HAIL-CAESAR Documentation

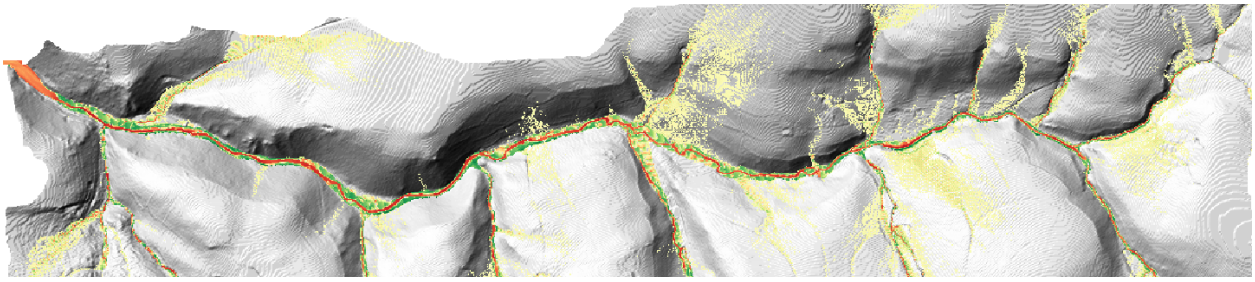
Release 1.0

Declan Valters

Mar 31, 2021

Contents

Welcome to the documentation pages for the HAIL-CAESAR catchment modelling software. These pages will help you set up and run the model, and configure it to do hydrological, flood-inudation and sediment erosion modelling. Have fun!

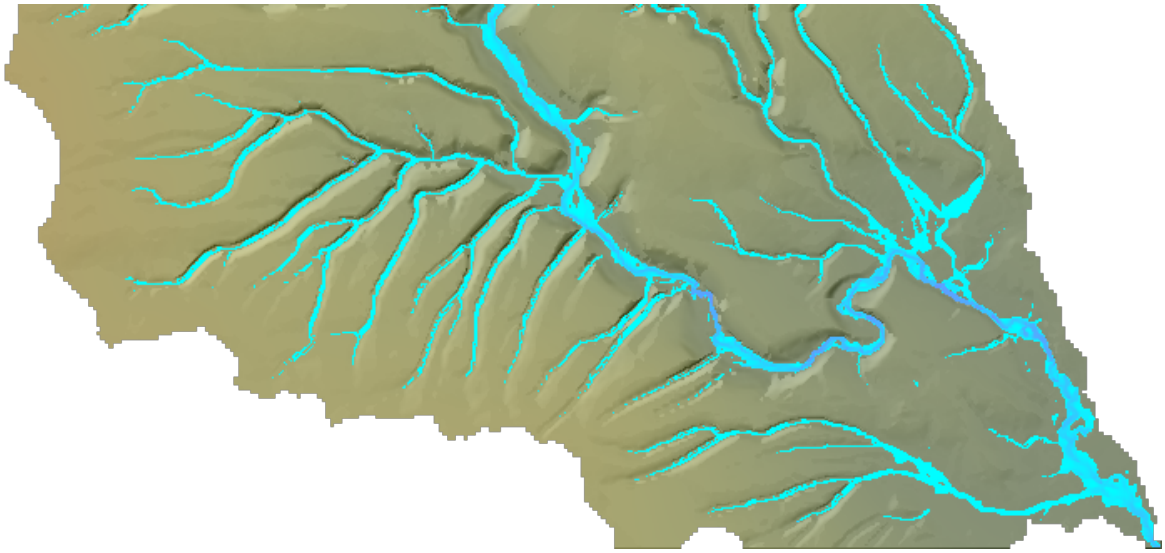


CHAPTER 1

Contents

1.1 Introduction

1.1.1 Hydrological and Erosion Modelling



Numerical models are useful tools for testing hypotheses of geomorphic processes and landscape evolution. Long term landscape evolution takes place over extensive geological time periods, but is controlled by the dynamics of water flow, and sediment transport within river catchments on much shorter, day-to-day time scales. The HAIL-CAESAR package provides a model for simulating catchment processes at short-term scales.

