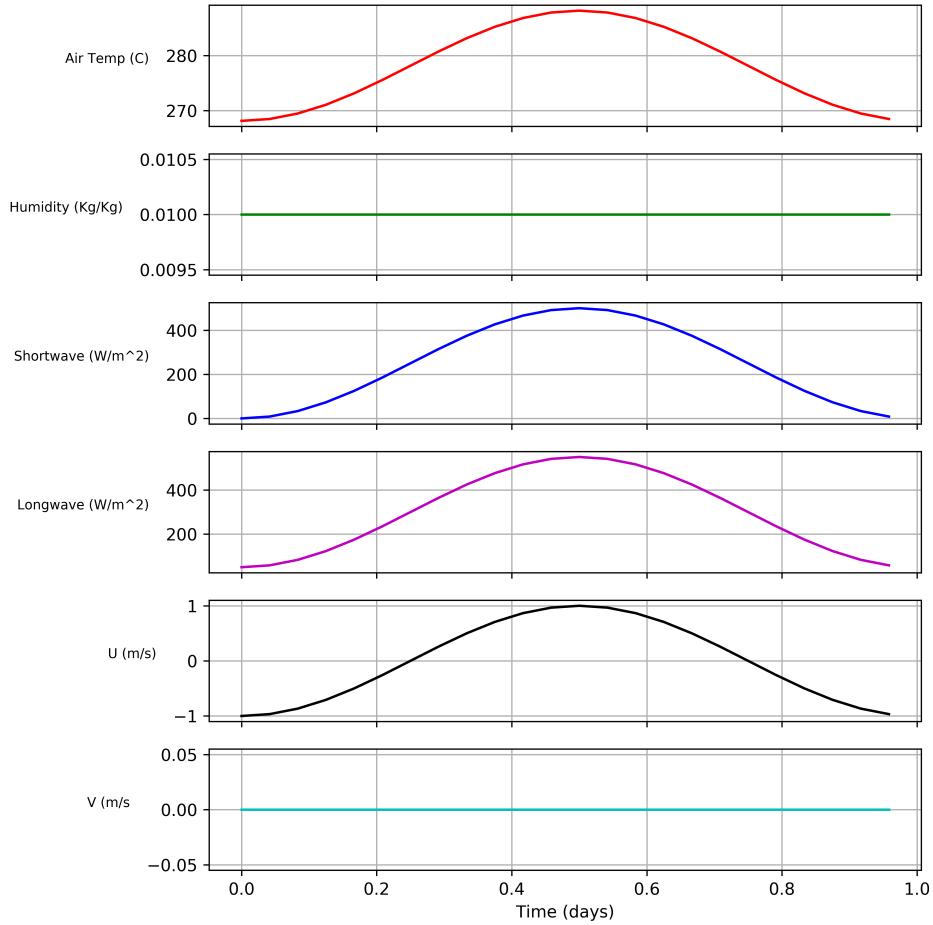


Figure 12: Time-dependent forcing values. Fields are sampled every hour (same frequency as output files are generated).

5.4.2 Run with constant temperature

We want to see what happens when we run this case with no temperature stratification. The response curves above have the strange "capped" structure, and we want to know if this goes away or stays when we take out the stratification. The capped structure could be due to the wind creating an internal seiche, and removing the stratification would remove the seiche. We set the initial water to constant 0 °C.

Running this case, the response curves have the same shape (figure 14).

