CryoGrid Community Model

S. Westermann et al.

Dec 04, 2020

CONTENTS:

CHAPTER

ONE

QUICK START

1.1 Software requirements

CryoGrid is written in Matlab. Version 2018x or higher is required.

1.2 Known Limitations

- Multi-tile (3D) functionality not yet implemented.
- The writing of this documentation is in progress and not yet complete. There is a pdf file available with more in depth explanation of the model (CryoGrid_documentation.pdf). The information from this document will be gradually included in this documentation.

1.3 Brief model description

CryoGrid is a simulation tool to calculate ground temperatures and volumetric water/ice contents (as well as salt concentration, etc. depending on the selected `SUBSURFACE` classes) in single-tile (1D) stratigraphies. Multi-tile (3D) models can be realized by coupling several 1D stratigraphies.

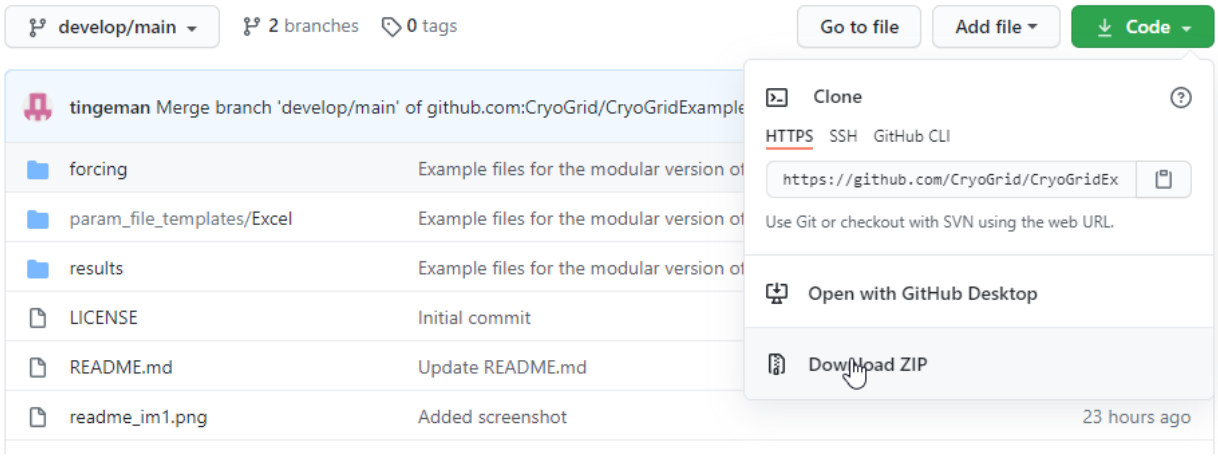
A stratigraphy is realized by stacking one or several `SUBSURFACE` classes, which each account for different physical processes. The different `SUBSURFACE` classes typically have specific state variables and model parameters and use different constitutive equations to calculate those variables.

1.4 Get started - getting code and examples for running your first model

Here we describe how to set up the code to run an initial test model as a first time user. We have a separate section on obtaining the necessary files through git: *see below*

1.4.1 Get the test examples

First download the test example from the github repository CryoGrid/CryoGridExamples. You may download the repository contents as a zip-file, by clicking the green `code` button and choose `Download Zip`. See screenshot below. (In the upper left hand corner, make sure the `develop/main` branch is selected).

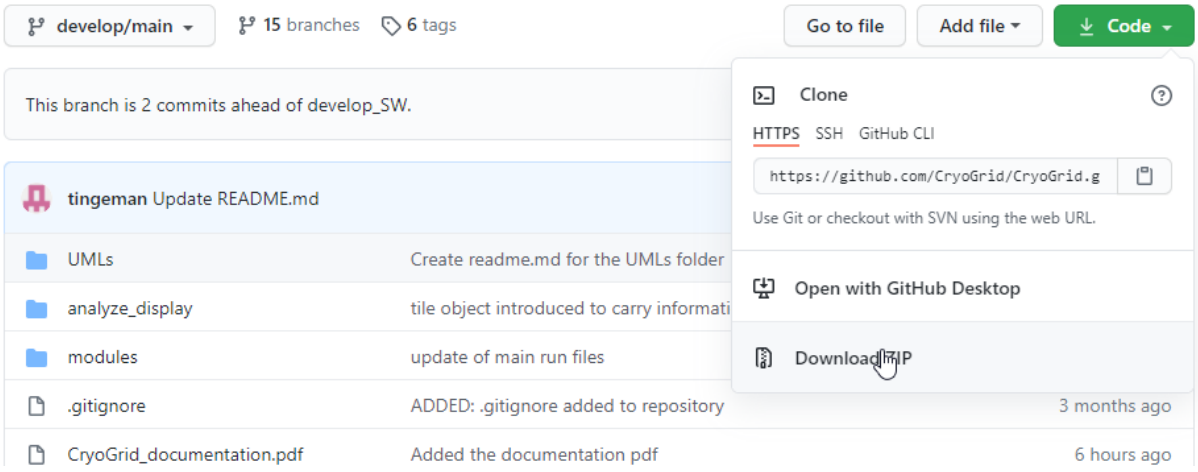


Unzip the the contents to your preferred folder, f.ex. `c:\my_matlab_code\cryogrid\`. You will now have the following folder structure: `c:\my_matlab_code\cryogrid\CryoGridExamples-develop-main`, which will contain example run files and model definitions. Rename this folder:

```
rename c:\my_matlab_code\cryogrid\CryoGridExamples-develop-main c:\my_matlab_code\
↪cryogrid\CryoGridExamples
```