This chapter documents how to use the HAIL-CAESAR model to set up and run simulations of hydrology and catchment-scale erosion and evolution. HAIL-CAESAR is a 2.5D numerical model of landscape evolution based on the CAESAR-Lisflood model (Coulthard et al., 2013). The model is a cellular automaton model, whereby the grid cells in the model domain each have their own parameters associated with them (such as elevation, water depth, water

velocity, sediment load, etc.) and the change in these parameters is determined by the state of the neighbouring grid cells.

Unlike many models of landscape evolution, this model features an explicit calculation of water flow through the landscape based on a highly simplified form of the shallow water equations (Bates et al., 2010). Most other models, in order to simulate long term evolution, rely on approximations for water discharge based on drainage area, and assume instantaneous run-off of water under hydrological steady-state conditions. The HAIL-CAESAR model is therefore suited to modelling both flooding and erosional processes. It can be run as a standalone hydrological model (no erosion) if so desired.

The model is also developed to run on parallel computing architectures (Multi-core/cluster computers).

Quick guide if you know roughly what you are doing

Overview of running HAIL-CAESAR if you have used it before:

1. Prepare your DEM so it has an outlet point touching the side of the DEM bounds. (You may need to rotate it or draw on a channel outlet)
2. Make sure the DEM is in ascii (.asc) format.
3. Prepare a rainfall input file (A list of rain rates per time step, e.g. hourly rainfall). Just a plain text file will do.
4. Put all your input files and the paramter file in the same folder. Make sure you specify this path in the parameter file as well.
5. If not already done so, compile the code with the makefile: *make*, when in the top-level directory.
6. Double check that all your parameters are set correctly in the parameter file.
7. Run the program with *./HAIL-CAESAR.exe /path/to/data/ parameter_file.params*
8. View the timeseries data and output rasters with a scripting language or GIS of your choice.

1.2 Background

The model is essentially a stripped down translation of the CAESAR-Lisflood model, but without the GUI, and it runs on Linux unlike CAESAR-Lisflood, which is Windows only (as of 2017). If you are happy to run on Windows, and would prefer to use a model with a GUI-mode where you can view the output of your simulation in real-time, then CAESAR-Lisflood is a good choice for this. HAIL-CAESAR is designed to be faster and cross platform, will scale up to running across multi-core compute workstations and cluster PCs, and can be run to simulate multiple-member ensembles to explore model sensitivity. One of the drawbacks to the original model is that you need a dedicated Windows box to do your simulations, you can't send them off to a cluster computer-type facility, which typically run Unix. (Unless you have a bunch of spare Windows PCs. . .)

The raster data handling is done using the libraries from the LSDTopoTools package, and so can be used in the same workflow as LSDTopoTools to do topographic analysis on your model output.

1.3 Installation

1. Download the software, either as a zip/tar file, or by cloning the latest version from github.
2. The makefile is currently set up to use the gcc compiler, edit Makefile if you wish to change this, otherwise:
3. Run make in the same folder as *Makefile*.
4. Wait for the code to compile, when finished it will have produced an executable called *HAIL-CAESAR.exe*.

5. You are now ready to run the model!

1.3.1 Dependencies

Note: *Your compiler must support the C++11 standard.* Almost all compilers do as of 2016, as long as you are running a fairly recent version. (GCC version 4.7 onwards should be fine, preferrably GCC 4.8+) Currently, support is not usually enabled by default. (UPDATE 2016: It is in GCC 6.1) The C++11 flag to turn on support for the standard is included in the makefile. The code uses some features of the C++11 language standard not supported by older standards. You don't need to know anything about it further to use the model.

