**Heat Equation Solver in Sea Ice**

**Overview**

This repository is designed to hold all of the codes for solving one-dimensional the heat equation numerically in a block of sea ice floating on sea water. Eventually, this code will be coupled with the Large Eddy Simulation (LES) code that is being used in the Environmental Fluid Mechanics laboratory at Princeton University.

Collaborators:

* Joseph J. Fogarty
* Elie Bou-Zeid

**Physical System**

We now describe our physical system and problem statement. We consider a block of sea ice floating in the Arctic ocean that is 2 meters deep, with the top of the ice exposed to the arctic atmosphere, the bottom of the ice below the water. If we consider the temperature to be constant in the horizontal, then we are able to solve the 1D heat equation to obtain a vertical profile for the temperature of the ice.

There are a number of fluxes that occur on the bottom and top of this sheet of ice. The figure below summarizes all of the fluxes.

### Top Fluxes

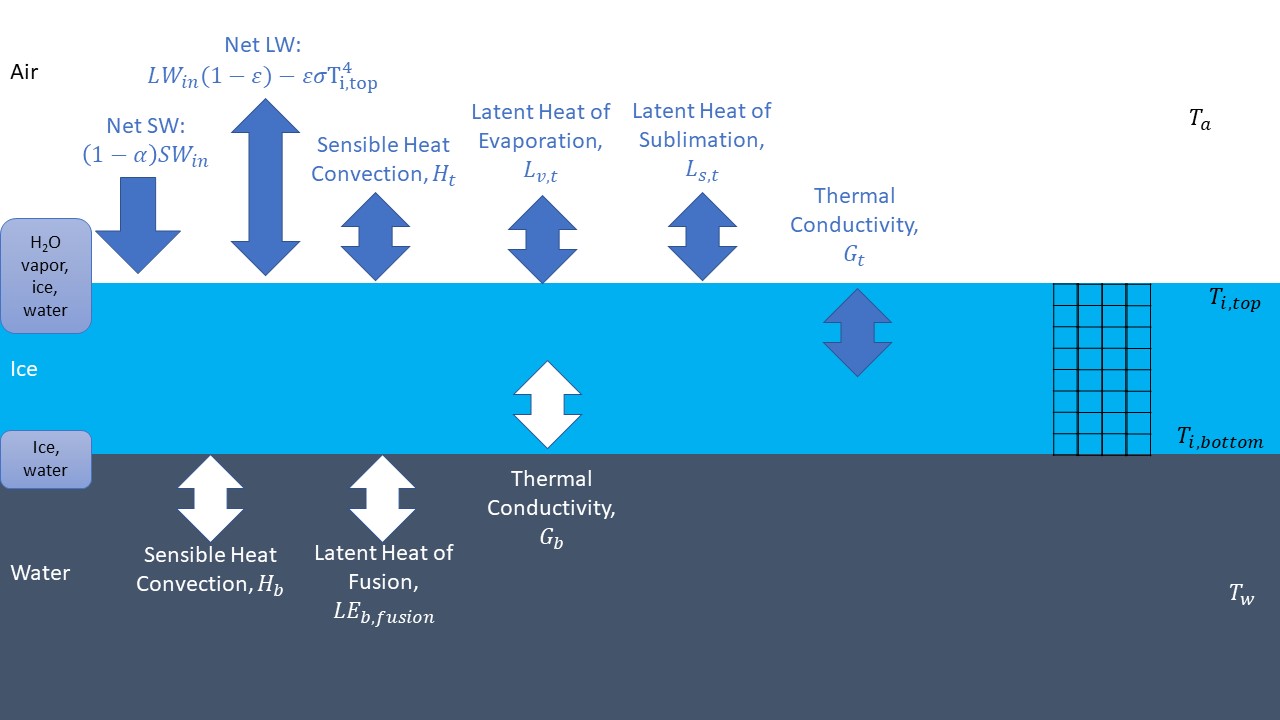
Net Radiation: Composed of net shortwave and longwave fluxes. Net shortwave flux is simply the solar radiation in that is not reflected by the surface. Net longwave in is any longwave in not reflected by the flux. Net longwave out simply follows the Stefan-Boltzmann law.

Sensible Heat Convection:

Latent Heat of Evaporation:

Latent Heat of Sublimation:

Thermal Conductivity: This flux follows Fourier’s law of heat transfer from inside of the ice to the top of the ice. In this model, it uses the first interior gridpoint to calculate the flux to the boundary condition.



**Numerics of the Code**

As mentioned above, the domain for this ice block is 2 meters. Using a spatial mesh of 400 cells, we have a length of 5 mm between nodes. We set a dt of 0.5 seconds, however this number may be changed in the code.

The main numerical method utilized here is the Crank-Nicolson method, an implicit method that is second-order accurate in space and time. The Crank-Nicolson method gives rise to solving a sparse linear system every time step. The Python module scipy.sparse allows for quick iterations of solving these sparse systems.

**Models in the Code**

There are a few models in the code based off of real-world observations and theory. First, we define a function to calculate the specific humidity above ice given a certain temperature:

# Calculate q from T

def ice\_q(T):

e = 611\*np.exp((Ls/R\_v)\*((1/273)-(1/T)))

return (eps\_R\*e)/(p\_a+e\*(eps\_R-1))

This function is based on the model… (cite thermo book)

There is also a function to calculate the total shortwave flux via solar radiation.

Function here

The sunrise and sunset values (0700 and 2200, respectively) were estimated from picking some arbitrary day in early April and looking at the sunrise and sunset times (via timeanddate.com):

A function for air temperature is also set:

Function here

The constants in these functions used may be found in the appendix and the first lines of code in `main`.

# Animation Function

There is another file in this code called `animator.py`, which takes the output of solutions from the main code and creates an MPEG file of the time evolution of the temperature profile. It utilizes the `matplotlib` libraries `matplotlib.animation` as well as `matplotlib.pyplot`.