1.4.2 Get the main CryoGrid code

Download the main CryoGrid code as zip-file. From the main GitHub repository page, click the green `code` button and choose `Download Zip`. See screenshot below. (In the upper left hand corner, make sure the `develop/main` branch is selected).



Now unzip the contents of the zip file to the folder `c:\my_matlab_code\cryogrid\CryoGridExamples`. You will now have the following folder structure: `c:\my_matlab_code\cryogrid\CryoGridExamples\CryoGrid-develop-main` Rename this folder:

```
rename c:\my_matlab_code\cryogrid\CryoGridExamples\CryoGrid-develop-main c:\my_matlab_
↪code\cryogrid\CryoGridExamples\CryoGrid
```

1.4.3 Run the test model

Open MatLab and change the path to `c:\my_matlab_code\cryogrid\CryoGridExamples\`. This can be done either using the MatLab path selector dialog, or by typing in the command window:

```
cd c:\my_matlab_code\cryogrid\CryoGridExamples\
```

Change the paths according to the actual install folder you chose for the example code.

To run the example model, in the MatLab terminal run the file run_CG.m file by typing:

```
run_CG
```

The code will start running. It will produce one ouput file per year, which is written to disc at a specific date (defined in the parameter Excel file `c:\my_matlab_code\cryogrid\CryoGridExamples\results\test\test.xlsx`). You can stop the code any time after a full year has been calculated (to ensure you have an output file written to disc).

The first output file written to disc by the example model will have the name `c:\my_matlab_code\cryogrid\CryoGridExamples\results\test\test_19800901.mat`.

1.4.4 View the output of the model

To plot the results, change the path to `c:\my_matlab_code\cryogrid\CryoGridExamples\CryoGrid\analyze_display\`:

```
cd c:\my_matlab_code\cryogrid\CryoGridExamples\CryoGrid\analyze_display\
load(`c:\my_matlab_code\cryogrid\CryoGridExamples\results\test\test_19810901.mat`)
read_display_out()
```

Plots will be generated for several parameters. Not all of them are meaningful for all model configurations. Find and inspect the figure showing the temperature field.

1.4.5 To change the model parameters

The model is defined in the file `c:\my_matlab_code\cryogrid\CryoGridExamples\results\test\test.xlsx`. You may play around with the model parameters and se how the output changes.

For example, you could change the thickness of layer 1:

- Open the excel file
- Find the section `STRAT_layers`
- First column in the defined matrix lists the depth to the bottom of each layer (so row 1 has the depth to the bottom of layer 1)
- The first layer is by default from 0 - 0.5 m (0.5 m thick).
- To change the thickness of the first layer to 1 m, simply change the value from 0.5 to 1.

Rerun the model to see the changes. (Be aware that the out files are overwritten, back them up if you want to store for comparison.)

1.4.6 Getting the code and examples using the git commandline tool

1. Clone the CryoGridExamples repository to a new directory (fx `c:\my_matlab_code\cryogrid`):

```
git clone --single-branch --branch develop/main https://github.com/CryoGrid/
↪CryoGridExamples.git
```

1. Navigate into the new directory `c:\my_matlab_code\cryogrid\CryoGridExamples`

```
cd c:\my_matlab_code\cryogrid\CryoGridExamples
```

1. Clone the main CryoGrid model code

```
git clone --single-branch --branch develop/main https://github.com/CryoGrid/CryoGrid.
↪git
```

Continue with running the model as described *above*

CHAPTER

TWO

CRYOGRID CODE STRUCTURE

2.1 Folder structure

The CryoGrid model code is contained in the folder "modules", in which it is organized as follows:

- `modules/TIER0`: base level: contains the basic class definitions for CryoGrid SUBSURFACE and INTERACTION classes. TIER0 does not contain functional CryoGrid classes.
- `modules/TIER1`: library level: inherits from TIER0 base classes, contains classes comprising all functions related to a certain physical process. TIER1 does not contain functional CryoGrid classes.
- `modules/TIER2`: first SUBSURFACE class level: inherits from TIER1 library classes, contains fully functional CryoGrid classes
- `modules/TIER3`: second SUBSURFACE class level: inherits from TIER2 SUBSURFACE classes, contains fully functional CryoGrid classes. TIER3 in particular contains the SUBSURFACE classes (GROUND, LAKE, etc. classes) that can interact with a (SUBSURFACE) SNOW class.
- `modulesTIERXX/INTERACTION`: INTERACTION (IA) classes defining interactions and fluxes between pairs of SUBSURFACE classes, same TIER structure as for SUBSURFACE classes

and so on ....

