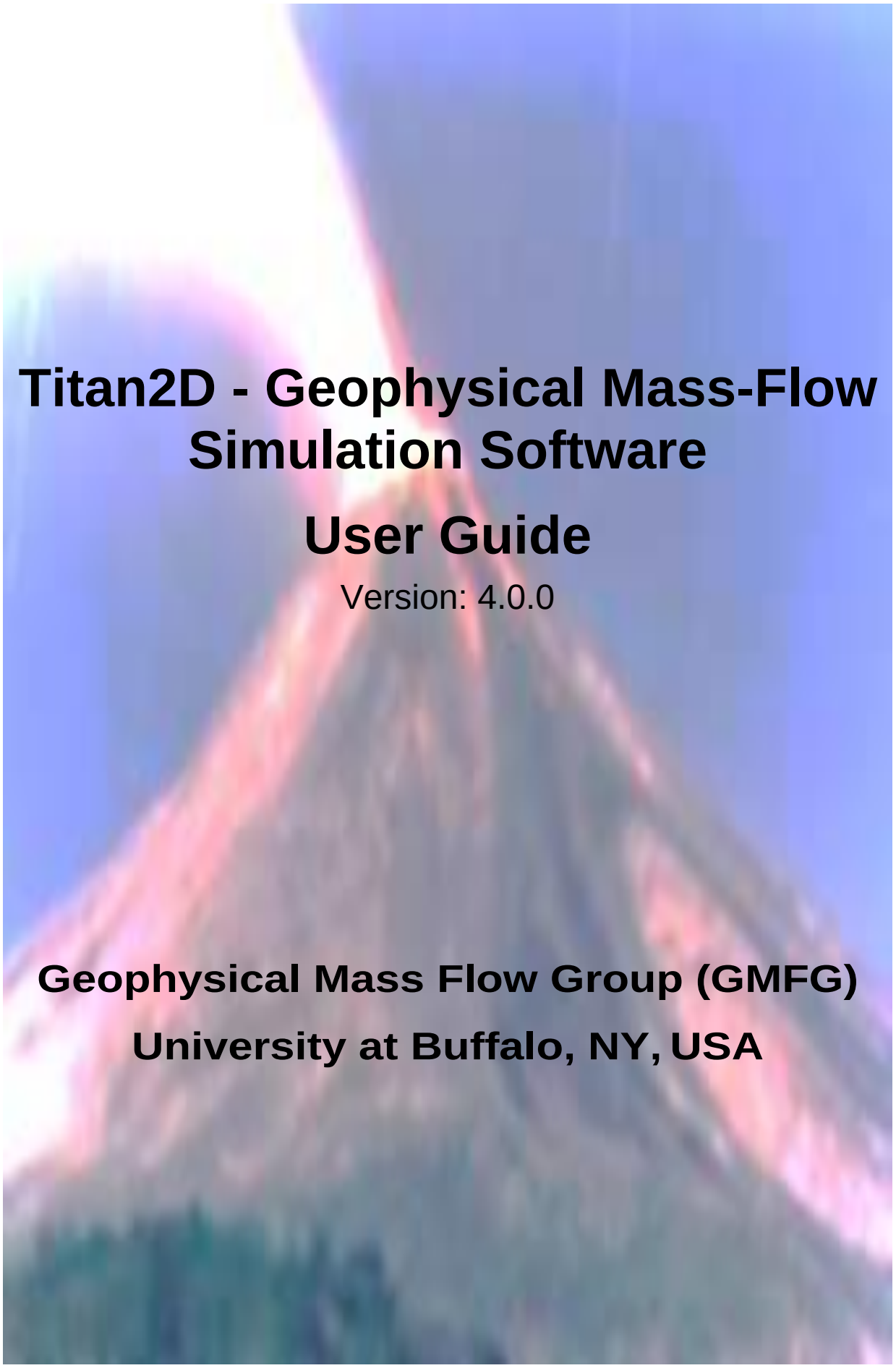

# Titan2D - Geophysical Mass-Flow Simulation Software

# User Guide

Version: 4.0.0

**Geophysical Mass Flow Group (GMFG)**

**University at Buffalo, NY, USA**

## Contents

## Table of Figures

# 1 Introduction to Titan2D

Titan2D is a computer program developed by the GMFG (Geophysical Mass Flow Group) at State University of New York at Buffalo - led by Dr. Abani Patra (Professor, Dept. of Mechanical and Aerospace Engineering), for the purpose of simulating granular avalanches over digital elevation models of natural terrain. The program is designed for simulating geological mass flows such as volcanic flows, debris avalanches and landslides. Titan2D combines numerical simulations of a flow with digital elevation data of natural terrain supported through a Geographical Information System (GIS) interface.