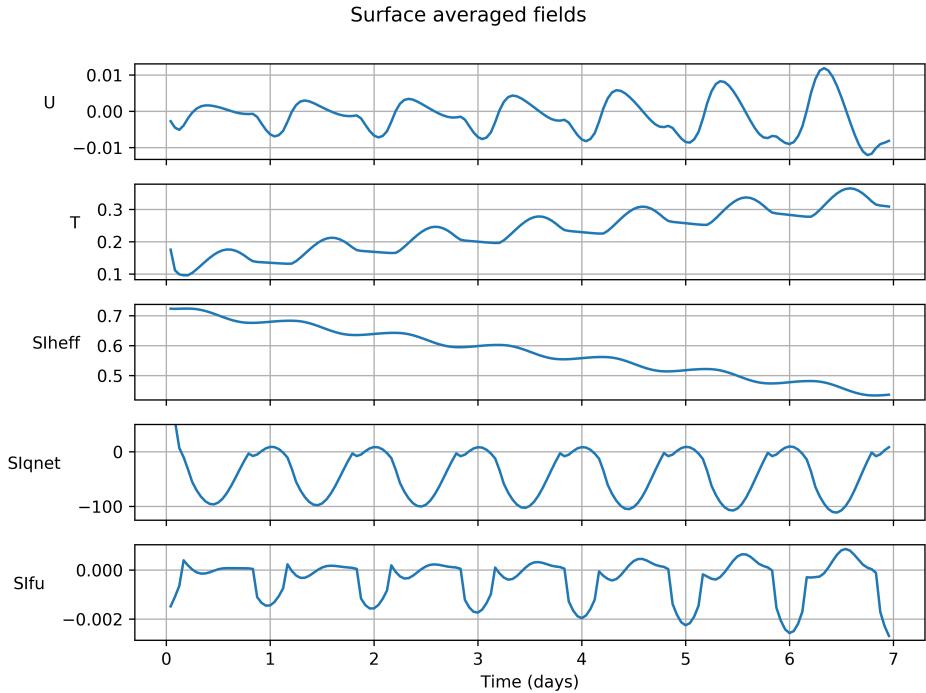


Figure 13: Mean value of output fields in response to periodic forcing. From top to bottom: mean surface U currents, mean surface temperature, mean effective ice thickness, mean heat transfer from the lake into the ice, mean surface wind stress (U direction)

This case NaN'd before the end of the model time (1 week). The other case did not NaN. I don't know why this case broke and the other didn't.

However, the total kinetic energy is significantly higher in the constant temperature case, shown in figure 15a. The difference is mostly due to the vertical velocity. When a cold parcel of water near the top of the lake is heated by the sun it warms, and gets denser. This water then starts moving downward. In the stratified case, near the pycnocline there is a restoring buoyant force which should slow or stop the vertical motion. In the non-stratified case, there is no restoring force and the parcel should continue falling and accelerating. I think this is the reason for the significant difference in Kinetic energy. However, this should be computed with available/background potential energy.

In the end, the kinetic energy difference might not be very significant.

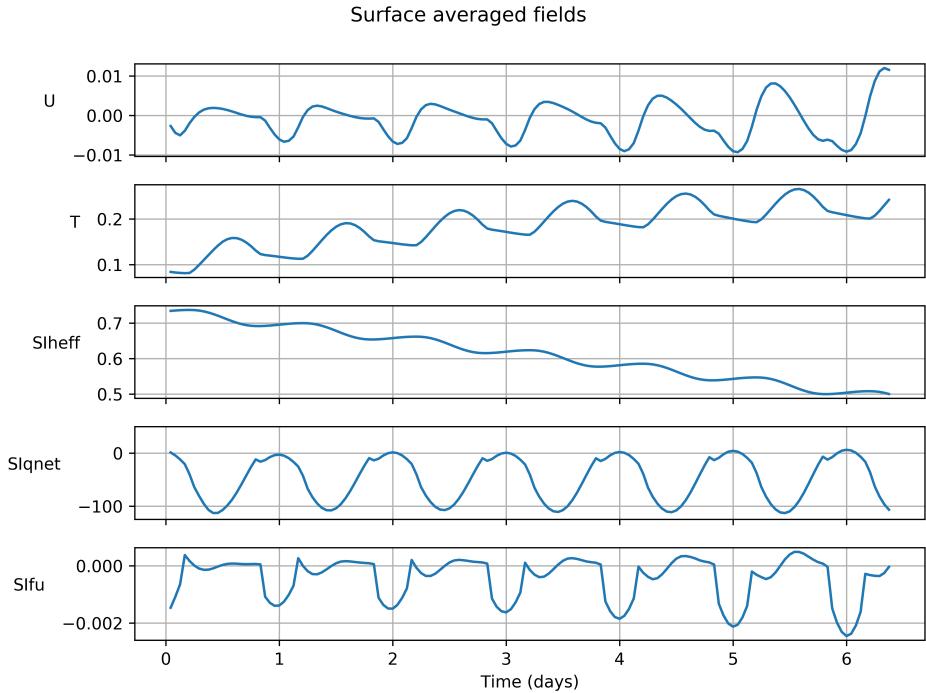


Figure 14: Mean value of output fields in response to periodic forcing, with no temperature stratification. From top to bottom: mean surface U currents, mean surface temperature, mean effective ice thickness, mean heat transfer from the lake into the ice, mean surface wind stress (U direction)

The internal energy (15b) is 7 orders of magnitude greater than the kinetic energy by the end of the simulation.