The parallel version of the code uses the OpenMP libraries, which are powerful (and somewhat magical...) libraries for compiling the code to run in parallel. These are widely supported libraries and many systems will come with them pre-installed. *But you may need to install the 'gcc-devel' package on linux, if using gcc*. Again, the compiler flag is taken care of in the makefile. The code has been tested on the *gcc* (versions 4.8 and above) Cray compiler, and *icc* (Intel) compiler v12. The code has not been extensively tested on Windows, but a recent enough Microsoft C++ compiler (such as comes with Visual Studio), should be sufficient to compile the code. You may need to modify the *Makefile* when using other compilers.

All other libraries are supplied with the source code (The TNT, *Template Numerical Toolkit* library, which provides array/matrix structures for the model data). You do not need to install them separately.

1.4 Running the Model

The model runs from the command line/terminal/console. You specify the model executable name (HAIL-CAESAR.exe) followed by the path name to the parameter file and the parameter file itself. The model will print out updates to the terminal window regularly, keeping you updated to the stage it is at and if there are any errors. The DEM of your catchment must be present in the same folder as your parameter file and must be correctly formatted.

You need a minimum of three input files:

1. Parameter file
2. DEM file of your catchment (currently only ASCII format is supported, sorry `.bil/flt` fans!)
3. Rainfall time series text file (There is currently no option to generate rainfall within the model, but this is coming soon)

Your input files (DEM etc, parameter file) can be in any folder you like, as the path to the input files is specified in the parameter file. This means the executable file can be kept separate, once it has been compiled.

The model is run like so:

```
./HAIL-CAESAR.exe [PATH-TO-FOLDER-WITH-INPUT-FILES] ParameterFile.txt
```

As you can see, the executable takes two arguments, the first is the path where your parameter file and input files can be found, the second is the name of your parameter file. Note that the names of the input DEM and rainfall file are specified in the parameter file.

When the model runs, it will print to screen the parameters that have been read from the parameter file, for a sanity check. The debug version prints a lot of other information to screen, to help find bugs and errors. I suggest turning this off and running with the optimised version unless you are trying to trace a bug. The program will update you at certain stages of the data ingestion process. (This usually only takes a few seconds). When the model runs, a counter displays the number of elapsed minutes in model-time. (There is an option to turn this off in the parameter file - set `debug_print_cycle_on` to `no`.

The model also prints out when it is writing output raster files, such as water depths, elevation difference etc. These files are outputted to the directory specified in the parameter file.

1.4.1 Outputs

HAIL-CAESAR generates similar outputs to CAESAR-Lisflood, i.e. a timeseries text file of the water discharge, and sediment fluxes; as well as raster files for water depths, elevations, erosion amounts, and so on. These can be outputted at an interval specified in the parameter file. Output files can be saved in ascii (.asc) form only currently.

1.4.2 DEM preparation

> **Important**
>
> You will need to check your DEM is correctly formatted before use. LSDCatchmentModel has specific requirements about DEM layout.

Currently, you will have to prepare your own DEM as a separate stage in the workflow. (Using whichever GIS tool you like, or preferably our own software!). The DEM should be set up so that one side of the catchment will act as the flow exit point. If you do not have the intended catchment outlet point touching one of the DEM edges, you will get unrealistic pooling of water and flood the entire catchment, as water will not be able to leave the model domain. **In other words: There should be no 'NODATA' values between the intended outlet cell(s) and the edge the DEM file.** This is very important for the model to work correctly. You may have to rotate your DEM or add on a channel artificially so that your catchment has a suitable outlet point at one side of the DEM file.

> **Note**
>
> The model will actually route water off **all** edges of the catchment, if the geometry of your catchment allows it. This might be fine for your intended use, but note that the discharge timeseries file will report total water discharge and sediment output as a total from ALL edge cells, not just the ones you think are the main catchment outlet point. As a side effect, you can use the model to simulate range scale runoff and multiple catchments, just be aware that you will get one value for total discharge for the whole DEM.

