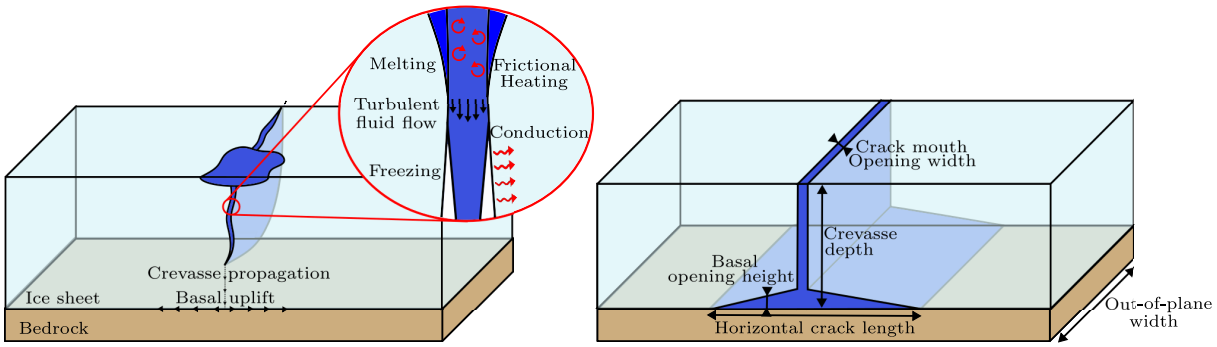

Figure 1: Overview of the system simulated using the presented MATLAB code

IceHydroFrac: A MATLAB code to simulate water-filled crevasse propagation and uplifting of ice sheets

Tim Hageman[a,*], Jessica Mejía[b], Ravindra Duddu[c], Emilio Martínez-Pañeda[a]

[a] *Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, UK*
[b] *Department of Geology, University at Buffalo, Buffalo, NY 14260, USA*
[c] *Department of Civil and Environmental Engineering, Department of Earth and Environmental Sciences, Vanderbilt University, Nashville, TN 37235, USA*

Abstract

Documentation that accompanies the *MATLAB* code IceHydroFrac, available from here. This documentation explains the usage of the implemented finite element framework, and highlight the main files.

If using this module, please cite: T Hageman, JZ Mejia,R Duddu, and E Martinez-Pañeda. *Ice Viscosity Governs Hydraulic Fracture Causing Rapid Drainage of Supraglacial Lakes.* The cryosphere [1].

Keywords: MATLAB, Hydraulic fracture, Greenland Ice Sheet, Crevasse, Numerical simulation, Viscous effects

*Corresponding author
Email address: tim.hageman@eng.ox.ac.uk (Tim Hageman)

Contents

1. Introduction

Intro sentences

1.1. Basic usage

All parameters are set within and relevant functions are called from the matlab file "main.m", and running this file performs the full simulation. Parameters are also set within this file, for instance defining the visco-plastic model through:

main.m

```
64  %Interior of ice and rock: Momentum balance
65  physics_in{1}.type = "ViscoElastic";
66  physics_in{1}.Egroup = "Internal";
67  physics_in{1}.young = [9e9; 20e9];      %Youngs modulus of Ice and Rock [Pa]
68  physics_in{1}.poisson = [0.33; 0.25];   %Poisson ratio of ice and rock [-]
69  physics_in{1}.A0 = 5e-24;               %Glenns law creep coefficient [Pa^-3 s^-1]
70  physics_in{1}.Q = 150e3;                %Energy for scaling creep with temperature
        [J]
71  physics_in{1}.TRef = 273.15;            %Reference temperature at which A0 is
        determined [K]
72  physics_in{1}.n = 3;                    %Glenns law creep exponent [-]
73  physics_in{1}.T_Ice = T_Ice;            %Temperature profile for the ice
```

These parameters are automatically passed along to the relevant physics models once they are initialized. As such, no changes in other files are needed to adapt the simulation set-up for other parameters.

2. Summary of included files

The code is set up in a object-oriented manner, defining matlab classes for each sub-component and providing their accompanying methods. As a result, a clear distinction is made between different components, and each can be used and altered with limited/no impact on other components. Here, the different classes are described. The commenting style employed within the code is compatible with the matlab help function, as such information about all usable methods within a class can be accessed by including the relevant folders, and typing, for instance, "help Solver" to print all variables contained within and all function available from the solver.