The Titan2D program is based upon a depth-averaged model for an incompressible continuum, a "shallow-water" granular flow. Please see Appendix A for the shallow-water model conservation equations solved by Titan2D. We will briefly review the functionality here; details of the modeling and numerical methodology may be found in the literature [1] [2]. The conservation equations for mass and momentum are solved with different rheologies modeling the interactions between the grains of the media and between the granular material and the basal surface. The resulting hyperbolic system of equations is solved using a parallel, adaptive mesh, Godunov scheme.

---

[1] A.K. Patra, A.C. Bauer, C.C. Nichita, E.B. Pitman, M.F. Sheridan, M. Bursik, B. Rupp, A. Webber, Stinton, L. Namikawa, and C. Renschler, Parallel Adaptive Numerical Simulation of Dry Avalanches Over Natural Terrain, Journal of Volcanology and Geophysical Research, 139 (2005) 1-21

[2] E.B. Pitman, C.C. Nichita, A.K. Patra, A.C. Bauer, M.F. Sheridan, and M. Bursik, Computing Granular Avalanches and Landslides, Physics of Fluids, Vol. 15, Number 12 (December 2003)

The Message Passing Interface (MPI) [http://www-unix.mcs.anl.gov/mpi/] and OpenMP (Open Multi-Processing) [http://openmp.org/wp/] allow for computing on multiple processors, increased computational power, decreased computing time, and allows for the use of large data sets (note that MPI performance is currently slower than serial, choose OpenMP). Adaptive gridding allows for the concentration of computing power on regions of special interest. Mesh refinement captures the complex flow features at the leading edge of the flow, as well as locations where the topography changes rapidly. Mesh un-refinement is applied where solution values are relatively constant or small to further improve computational efficiency.

Titan2D supports several material models. Please see the Material Model and Map section for more details. These models assume a pile of granular material (or an effusive flux), pulled downslope by gravity. Friction between particles and between particles and ground resist this momentum. Governing equations for these models, the conservation of mass and conservation of momentum, are solved using approximate numerical solution methods, e.g. finite volumes etc. The direct outputs of Titan2D are flow depth and momentum. These may then be used to compute, at different points, field observable variables like run-up height, inundation area, discharge rates and time of flow.

Titan2D interfaces with users via a Java Graphical User Interface (GUI) (or for advanced users through a python script). Through this interface, the user inputs the parameters needed to successfully run the program such as pile dimensions, starting coordinates, internal and bed friction angles, and simulation time. The simulation is computed on a Digital Elevation Model (DEM) of the desired region and results can be displayed through the Titan2D viewer utilities, or other visualization software packages. The Titan2D viewer utilities are designed to present end-users with a clear representation of various properties of the mass flow such as pile height and velocity magnitude. The attributes embedded in the data elements that constitute the polygonal mesh are color-coded and applied as a texture over the terrain.

This user guide provides information for installing and running Titan2D. Included are sections which describe the graphical user interface and describe the Titan2D python input file which is now used to control the Titan2D simulations. For convenience, the Table of Contents contains hyperlinks to all listed sections.

# 2 New Features in Titan2D Release 4.0.0

In this completely redeveloped release of Titan2D, many improvements have been incorporated that add additional capabilities to the simulator and increase the efficiency Titan2D. These are summarized as follows:

## User Facing

- Improved runtime performance.
- Added support for several alternate rheologies to the Mohr-Coulomb model– namely the Voellmy-Salm and Pouliquen-Forterre models.
- Full restart capabilities to extend simulations.
- Expanded DEM format support to almost all known GIS Formats using the GDAL library.
- Provides a consistent GUI for running Titan2D on VHub and other 64-bit Linux platforms.
- Extended VHub's Titan2D capability to generate Google Earth Keyhole Markup Language Zipped files (KMZ). Please see the Titan2D Viewers section for more information.