Technically, the DEM doesn't need to be pit-filled, but it may be worthwhile to do so as parts of the model can be sped up when the catchment is in a low-flow or steady-flow state. Again, it depends on your intended usage of the model.

1.4.3 Model run time controls

A sample parameter file is provided for the Boscastle floods simulation. This is a 48-hour simulation using a 5m DEM, over a catchment 3km x 5.5km (about 700000 grid cells). It will take about 2-3 hours to run on a mid-range Desktop machine. (You can dramatically speed this up by using a coarser DEM.) Number of domain grid cells is the main control on compute time. With a multi-core machine, the run time can be significantly reduced, depending on the number of cores you have.

> **Tip**
>
> If running in a multi-core environment, you can get this down to around 11 minutes using a 48-core machine.!

Note that some of the parameters in the paramter file will have an effect on model run time. For example: **in_out_difference**, **courant_number**, and many of the minimum threshold values for flow and erosion can all be tweaked to get model speed up in some way. See the parameter file guide for advice.

1.5 Parameters

The parameter file (`HailCaesar.params`) is read by the program when it starts, setting the variables for various components of the model, as well as the names of the DEM files and supplementary input files needed for the model simulation. It is a plain text file and can be called anything you like. By convention, the example parameter file supplied with the code has the suffix `.params`, but you may use whatever you like.

Anything in the parameter file preceded by a `#` will be treated as a comment. You are free to comment away in the parameter file to your heart's content.

The order of the parameters in the parameter file is not strict, but we stick to the convention used in the sample file for reference here. The parameter names must not be changed, and must be lowercase.

1.5.1 Parameter File

This section contains explanations on each of the parameters

File Information

`read_fname`

The name of the DEM (digital elevation model) file that contains the terrain surface/topography. **No file extension** should be included here.

`dem_read_extension`

The extension and format of the DEM file. The format is inferred from the extension given.

Options

- asc (ASCII Grid/ESRI Grid)

`dem_write_extension`

The extension of the output rasters. This extension determines the format of the files as well, so specifying `flt` for example will also write them out in binary format as well as appending the extension itself.

Options

- asc
- flt
- bil

`read_path`

The full path that the input files will be read from. e.g. DEMs, rainfall files, grain data files.

write_path

The full path that the output files will be written to.

write_fname

The output name for the timeseries output. (e.g. Hydrograph and Sedigraph.) Will be a plain text file.

timeseries_save_interval

The interval that data is written to the timeseries file. Should be in model minutes, I.e. simulated minutes. (Integer)

Supplementary Files

hydroindex_file

When using the spatially variable rainfall configuration. The hydroindex file name (with extension). Only ascii (asc) currently supported. This file is a DEM the same resolution and extent as the main terrain DEM, but the cell values are integers from 1, marking each region that will receive a different rainfall input record.

rainfall_data_file

Name of the text file timeseries containing the rainfall rate at each rainfall input timestep in mm/hr.

grain_data_file

Name of the file that contains the graindata information. I.e. if you are prepopulating the model domain with grain size and grain fraction information, you need tto set this file. More details are given in the input files section.

bedrock_data_file

Name of the DEM that will be used to represent the bedrock layer, if the model is configured with this mode. (`bedrock_layer_on`). If this file is specified, then the main terrain DEM, (`read_fname`) will still act as the uppermost terrain surface, with the bedrock DEM situated below it - but you must have correctly set the elevations on the bedrock DEM file to be less than the surface DEM. (i.e. the model currently cannot do this for you.)

Run time and timestep

These parameters control things like time stepping and how long the model runs for.

min_time_step

The minimum timestep used in the model.

- **Units, data type**: Model seconds, integer
- **Default**: 1 (second)

max_time_step

The maximum timestep used in the model

- **Units, data type**: Model seconds, integer
- **Default value**: 3600 (seconds)

run_time_start

The model run start time - only zero is currently supported as the restart function is currently implemented.

