Computing flowing avalanches using AVAC: the r.avac GRASS module

October 2020

Vincent Bain[1]

[1]Toraval France, 2838 route des Annuits, F-73400 Ugine

1 Introduction

The AVAC code is used to run numerical simulations of flowing avalanches. It is based on the ClawPack software and its GeoClaw library. Preparing the input data and displaying the output files may be difficult. Here we propose an additional module to the GRASS geographic information system (GRASS AddOn) called `r.avac`. This document provides instructions for a quick start.

The `r.avac` module involves several programs commonly found in GNU/Linux repositories. It has not been tested for Mac OS and Windows operating systems. The easiest way to avoid compatibility problems is probably to work on a workstation running under GNU/Linux, physically or virtually (e.g., Oracle VM VirtualBox).

2 Prerequisites

We assume that the users have followed the installation instructions for the AVAC code, i.e. they have a functional Clawpack environment, in particular:

- The environment variable $CLAW points to the Clawpack installation directory.
- the `qinit_module.f90` file located in the `$CLAW/geoclaw/src/2d/shallow` directory has been correctly modified (see AVAC documentation).

You must install the GNU/parallel and OpenMP programs in order to take advantage of parallelisation.

Finally, the `r.avac` module requires the Temporary features of GRASS GIS, so it is advisable to use a GRASS version higher than 7.0.

3 Installation

Clone the GitHub ravac repository

```
git clone https://github.com/xyleme/avac
cd ravac
```

The `ravac` directory contains three elements:

- The directory named `avac`, which must be moved into the geoclaw `examples/` directory:

  ```
  mv avac/ $AVAC/geoclaw/examples/
  ```

- The `r.avac` executable file, which must be moved into a directory recognized by GRASS GIS (and thus be included as an AddOn source). The module in its current version does not appear in the official GRASS GIS repositories, so it cannot be installed from the `g.extension` utility. Rather than placing `r.avac` in the dedicated directory (`~/.grass7/addons/bin/`) we prefer to place it in a separate directory `/usr/local/grass-addons/`: `$ mkdir /usr/local/grass-addons/`

  `$ mv r.avac /usr/local/grass-addons/` then declare the corresponding environment variable in the `.bashrc` file (if the system shell is bash), as follows: `export GRASS_ADDON_PATH=/usr/local/grass-addons/`

- The `boussolenc.gpkg` file which contains a set of cartographic data that can be used to test `r.avac` .

4 Input data preparation

As stated in the preamble, the `r.avac` module aims to simplify the use of AVAC by providing the user with simple tools for preparing the required input data and defining model parameters. GRASS provides an appropriate framework and routines to that end. The user familiar with GIS—and in particular GRASS—can refer directly to the following section, in which we detail the procedure to run the code.

4.1 Location and Mapset

GRASS requires a set of files that defines the reference map coordinate system (directory named *Location*) and data sets (subdirectories *Mapsets*). Let us consider that we are working on a slope located in the French Alps, therefore we use the Lambert 93 coordinate system, and create an L93 location. When the program is started in graphical mode for the first time, a sequence of steps guides the user in determining a suitable reference system (see figure 1). The program suggests creating a mapset, which has the user's name by default.

This step can also be executed from the command window using the following command:

```
grass79 -c EPSG:2154 L93
```

GRASS GIS allows the user to manage raster data, vector data and attribute data. The `r.avac` module uses these three types of data.

4.2 Raster data: importation of topographical data

The `boussolenc.gpkg` file is a geopackage whose data are georeferenced in the Lambert 93 system. It includes a *digital elevation model* (DEM) named `topo`, and a shaded relief map named `shading` that we will import into our location by typing the following command lines in the command window:

```
r.in.gdal input=boussolenc.gpkg gdal_doo='table=topo_2m' output=topo
```