2.1. main.m

This is the main file, from which all classes are constructed and the actual simulation is performed. Within it, all properties used within other classes are defined as inputs. These properties are then passed on to initialize the "physics" object, via

main.m

```
147 %initialize physics
148 physics = Physics(mesh, physics_in, dt0);
```

taking all relevant input parameters within the physics_in structure, and an initial time step increment dt0. In a similar manner, the mesh is also initialized from a structure of properties,

```
142 %load mesh from file
143 mesh = Mesh(mesh_in);
144 mesh.plot(true, true, false, false);
145 mesh.check();
```

which reads the mesh from a file, initializes the required elements and node groups, displays the mesh in a figure, and confirms the mesh is valid and prints statistics related to the area of each separate element group to the output. Finally, the main file performs the time-stepping, calling the function:

```
195         %solve current time increment
196         solver.Solve();
```

to solve the actual time increments.

After each time increment, outputs are saved into a single structure for later plotting,

```
198         %save timedata
199         physics.models{8}.updateSurfaceElevation(physics);
200         TimeSeries.tvec(end+1) = physics.time;%times of outputs[s]
201         TimeSeries.Lfrac(end+1) = mesh.Area(9);%crevasse depth/length
202         TimeSeries.Qvec(end+1,:) = physics.models{5}.QMeltTot;%thermal fluxes ('Ice
                desorbtion', 'Flow produced', 'Melting process')
203         TimeSeries.Qinflow(end+1) = physics.models{6}.QTotal(end);%total volume of fluid
                that has entered the crevasse
204         TimeSeries.qCurrent(end+1) = physics.models{6}.qCurrent(end);%current inflow rate
205         TimeSeries.upLift(end+1,1) = physics.models{6}.dxCurrent(end);%surface uplift at
                the centre of the surface
206         TimeSeries.upLift(end,2) = physics.models{6}.dyCurrent(end);%surface uplift at the
                centre of the surface
207         TimeSeries.SurfaceDisp(end+1,:,1) = physics.models{6}.surface_dX;%full uplift
                profile at top surface    (horizontal displacement)
208         TimeSeries.SurfaceDisp(end,:,2) = physics.models{6}.surface_dY;%full uplift profile
                 at top surface (vertical displacement)
```

These outputs are:

200. tvec: This contains all time increments at which the other elements within this structure have outputted data. Notably, as the time increment varies between the initialization period ($tvec(i) < 0$) and actual simulations ($tvec(i) \geq 0$) differs, this vector is also required to translate the number of the time step (as used within the naming of full output files) to the actual time of the outputs.
201. Lfrac: This is the length all fractured interfaces. When the crevasse has yet to reach the base, this corresponds to the depth of the crevasse. After reaching the bottom, this is the depth of the crevasse (=the ice thickness) plus the total length of the horizontal cracks (both directions summed together).
202. Qvec: This reports the total thermal energies produced/consumed throughout the simulation. The first element of this vector corresponds to the thermal energy conducted into the ice, the second to the heat produced by the turbulent flow due to friction, and the final element corresponds to the thermal energy used to cause freezing/melting of the crevasse walls. These values are the integrated totals for the complete crack, and are also integrated over the complete time.
203. Qinflow: The total volume of fluid that has entered the crevasse from the inlet at the surface. As with all other outputs, this is given per metre of unit depth.
204. qCurrent: The current fluid inflow at the top inlet, $\partial \text{Qinflow}/\partial t$.
205. upLift: Surface displacement at the centre of the top surface, coinciding with the crevasse. This contains the horizontal displacement (half the crevasse opening height), and the vertical displacement.
207. SurfaceDisp: Vectors containing horizontal and vertical displacements for the complete top surface of the ice-sheet. The coordinates that correspond to the given data points are saved within TimeSeries.SurfaceCoords.

In addition to these time series, full outputs are saved after every 10 time steps,

```
215         %save outputs for post-processing and restarting
216         if mod(tstep, 10) == 0
217             filename = savefolder+string(tstep);
218             save(filename, "mesh","physics","solver","t_max","TimeSeries");
219         end
```

These output files are appended with the number of the time increment during which the output is saved, and they contain all information required to restart a previously interrupted simulation.

References

[1] T. Hageman, J. Z. Mejia, R. Duddu, E. Martinez-Pañeda, Ice viscosity governs hydraulic fracture causing rapid drainage of supraglacial lakes [Submitted], The Cryosphere.