5.4.3 What we learned

From running this case, we learned how to include time-dependent forcing with the EXF package (see description of EXF package, section 3.1.3)

5.4.4 Next steps

For this case, there are some clear next steps:

- Simulate a more realistic daily cycle in the forcing, and try to bring in real meteorological observations

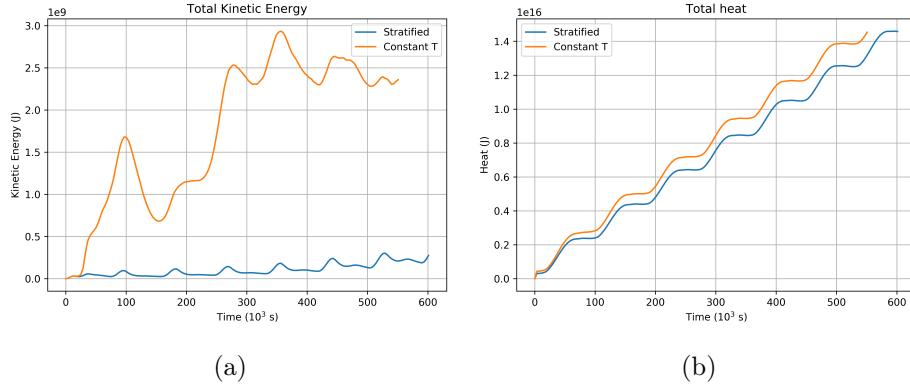


Figure 15: Comparison of total kinetic energy (top) and internal energy (bottom) between temperature stratified and non-stratified cases.

- Compute the full potential/kinetic/internal energy budget
- Diagnose why the constant temperature case NaN before the stratified case

5.5 1D comparison

Taken together, the results presented above show the model behaves well qualitatively. Now we want to validate the model quantitatively.

One way to quantitatively validate the model is to compare to a simple 1D model which has been validated against observations. A good candidate for this is the Canadian Lake Ice Model (CLIMo) ([1]). CLIMo has a simple formulation compared to MITgcm, and is very quick to run (seconds for a year of model time).

5.5.1 Comparison case

We design a very simple case to compare the two models. We run MITgcm with a small, shallow square lake (2 km x 2 km x 2 m). We exclude wind forcing, and include constant air temperature, humidity, shortwave and longwave downward radiation. We run a few cases with varying downward radiation values.

CLIMo had to be modified slightly for our purposes: The shortwave and longwave downward radiation were forced to a constant specified value that is a new input into the model instead of computing values from cloud cover and seasonal cycles. To better match MITgcm, CLIMo's ice emissivity

is set to be $\varepsilon = \frac{5.5 \times 10^{-8}}{5.67 \times 10^{-8}} \approx 0.98$, which is exactly what MITgcm uses. CLIMo's freezing temperature is set to -1.96°C , which is the value of the input `SEAICE_freeze` in MITgcm.

I changed CLIMo to make these changes, which in some sense is backwards. We should ideally be changing MITgcm to agree better with CLIMo since CLIMo is supposed to be the validated model. In practice it is much easier and faster to change parameters in CLIMo and see how it changes the agreement. Then once they agree better, we can revert CLIMo to its validated state and change the corresponding parameters in MITgcm. That being said, some of the changes to CLIMo are trying to explain the disagreement rather than correct it, including the suggestion to reduce CLIMo to have 1 ice layer.

We run both models with four radiation cases:

1. 0 W m^{-2} longwave, 0 W m^{-2} shortwave
2. 75 W m^{-2} longwave, 75 W m^{-2} shortwave
3. 125 W m^{-2} longwave, 125 W m^{-2} shortwave
4. 0 W m^{-2} longwave, 300 W m^{-2} shortwave

5.5.2 Comparison results

The mean ice thickness between the models is compared in figure 16. The models agree very well with no downward radiation, and the agreement gets systematically worse with increasing radiation. Considering the differences between the model formulations, the agreement is better than expected.