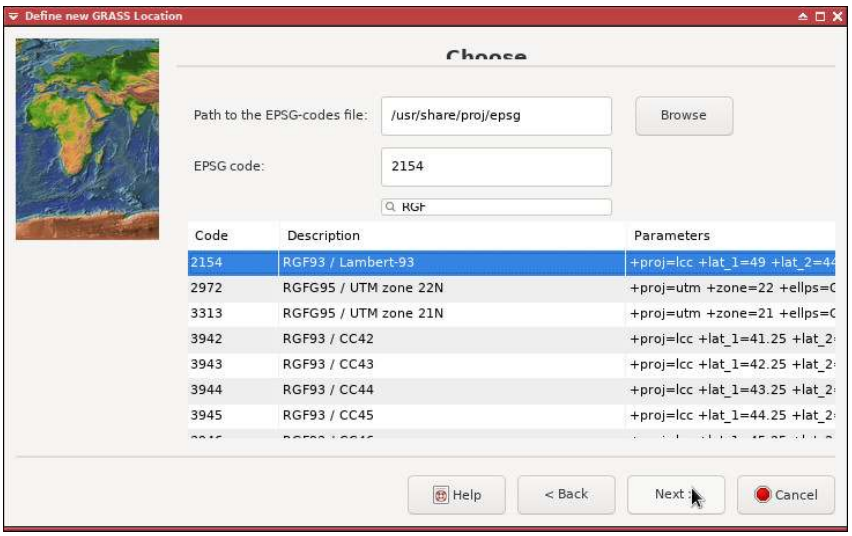

Fig. 1: The procedure for creating a location in GRASS GIS asks the user to specify a reference coordinate system.

```
r.in.gdal input=boussolenc.gpkg gdal_doo='table=ombrage' output=shade
```

In the *Layer Manager* window, the user can add these maps to the current view (see figure 2).

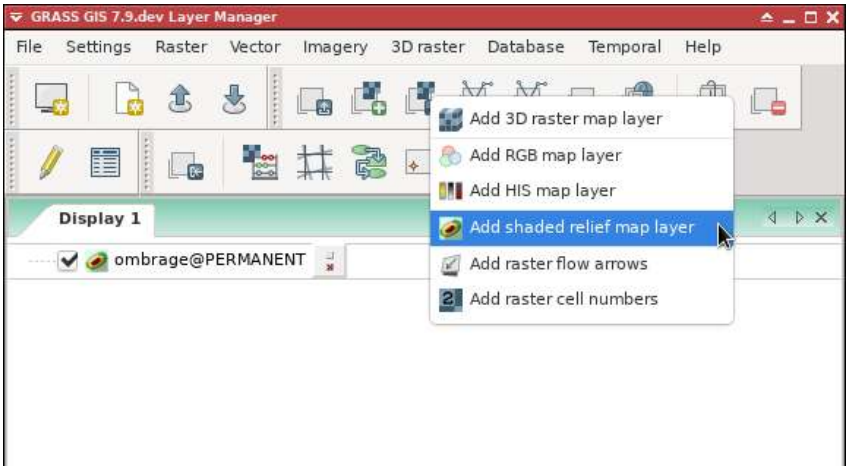

Fig. 2: The *Add various raster map layers > Add shaded relief map* menu makes it possible to combine shaded relief and DEM.

To provide AVAC with the desired topographic data, `r.avac` transforms the raster map and passes it on to the code by means of the file named `topo.asc`. The resolution and geographical extension of this file are controlled by the user when defining the working region.

4.3 Notion of region

A central notion of GRASS GIS is the *region*. It is a set of numerical values indicating the position, range, and resolution of a grid that discretises the working area (or computational

domain). By default in a blank location, this region is defined by a single cell located at the origin of the current projection's Cartesian frame. **If we want GRASS to process our data properly, it is essential to indicate the region parameters related to the working area**. In this case, run the following command to make the region coincide with the extension and resolution of the digital elevation model (imported during the previous step):

```
g.region -p rast=topo
```

The `-p` option forces to display the new region parameters in the output.

The map display window allows the user to view the current region (see figure 3). The user can also define the region extension interactively in the map display window (Various zoom options icon > set computational region extent interactively).