- **Units, data type**: Model hours, integer
- **Default value**: 0

max_run_duration

The maximum run duration of the model. I.e. the model stops when it reaches this point.

- **Units, data type**: Model hours, integer
- (No default value)

memory_limit

This can be ignored, it was a feature of CAESAR-Lisflood to restrict the size of the arrays such as the huge grain size array in memory. It is set to 1.

Sediment Transport

transport_law

Determines which sediment transport law is used by the model. The options are Wilcock and Crowe or Einstein-Brown. See the pages in the documentation for more detail.

Options

- `wilcock`
- `einstein`

max_tau_velocity

Limits the maximum velocity used to calculate sediment transport. The default is 5. It is rarely invoked except in very steep slopes.

active_layer_thickness

This controls the thickness of layer representing the surface, bedload or subsurface. It should be around 0.1 to 0.2. It must be at least 4 times the erode_limit parameter. (See below)

chann_lateral_erosion

In channel lateral erosion rate. Prevents overdeepening feedback. See explanation here.

erode_limit

Maximum erosion limit per cell (or deposition). Prevents numerical instabilities by transferring too much between cell to cell. Should be around 0.01 for 10m or less DEMs, slightly higher for coarse DEMs.

suspended_sediment_on

Turns on suspended sediment for the first fraction only at present. (i.e. the smallest grainsize fraction.

(**yes** | **no**)

read_in_graindata_from_file

Reads in the initial grain size data from a file. Normally the initial distribution of grainsizes is uniform across the landscape if this is not specified.

bedrock_layer_on

(**yes** | **no**)

Lateral Bank Erosion

Please note that this feature is untested in HAIL-CAESAR as of yet.

lateral_erosion_on

Turns on lateral erosion in channels.

(**yes** | **no**)

lateral_erosion_const

See Coulthard and Van de Wiel (2007) for details. This parameter sets the lateral erosion rates, and ought to be calibrated to a field site. Values can ramge from 0.01-0.001 for braided rivers; 0.0001 for meandering channels. This value is grid cell size independent.

edge_smoothing_passes

Determines how well smoothed the curvature of the channels will be. Should be set as the frequency of meanders/distance between two meanders in grid cell size.

downstream_cell_shift

The gradient used to determine lateral erosion rates can be shifted downstream, allowing meander bars and bends to migrate down channel. Suggested values are around 10% of the edge_smoothing_passes parameter. So around 1-5.

lateral_cross_chan_smoothing

The maximum difference allowed in the the cross channel smoothing of edge values. After calculating the radius of curvature for the outside of a meander bend, the model must interpolate this value to determine how much sediment can be moved laterally. Smaller values result in better cross channel gradients, but take up more CPU time. Typical default value is 0.0001. If your channels are well resolved by DEM resolution, you may need to reduce by an order of magnitude.

Hydrology

hydro_model_only

Runs the model as a hydrological/flood inundation model only if set to yes. Will turn off all the erosion routines and terrain will be fixed. Speeds up run times considerably.

(**yes** | **no**)

topmodel_m_value

As well as the water routing sub-model, LSDCatchmentModel also calculates the discharge based on Beven's TOPMODEL (i.e. discharge approximation based on drainage area and topography. The model contains the infamous m parameter, which varies depending on environment. You should consult the literature for appropriate values.