5.5.3 Next steps

The agreement from figure 16 is very good, but we would like to explain why the error systematically increases with increasing radiation. Some more modifications to try are:

- Reduce CLIMo to a 1 layer ice model. The current configuration uses 5 layers, and it may agree better with layer. There is supposed to be radiation and heat conduction between ice layers, which may explain some of the differences.
- Make the albedo values agree better. As discussed in sec 7.3, the albedo used in MITgcm is very high. The CLIMo albedo is computed as $\alpha_{CLIMo} = \max(0.44(h)^{0.28} + 0.08, \alpha_{water})$, which is lower than the

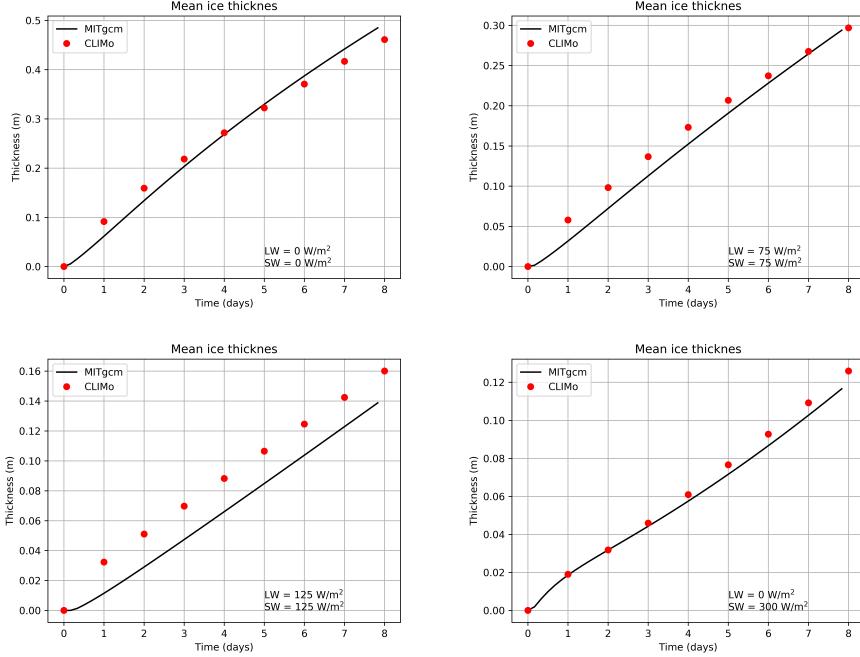


Figure 16: Mean ice thickness comparison between MITgcm and CLIMo for varying downward radiation values.

MITgcm albedo until 4.5 m, and then is greater. Even worse, it is suggested that CLIMo over-estimates snow-free ice albedo ([4]). However, if we decrease the MITgcm albedo we would expect that for increasing radiation it would grow ice slower, which would make the agreement worse.

- Turn off CLIMo’s snow capabilities entirely by setting the number of snow layers to 0
- Investigate the shortwave radiation extinction scale in CLIMo. The MITgcm parameterization is discussed in section 7.7. We know MITgcm is hard-coded to a clear ocean water type, maybe if we make the shortwave radiation extinction agree with CLIMo better it will be more applicable to lake modelling.
- For completeness, run a case with downward longwave radiation and no downward shortwave radiation

If the comparison is still interesting, some more long term things to consider:

- Compare seasonal ice growth between the models. Run CLIMo and MITgcm with seasonal forcing and compare ice on/off dates and maximum ice thickness.
- Keep CLIMo around, and periodically compare more complicated runs. This might point out areas where either model could be improved

6 Recurring issues

6.1 NetCDF/HDF5 libraries

I have periodically had problems with the NetCDF/HDF5 libraries. These problems usually came from running out of disk space or disk quotas, but sometimes show up when the wrong modules are loaded. When processing the model binary output files to netCDF files, sometimes the process fails with a generic netCDF or HDF library error. This may be caused by one of the two issues below.