Finally, it is possible, and desirable, to determine the resolution on which we wish to work. The previous command tells us that the sampling step of the topography we imported is 2 m. We can decide to work with a mesh grid of 5 m, by giving the following instruction:

```
g.region -pa res=5
```

The `-a` option forces the region definition by imposing that its dimensions are proportional to the desired resolution.

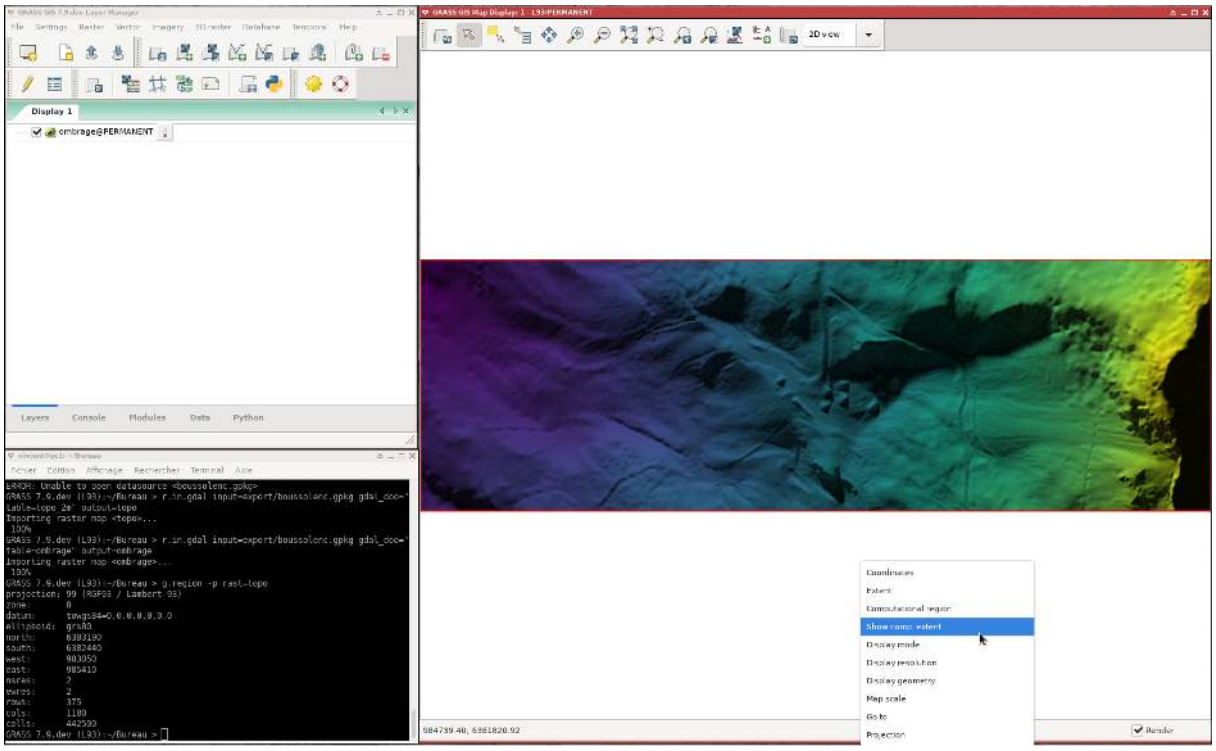

Fig. 3: Shaded relief display of the digital elevation model. The *map display* window displays the current region as a red frame.

The current version of `r.avac` rpovides a rough support of AMR (adaptive mesh refinement) functionality; ordering GeoClaw to work locally at a given AMR can be done by specifying a rectangular area (via `amr` option) previously set within GRASS GIS: the user determines a rectangular polygon which category (`cat`) stands for theamr wished level (the current version only support level 2). A quite handy way to proceed is as follows:

- interactively define a new region in the map display (see above);

- create a vector map sourced from this new region setup:

```
v.in.region cat=2 out=amr2 --o
```

4.4 Vector data: creation of a vector map

The file named `boussolenc.gpkg` is a geographic dataset containing among others a contour level layer, which is easily imported in GRASS GIS:

```
v.in.ogr in=boussolenc.gpkg layer='cbn' out=cbn
```

On top of the previously imported raster stack, it provides a neat perception of terrain shape.