CHAPTER

THREE

ADDING NEW MODULES AND FUNCTIONALITY TO CRYOGRID

3.1 Style guide

- Most important **(THIS IS A MUST!): Use clear and understandable variable names and not abbreviations**
- Physical properties should be named by their SI symbols (if it exists and makes sense).
- In functions, used variables from containers should be saved by their name, i.e. `theta = ground.CONST.theta`
- Variables and functions should be in lower case. For variable names consisting of several words, camel case or underscores can be used.
- If you use equations and constants, cite their source in comments.
- The CryoGrid code contains many cases in which the style guide was not followed, partly due to implementation of legacy code, partly due to negligence. To ensure readability of the code and error messages, rule No.1 is by far the most important and must be followed!!

3.2 Class documentation

Every author should indicate her/his name and the date in the header of each class (same for files that are not a class). If major changes or updates are done, the author(s) of the changes should again include name(s) and date, but not remove theprevious entries in the header. In the code, comments should be inserted to make it understandable. In addition, each CryoGrid class should be described in this manual. Every author is responsible for the documentation of new CryoGrid classes.

3.3 Not so Quick Start

How to create a new `SUBSURFACE` class

There is no definite scheme for compiling new SUBSURFACE classes, so this is only a rough guide. Discuss your plans with an experienced developer prior to setting off! Most important: Design and create new classes without changing anything (!) in existing functions or classes! Even if only a minor change in e.g. a function in a TIER1 class would be necessary, add a new function to the TIER1 class instead of changing the existing function. This ensures that existing functionality is not affected by the new class. If changes to existing code become necessary, this must be discussed with all developers.

1. Always develop and test a new `SUBSURFACE` class myClasswithout taking the snow cover into account. Normally (there are exceptions), the variable snowfall provided by the FORCING class should not be used at all. This means that development should start at the `TIER2` level.

2. Choose the TIER2 class *oldClass* which is closest in functionality to the new class and copy + rename this class to newClass.
3. Change the class name in the initialize function. Add additional parameters (PARA), state variables (STATVAR) and constants (CONST) that must be initialized form the parameter and constant files in the appropriate private methods of newClass(i.e. provide_PARA, provide_STATVAR and provide_CONST). The variables in these functions must be declared empty ([]) and will be automatically populated during the initialization process. Dependent variables (i.e. variables computed from the variables populated during initialization) should not be declared here, but the appropriate code computing dependent variables should be added to the finalize_init() function.
4. Add statements to the mandatory functions in the new SUBSURFACE class. In general, additional functionality should be coded as functions in TIER1 classes, and then called as a single line statement in the TIER2 class. If adequate, make a new TIER1 class and add it to the list of classes after the class_defstatement.
5. Test the new class without interactions with other classes, i.e. by making a parameter file in which the stratigraphy consists only of newClass (i.e. no other classes).
6. Compatibility with other classes: check the interaction classes used for oldClassand, if changes arenecessary, copy, rename and modify them in the same way as for the SUBSURFACE class, paying attention to the TIER1 and TIER2 levels. Add *newClass* and potential new INTERACTION classes to the function get_IA_class()in TIER2/INTERACTION. Carefully update the compatibility matrix in get_IA_class.m. Make sure you don't change anything related to existing SUBSURFACE class combinations!
7. Test all combinations with other classes in dedicated test examples.
8. Compatibility with snow cover: To add snow cover representation to newClass, copy one of the TIER3 classes (oldClass_snow.m) and rename it newClass_snow.m. Search and replace all occurrences of "oldClass" by "newClass", as well as oldClass_snow with newClass_snowin thclass_def statement and the constructor. In general, all TIER3 classes will work as template, the only rule is not to mix Xice classes and non-Xiceclasses (if myClass features Xice, use an oldClass which has Xice). Redo the previous two steps fornewClass_snow, but also ensure compatibility with the SNOW classes through dedicated INTERACTION classes.
9. Lateral interactions: **IMPORTANT!** All new code related to lateral fluxes needs to be added to TIER2 *newClass.m* and related TIER1 classes, not in the LATERAL classes. Also here, check the dedicated functions(in generalpush_... and pull_...)in oldClass and modify them. To ensure compatibility with a lateral class in LATERAL/LAT1D or LAT3D, check which SUBSURFACE class functions are called. In general, this will be in the functionspush and pull. Add these functions to TIER2 newClass.m, placing the actual code in a suitable TIER1 class.
10. Comment the new code and add a description to "Section 5: Detailed Documentation" in this document

CHAPTER

FOUR

CATEGORIES OF CRYOGRID CLASSES

CHAPTER

FIVE

DETAILED DOCUMENTATION

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search