6.1.1 Disk space or quota

The first time this issue came up, I had run out of my disk quota on Graham. Try running the bash command `ls -l`. You want to see something like `drwxr-s--- 3 tghill ctb-mmstastn 4096 Jul 24 14:08 include` for each line. Of the two names, the second should be `ctb-mmstastn`, because then your file storage will count towards the contributed resource space, instead of your own (or elsewhere). If a file shows up with group ownership that is your username (or another supervisor who has smaller storage space), you can change the ownership by running `chown [username]:ctb-mmstastn` (or a different group name).

If you're running MITgcm a lot on Hood, ask Robyn for access to a larger storage drive to store your data on. The home disk is relatively small.

6.1.2 Modules

The loaded modules on Graham can also present problems. I haven't been able to determine which modules don't work together when running or compiling the model, but to install and use NetCDF tools, the NetCDF and HDF modules should be loaded. You can search for the current versions with the command `module spider netcdf`, `module spider hdf5`. If you're compiling and running in parallel, you might need to load the mpi versions. The list

of modules below has worked for me for compiling, running, and converting output to netCDF.

- StdEnv/2016.4
- nixpkgs/16.09
- imkl/11.3.4.258
- intel/2016.4
- openmpi/2.1.1
- hdf5/1.8.18
- netcdf-mpi/4.1.3

6.2 MITgcm python utils

A python `FutureWarning` comes up when running the netCDF conversion package from `gcmpy` with python 3. This might become an Exception in python 3.7. This is caused by a problem in the `MITgcmutils` package that ships with `MITgcm`.

The problem line was fixed in commit 36b33d0d, see this pull request. Any model code before this change should be updated to the most recent version with this issue fixed.

7 Physics of the model

The model uses physical parameterizations that are usually documented to some extent in the `MITgcm` documentation. As part of getting the model up and running, I looked into some of the parameterizations and compared to what we expect.

7.1 Radiative cooling

We looked into the radiative cooling of ice when starting to compare `MITgcm` ice growth rates to the 1-D model CLIMo (Canadian Lake Ice Model).

The radiative cooling should follow the Stefan-Boltzmann law for total radiative power M

$$M = \varepsilon\sigma T^4 \tag{1}$$

Where M is the radiant emittance (W m^{-2}), ε is the emissivity, and $\sigma = 5.670\,373 \times 10^{-8} \text{ W m}^{-2} \text{ K}^{-4}$ is the Stefan-Boltzmann constant.

The peak wavelength of the spectral distribution should follow Wien's law

$$\lambda_{max} = \frac{b}{T} \quad (2)$$

Where $b = 2.898 \times 10^{-3} \text{ m K}$ is Wien's displacement constant and T is the temperature. For 0°C , $\lambda_{max} \approx 10^{-5} \text{ m}$ is deep in the infrared (longwave). Therefore, all the radiant emittance should be captured as upward longwave radiation.

The emittance will depend on the value of emissivity used. The SEAICE packages takes in ice emissivity with the variable `SEAICE_emissivity` which is actually $\sigma\varepsilon$. The default value is $\sigma\varepsilon = 5.50 \times 10^{-8} \text{ W m}^{-2} \text{ K}^{-4}$. Computing the expected emittance,

$$M = 306.17 \text{ W m}^{-2}$$

The model shows very good agreement with this value (fig. ??)

7.2 Water surface energy budget

As one way to compare the effect of parameters and meteorological forcing values, we compute the total internal energy in heat of the water, as well as the heat fluxes across the water boundary. We expect that

$$\frac{dQ}{dt} = \Phi_{net} = \Phi_{ice} + \Phi_{atm} \quad (3)$$

The model outputs $-\Phi_{net}$ as the diagnostic `SIqnet` (heat flux from the water into ice if ice covered, atmosphere if open water). To check this, we compute the internal energy of the water as

$$Q = c_w \sum_{x,y,z} \rho \Delta V \Delta T \quad (4)$$

Where the sum is over all grid cells, ΔV is the volume of each cell, and ΔT is the temperature at time t minus the initial temperature.

The derivative is taken using the most basic scheme,

$$\frac{dQ}{dt} \approx \frac{Q(t_{n+1}) - Q(t_n)}{t_{n+1} - t_n}$$

Finally, a plot of $\frac{dQ}{dt} + \text{SIqnet}$ should be near zero (not identically zero because of the derivative scheme) for all times. We have found this to be the case.