The AVAC code needs to know the location of the avalanche starting area(s), as well as the initial height of snow mobilised in each area. The `r.avac` module provides these initial data in the form of a vector map called `ZA`, comprising as many polygons as there are starting zones. The initial snow height in each area is contained in the attribute called `h`.

The `boussolenc.gpkg` file includes an example of a vector layer named `ZA` in the format expected by `r.avac`. To import it into GRASS, run the following command line:

```
v.in.ogr in=boussolenc.gpkg layer='ZA' out=ZA
```

The following commands ensure that the `h` field is present in the ZA layer's attribute table:

```
v.db.connect ZA -p
db.columns ZA
```

The user is referred to the GRASS GIS documentation for further information, in particular for a presentation of:

- vector models https://grass.osgeo.org/grass76/manuals/vectorintro.html,
- attributes https://grasswiki.osgeo.org/wiki/Vector_Database_Management, and
- the editing tools [1] (see figure 4).

To provide AVAC with the required input data, `r.avac` discretises the vector map and passes it on to the code via the file named `initials.xyz`.

4.5 r.avac features

Rheologic law AVAC code allows to choose between two snow rheologies. Friction can be expressed:

- either by a *Voellmy* law (with two parameters μ, ξ), or
- by a *Coulomb* law (with a single parameterμ).

The user tells `r.avac` which law to use providing the `rheology` argument:

- `rheology=0` stands for Voellmy, the default value ;
- `rheology=1` stands for Coulomb.

Choosing `rheology=1` cancels ξ parameter.

[1] https://grass.osgeo.org/grass76/manuals/wxGUI.vdigit.html

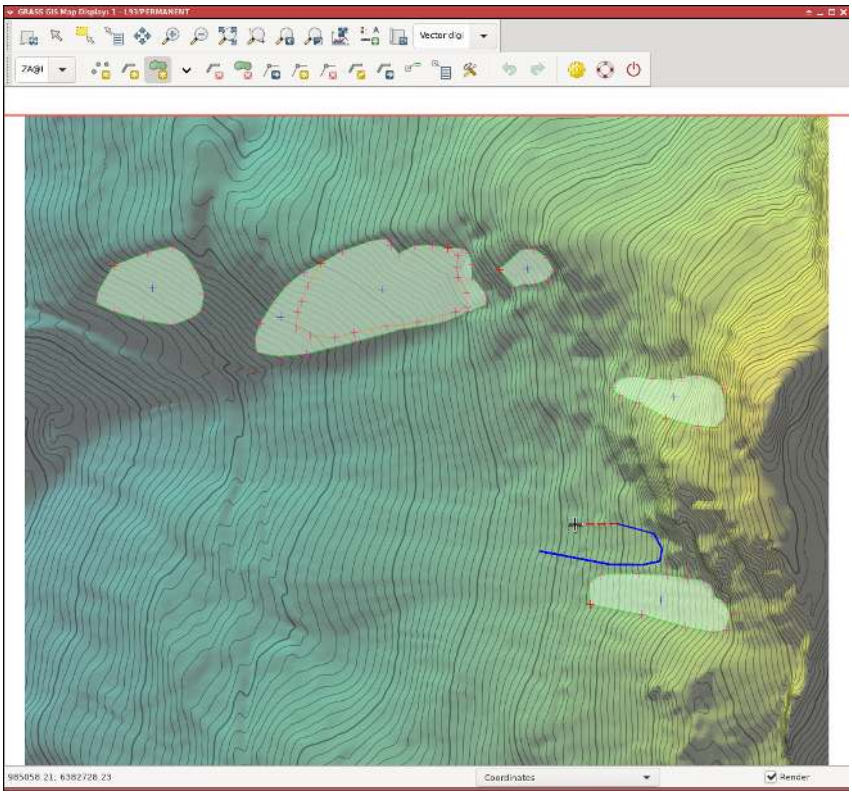

Fig. 4: The ZA vector layer can be edited to modify, add, delete start areas.