For catchment mode this is an important variable as it controls the peak and duration of the hydrograph generated by a rain event. It is the same as the 'm' value in TOPMODEL, that CAESAR-lisflood's hydrological model is based on. Typical values for m are from 0.02 (meaning low flood peaks and long duration hydrographs) to 0.005 (flashier peaks) and examples of values used can be found in the CAESAR and TOPMODEL literature.

in_out_difference

If greater than 0, allows the model to run faster in periods of hydrological steady state. If the difference between water entering the catchment and water leaving the catchment is equal to or less than this value, the model will increase the time step. The time step will then be determined by erosional and depositional processes, which are typically much slower acting. Can be set to a low mean annual flow value for the river.

min_q_for_depth_calc

Threshold for calculating flow depths. The model will not calculate flow depths when the discharge at a cell is below this value, avoiding CPU time spent calculating incredibly small flow amounts. Should be set to approximately 10% of grid cell size. e.g 0.5 for a 50m DEM.

max_q_for_depth_calc

An upper discharge threshold that will prevent water being added above the given discharge threshold. Typically 1000.0, but lowering the value will shift the balance of water being added to the headwaters, rather than lower down through the catchment.

water_depth_erosion_threshold

If water depths are below this threshold, the model will not calculate erosion for that cell in that timestep. Used to prevent CPU time being spent on incredibly small amounts of erosion calculations.

slope_on_edge_cell

The slope used to calculate water flow on the edge of the DEM (since there is no neighbouring cell to calculate the downstream gradient. You should set this to approximately the same as the average channel gradient near the outlet of your river.

evaporation_rate

Untested/unimplemented yet.

courant_number

Controls the numerical stability and execution speed of the flow model. See Bates et al (2009). Typical values should be between 0.3 and 0.7. Higher values will speed up the model, but are more unstable. Parameter is dependent on grid cell size. DEMs of 20-50m resolution can use values of 0.7, finer DEMs (e.g. <2m) will need the lowest value of 0.3, but do not set it lower than 0.3.

froude_num_limit

Restricts flow between cells per time step, as too much can lead to checkerboarding effects. If this happens the froude number can be lowered. The default value of 0.8 results in subcritical flow - flow dominated by gravitational forces and behving in a slow or stable way. A value of 1 results in critical flow, which may be ok for shallow flows at coarse grid cell resolutions. Note that reducing flow with the Froude number will reduce the speed of a flood wave moving downstream.

mannings_n

A roughness coefficient used by the flow model. Values can be looked-up here. http://www.fsl.orst.edu/geowater/FX3/help/8_Hydraulic_Reference/Mannings_n_Tables.htm

hflow_threshold

This threshold prevents water being routed between adjacent cells when the gradient is incredibly small. A good default value is 0.00001.

Precipitation

rainfall_data_on

Reads in rainfall data from a rainfall.txt file timeseries.

(**yes** | **no**)

rain_data_time_step

The timestep of the rainfall inputs. Note: even if you set this to something other than `60` (for hourly rainfall), the units are always mm/hr. I.e. this is the instantaneous rainfall rate at each timestep.

spatial_var_rain

Use spatially variable rainfall inputs over the model domain. This requires a rainfall.txt file with multiple columns (for each region of the catchment receiving different rainfall rates.) and a hydroindex file to map rainfall columns in the timeseries file to the correct cells in the model domain.

num_unique_rain_cells

The number of rainfall regions/cells in your rainfall data and hydroindex file.

spatially_complex_rainfall_on

Not fully implemented yet.

(**yes** | **no**)

interpolation_method

Not fully implemented yet.

generate_artificial_rainfall

Not implemented yet.

(**yes** | **no**)

Vegetation

vegetation_on

(**yes** | **no**)

grass_grow_rate

vegetation_crit_shear

veg_erosion_prop

Hillslope

creep_rate

slope_failure_thresh

soil_erosion_rate

soil_j_mean_depends

call_muddpile_model

Output Rasters

raster_output_interval

write_waterdepth_file

(**yes** | **no**)

waterdepth_outfile_name

write_elev_file

(**yes** | **no**)

write_elevation_file

(**yes** | **no**)

write_grainsize_file

(**yes** | **no**)

grainsize_file

write_elevdiff_file

(**yes** | **no**)

elevdiff_outfile_name

raingrid_fname_out

Debug Options

CHAPTER 2

Indices and tables

- genindex
- modindex
- search