7.3 Surface albedo

In comparing the MITgcm ice growth rate to CLIMo, I investigated the albedo parameterization of MITgcm. The model classifies ice according to the temperature of the surface water T_{surf} compared to the freezing/melting temperature T_{melt}

$$\text{ice} = \begin{cases} \text{dry}, & T_{surf} < T_{melt} \\ \text{wet}, & T_{surf} \geq T_{melt} \end{cases}$$

The default ice albedo is

$$\alpha_i = \begin{cases} 0.75, & T_{surf} < T_{melt} \\ 0.66, & T_{surf} \geq T_{melt} \end{cases}$$

The default snow albedo is

$$\alpha_s = \begin{cases} 0.84, & T_{surf} < T_{melt} \\ 0.70, & T_{surf} \geq T_{melt} \end{cases}$$

And the values are controlled by the input variables `SEAICE_dryIceAlb`, `SEAICE_wetIceAlb`, `SEAICE_drySnowAlb`, and `SEAICE_wetSnowAlb`.

7.4 Thermodynamics

The SEAICE package is mainly intended for ice dynamics, but includes basic thermodynamics. The THSICE package is be fully compatible with SEAICE and include improved thermodynamics. However, the THSICE package is not well suited for freshwater simulations, so for the purpose of this document it is recommended to continue to use only the SEAICE package for dynamics and thermodynamics.

The THSICE package is intended to improve the thermodynamics of saltwater ice. When ice forms over salt water, some pockets of salt water called brine pockets get trapped in the ice. These brine pockets provide interesting thermodynamics. When the ice is warmed, some of the ice around the brine pockets will be melted. This meltwater must be freshwater, so it dilutes the salt in the brine pocket. This can sufficiently dilute the brine

pocket so that the pocket freezes completely, releasing latent heat to the surrounding ice [5].

Clearly this process is irrelevant for freshwater lake ice. While the three-layer formulation of [5] is an improvement over what is described in the SEAICE documentation, the implementation does not work for freshwater. The model tends to NaN once ice starts to form. Moreover, the SEAICE documentation says “NOTE: THIS SECTION IS TERRIBLY OUT OF DATE” about the thermodynamics description in the documentation. Therefore, we can’t really compare the packages anyways.

On a more practical note, the THSICE package breaks the model with any good advection schemes. It runs (with unphysical results, no ice motion) advection scheme 2. With 33 or 77 (77 is highly recommended for this package) it breaks the model very quickly, often after the first time step. We have enough justification to exclude the package from our simulations.

7.5 Total kinetic energy

I compute the total kinetic energy of the water as an additional way to compare the differences between different parameters and configurations. The total kinetic energy is computed as

$$\text{TKE} = \frac{1}{2} \sum_{x,y,z} \rho (u^2 + v^2 + w^2) \Delta V \quad (5)$$

Where $\rho = \rho_0 + \rho'$ is the total (reference + anomaly) density of the grid cell, u, v, w are the x, y, z velocities, and ΔV is the volume of the grid cell.

7.6 Dynamical ice model

The equation governing the dynamical evolution of the ice is the conservation of momentum equation

$$m \frac{D\mathbf{u}}{Dt} = -mf\mathbf{k} \times \mathbf{u} - m\nabla\phi(0) + \mathbf{F} + \tau_{\text{air}} + \tau_{\text{ocean}} \quad (6)$$

Where $m = m_i + m_s$ is the total mass of ice and snow per unit area, $\phi(0)$ is the sea surface height potential, $\mathbf{F} = \nabla \cdot \sigma$ is the divergence of the ice stress tensor, and $\tau_{\text{air}}, \tau_{\text{ocean}}$ are the wind-ice and ocean-ice stresses. The wind and ocean stresses are parameterized as

$$\begin{aligned} \tau_{\text{air}} &= \rho_{\text{air}} C_{\text{air}} \|\mathbf{U}_{\text{air}} - \mathbf{u}\| R_{\text{air}} (\mathbf{U}_{\text{air}} - \mathbf{u}) \\ \tau_{\text{water}} &= \rho_{\text{water}} C_{\text{water}} \|\mathbf{U}_{\text{water}} - \mathbf{u}\| R_{\text{water}} (\mathbf{U}_{\text{water}} - \mathbf{u}) \end{aligned} \quad (7)$$

Where \mathbf{U}_i are the surface winds of the atmosphere and water, C_i are air and water drag coefficients, ρ_i are reference densities, and R_i are rotation matrices.