Snow heights In this version `r.avac` can modulate the snow height, provided by the `depth` parameter or the `h` vector attribute, taking into account the local steepness of the slope in the starting area. The program uses a modified version of the de Qervain formula:

$$h(z) = h_0(z)\frac{\sin\theta_{ref} - \nu\cos\theta_{ref}}{\sin\theta - \nu\cos\theta},$$

where h_0 is the snow depth (on a flat ground) at a z given altitude, θ_{ref} is a reference angle (slope for which the correction factor is 1), θ is the slope angle of the starting area at the z altitude, and snu is an empirical coefficient. Default values $\nu = 0,2$ and $\theta_{ref} = 30$ can be tweaked inside the `r.avac` source code.

In addition, this version of `r.avac` allows to fine tune snow depths in the starting area by specifying a hypsometric gradient thanks the `grad` argument, expressed as a triplet (depth,gradient,altitude), e.g.

```
grad=1.8,0.03,1850
```

means the snow depth is 180 cm at 1850 m, and the hypsometric gradient is 3 cm per 100 m elevation. Giving this option to `r.avac` the initial condition grid values override any value provided by `depth` parameter or the `h` vector attribute.

5 Running r.avac

5.1 Syntax

`r.avac` can be executed at the terminal . For a first use, it may be more convenient to use the module's user interface. To that end, simply run at the grass prompt: `r.avac --ui`

Every argument in the *Required* tab must be filled. The *Optional* tab contains arguments whose values are set by default, including the AVAC model calibration parameters (see AVAC documentation).

The `--h` option yields the command's full syntax:

```
r.avac --h
```

```
Usage:
 r.avac [-mrc] simul=name dem=name za=name [depth=name] [amr=name]
   [dt=string] [t=string] [rheology=string] [rho=string] [xi=string]
   [mu=string] [grad=string] [vthres=string] [bslope=string]
   [--overwrite] [--help] [--verbose] [--quiet] [--ui]

Flags:
  -m   compute max(h) and max(p) maps throughout simul strds
  -r   reset existing simul strds
  -c   run geoclaw 'make clobber'

Parameters:
     simul   output simulation STRDS
       dem   input elevation raster map
        za   input starting area vector map
     depth   depth column for starting area
       amr   input AMR zone map
        dt   time increment
             default: 20
         t   simulation duration
             default: 100
  rheology   type of rheology
             default: 0
       rho   snow density
             default: 300
        xi   Voellmy xi
             default: 1500
        mu   Voellmy mu
             default: 0.15
      grad   depth hypsometric gradient
    vthres   velocity threshold
             default: 0.05
    bslope   beta slope
             default: 1.1
```

With a little practice, the user may prefer to execute `r.avac` from the GRASS console. For example:

```
r.avac -mr simul=sim2 rheology=0 dem=topo za=ZA dt=1 t=40 \
mu=0.2 xi=800 rho=300 grad=1.7,0.03,1850 amr=amr2 vthres=0.2
```

The advantage of using the command line rather than the user interface is that it is easier to do several iterations of the code by changing one or several parameters. Taking advantage of the command line history in the GRASS command window makes things easier.

5.2 How it works

When the command is run, the script provides the AVAC code with topographic data (`topo.asc`), initial conditions (`initial.xyz`), computational domain (according to the region parameters previously defined), and the computation parameters in `voellmy.data`, `AddSetrun.py`, and `setrun.py` files.

Once the computation has been done, `r.avac` processes the files `fort.qxxxx` produced by AVAC in order to import the flow depth $h(t)$ and kinetic pressure $p(t) = \frac{1}{2}\varrho u^2$ into GRASS, where ϱ and u are the snow density and velocity, respectively. The tiles generated by the code are then stitched to obtain a set of rasters describing the flow at the time step `dt` prescribed in the input file. Using the previous command example, rasters named `h.sim2_xxxx` and `p.sim2_xxxx` are incorporated respectively in rasters time series (spatial temporal raster dataset, strds): one named `sim2_h` describing the flow depth (expressed in m), the other named `sim2_p` describing the kinetic pressure p (expressed in kPa).