7.7 Radiation

The model can be forced with net or downward shortwave and longwave radiation. Longwave radiation is absorbed by the surface layer, while shortwave radiation penetrates into the interior of the water. This section describes the penetrating shortwave radiation. The equations here come from [3] and the source code, particularly `pkg/exf/exf_radiation.F`, `model/src/swfrac.F` and `model/src/external_forcing.F`.

First, consider the amount of shortwave radiation that is transmitted through the surface. The reflected shortwave radiation is the product of the surface albedo and the incoming shortwave radiation, αI_{down} . Then, the amount absorbed into the water is

$$I_0 = (1 - \alpha)I_{down} \quad (8)$$

The radiation transmission model [3] splits the radiation into two wavelength bands. Each decays exponentially

$$\frac{I}{I_0} = Re^{-z/\zeta_1} + (1 - R)e^{-z/\zeta_2} \quad (9)$$

Where I is the radiation at depth z (positive from the surface), and $R = 0.62$, $\zeta_1 = 0.60\text{ m}$, $\zeta_2 = 20\text{ m}$ are constants that come from Jerlov water type IA. This water type represents clear ocean water, and is hard-coded into MITgcm.

This treatment should work well for relatively deep water, but for shallow lakes this is a problem. For example, with a 2 m deep lake approximately 37% of the radiation makes it to the bottom of the lake. This radiation should then be absorbed by the bottom and eventually heat the lake from the bottom, but this is not accounted for.

7.8 Humidity

The EXF package uses specific humidity (input as `aqhfile`), which is the ratio of the mass of water vapour to the total mass of a unit volume of air. This is different than relative humidity, which we are generally more familiar with.

Symbol	Name	Value
N	Avogadro's number	$6.022 \times 10^{23} \text{ mol}^{-1}$
R	Ideal gas constant	$8.3145 \text{ J mol}^{-1}$
M_{H2O}	Molar mass of water	$18.015\ 34 \text{ g mol}^{-1}$
M_{dry}	Molar mass of dry air	$28.9644 \text{ g mol}^{-1}$

Table 7: Constants for relative humidity calculation

The comparison between relative humidity and specific humidity depends on temperature and atmospheric pressure. The conversion is outlined below. Constants are given in table 7

The most relevant direction is to compute specific humidity from relative humidity. We would usually choose a relative humidity value, and want to find the corresponding specific humidity to input to the model. The specific humidity is the mass ratio of water vapor to dry air. That is,

$$q = \frac{X_{H2O} M_{H2O}}{X_{H2O} M_{H2O} + (1 - X_{H2O}) M_{dry}} \quad (10)$$

Where X_{H2O} is the molar mixing ratio (equivalent to pressure ratio),

$$X_{H2O} = \frac{P_{H2O}}{P} \quad (11)$$

P_{H2O} is the partial pressure of water vapour, and P is the atmospheric pressure (in Pa). The partial pressure of water is given by

$$P_{H2O} = \frac{R \cdot e(T)}{100} \quad (12)$$

Where R is the relative humidity (%), T is the temperature ($^{\circ}\text{C}$) and $e(T)$ is the vapour pressure of water (Pa),

$$e = 100 [a_0 + T (a_1 + T (a_2 + T (a_3 + T (a_4 + T (a_5 + T a_6)))))]) \quad (13)$$

The constants are given for water and ice in table 8, and $e = \min(e_{water}, e_{ice})$.

All together with $e(T)$ given by 13,

$$q = \frac{1}{1 + \frac{P \cdot 100}{R \cdot e(T)} \cdot \frac{M_{dry}}{M_{H2O}}} \quad (14)$$

Going the other way, we see that with $q = 0.01$, $T = -10^{\circ}\text{C}$ and $P = 101325 \text{ Pa}$, the relative humidity is above 500%. Clearly this is a problem!

NA	Water	Ice
a_0	6.107 799 961	6.109 177 956
a_1	$4.436\,518\,521 \times 10^{-1}$	$5.034\,698\,970 \times 10^{-1}$
a_2	$1.428\,945\,805 \times 10^{-2}$	$1.886\,013\,408 \times 10^{-2}$
a_3	$2.650\,648\,471 \times 10^{-4}$	$4.176\,223\,716 \times 10^{-4}$
a_4	$3.031\,240\,396 \times 10^{-6}$	$5.824\,720\,280 \times 10^{-6}$
a_5	$2.034\,080\,748 \times 10^{-8}$	$4.838\,803\,174 \times 10^{-8}$
a_6	$6.136\,820\,929 \times 10^{-11}$	$1.838\,826\,904 \times 10^{-10}$

Table 8: Vapour pressure constants for equation 13

The model was run in the configuration from fig 16 with 125 W m^{-2} downward SW and LW radiation with the original humidity value ($q = 0.01$, $R \approx 580\%$) and a more realistic value ($q = 8.0 \times 10^{-4}$, $R = 50\%$). The output is exactly identical. Changing the humidity values has no effect on any of the output fields. This is suspicious - humidity should have some effect. This is probably worthwhile to investigate: when does humidity matter? It should matter to evaporation at least. Maybe this case just has no evaporation.

8 Visualization tools

Visualization tools for animations of top-down views and cross-sections were developed in python. These are good to get an idea of what is happening in a case, and to compare cases. Code and documentation for these post-processing tools live on GitHub [here](#).

Some basic scripts for 3D visualization in VisIt were also developed. These are more involved to use and develop, but produce much more interesting graphics. These will be added to the wiki as examples of VisIt scripts.

9 Conclusions

The cases presented in Results are good toy cases to validate the qualitative dynamics of the MITgcm ice model in freshwater lakes. The model clearly freezes water into ice and melts ice at physical temperatures (with the right advection scheme). The ice dynamics once it forms are intuitively correct. The ice is advected by wind forcing and surface currents, and the ice exerts

surface stress on the water in response ($-\tau_{water}$ is applied to the surface layer of water).

The various packages necessary for modelling freshwater lake ice have been used and documented here, and some of the physical parameterizations are examined here. It is determined that the THSICE package that is recommended to be used by the MITgcm documentation is not applicable to freshwater ice. The interesting thermodynamics it is meant to recreate do not exist in freshwater ice, and some detail of the implementation of the package does not work with freshwater.

The comparison to CLIMo is very promising, but has some more work to be done before the model is well validated.

10 Next steps

There are some clear directions to extend the cases from Results in that are discussed within the individual sections. There are some overall directions to go from here:

- Make the comparison to the 1D model more complete (sec 5.5)
- Include the recommendations from the SEAICE section 3.1.2 into the toy cases
- Learn what the thermodynamic formulation of the SEAICE package is. It might be an interesting project to write a better thermodynamics package for freshwater ice. It could be based on the layer structure of the THSICE package, but with salinity eliminated from the equations

References

- [1] Claude R. Duguay, Greg M. Flato, Martin O. Jeffries, Patrick Ménard, Kim Morris, and Wayne R. Rouse. Ice-cover variability on shallow lakes at high latitudes: Model simulations and observations. *Hydrological Processes*, 17(17):3465–3483, 2003.
- [2] David R. Jackett and Trevor J. McDougall. Minimal Adjustment of Hydrographic Profiles to Achieve Static Stability. *Journal of Atmospheric and Oceanic Technology*, 12(2):381–389, 1995.
- [3] Clayton A. Paulson and James J. Simpson. Irradiance Measurements in the Upper Ocean. *Journal of Physical Oceanography*, 7(6):952–956, 1977.

- [4] N. A. Svacina, C. R. Duguay, and L. C. Brown. Modelled and satellite-derived surface albedo of lake ice - Part I: Evaluation of the albedo parameterization scheme of the Canadian Lake Ice Model. *Hydrological Processes*, 28(16):4550–4561, 2014.
- [5] Michael Winton. A reformulated three-layer sea ice model. *Journal of Atmospheric and Oceanic Technology*, 17(4):525–531, 2000.