If the user wishes to interrupt the execution of the code, he can do so in graphic mode by pressing the Stop button. In line command, the <ctrl+c> shortcut stops the computations. The script performs cleaning operations before leaving (purging some parameters, deleting temporary files).

6 Visualisation and export

The user can visualise the simulation output by launching the interactive `g.gui.animation` module.

The colorimetric chart associated with each series can easily be modified. For example, if the user want to apply the pressure chart used in Toraval documents, which is recorded in the `p_toraval.file` file attached to our archive:

```
t.rast.colors input=sim5_h rules=[/path_to/]p_toraval.file
```

The `g.gui.animation` module offers the possibility to export animations in several forms. When the number of maps in the time series is large (typically a thousand files), it is no longer possible to view or export the series. The user can then choose to export a series of images in png format for post-processing with other tools, such as the `ffmpeg` software:

```
g.list rast | grep h.sim2 | parallel r.out.png in={} out={}
ffmpeg -f image2 -r 12 -i "h.sim2_0\%03d.png" -vcodec png out_h.avi

g.list rast | grep p.sim2 | parallel r.out.png in={} out={}
ffmpeg -f image2 -r 12 -i "p.sim2_0\%03d.png" -vcodec png out_p.avi
```

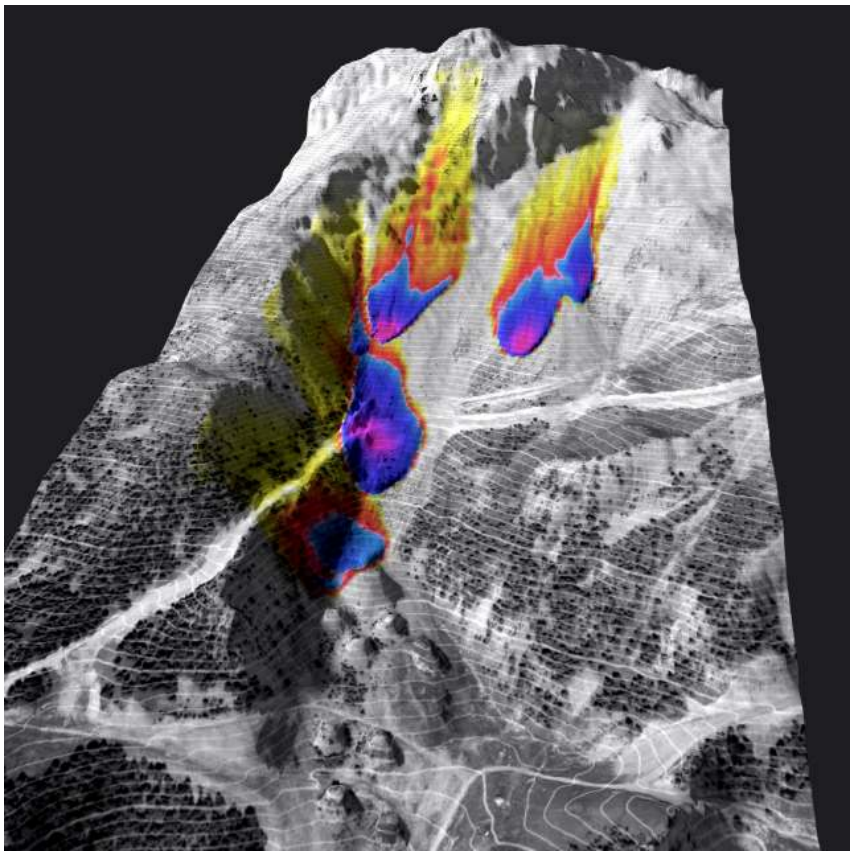

Fig. 5: Example of post-processing performed using Blender to produce an animation combining flow depth (displacement) and pressure (color). See video at http://telec.toraval.fr/boussolenc.avi

.