## Technical

- Added OpenMP Processing and improved vectorization.
- Expanded Titan2D input file. Information previously contained in simulation.data, frict.data and scale.data is now consolidated into one python input script.
- Expanded profiling information returned.

# 3 Getting Started

## System Requirements

The Titan2D software is open source and is built using only other open source systems. It is designed for both low end single processor use and high end distributed/shared memory multi-processor use. The installation and use procedure is largely similar but there are small differences on each system.

## Titan2D Program File

The latest version of Titan2D can be obtained from the Titan2D GitHub repository: [https://github.com/TITAN2D/titan2d/releases].On VHub, Titan2D is installed and available via the Titan2D Mass-Flow Simulation Tool.

On other 64-bit Linux platforms, Titan2D can be installed from either a binary package or source. The binaries should work on most modern CPUs and Linux distributions. The binaries are only marginally slower than binaries specifically compiled for particular CPUs (some compilers can generate slower native code than the provided generic binaries). The provided binaries support multi-thread parallelization. At this time, MPI parallelization is present however its performance is not optimal. Therefore, at this time, OpenMP execution is the preferred way to utilize multi-core CPUs. Please see instructions below on how to install Titan2D from the binary package. In some cases, it may be preferable to install the Titan2D executables compiled from source and Titan2D source code is also available. Please see instructions below on how to install the Titan2D executables from source. After successfully installing Titan2D, the Titan2D executable, “titan”, will appear in the titan2d “bin” directory, you will need to access this “bin” directory to run Titan2D.

## Installation from Binary Package

**Download the binary package tarball to location where Titan2D binaries will reside:**

```
Wget https://github.com/TITAN2D/titan2d/releases/download/v4.0.0/titan2d-v4.0.0-Linux-64bit.tar.gz
```

**Uncompress the tarball:**

```
tar xvzf titan2d-v4.0.0-Linux-64bit.tar.gz
```

For convenience, you can add the location of binaries to the PATH environment variable in your .bashrc file, export PATH=<path to root of titan2d>/bin:$PATH

## Installation from Source Using Titan2d_dep Package

Currently the supplied generic binaries have a performance very close to natively compiled code, in some cases natively compiled code is actually slower. We are still working on improving MPI performance which does not scale at all now due to the dramatic increase in serial and OpenMP performance. In addition, Titan2D has a growing number of dependencies and only a small subset of their versions were tested and some versions are not compatible with Titan2D. Therefore, there is not so much reason to compile it by yourself. One of them: no matter what MPI version is needed.

Due to large number of dependencies, we provide binary package with most non-standard and version specific dependencies.

**Download Dependencies Package:**

```
wget https://github.com/TITAN2D/titan2d/releases/download/v4.0.0/titan2d_dep-v4.0.0.tar.gz
```

**Uncompress the tarball:**

```
tar xvzf titan2d_dep-v4.0.0.tar.gz
```

**Download the source tarball to location where Titan2D binaries will reside:**

```
wget https://github.com/TITAN2D/titan2d/releases/download/v4.0.0/titan2d-v4.0.0-src.tar.gz
```

**Uncompress the tarball:**

```
tar xvzf titan2d-v4.0.0-src.tar.gz
```

**Create building directory:**

```
mkdir titan2d_bld
```

**Configure, compile and install:**

```
cd titan2d_bld

../titan2d-v4.0.0/configure --prefix=/full/path/to/install --enable-openmp --with-titan2d-dep=/full/path/to/titan2d_dep
```

#--enable-portable will copy titan2d_dep to installation folder and

#thus original titan2d_dep can be removed

make -j 4

make install

**For VHub builds, need to also specify the --enable-vhub configure option:**

```
../titan2d-v4.0.0/configure --prefix=/full/path/to/install --enable-openmp --enable-vhub --with-titan2d-dep=/full/path/to/titan2d_dep
```

## Other Notes:

It is highly recommended to install Titan2D from source using the Titan2D dependency package, Titan2D_dep, as detailed in the previous section. Otherwise, follow the instructions in the INSTALL text file to install Python, PCRE (used by SWIG), SWIG, HDF5, GRASS and GDAL (if not already installed on your system). Next, follow the instructions to install Titan2D. To install Titan2D for OpenMP mode, include –enable-openmp as a configure command line option. To install Titan2D for MPI mode, include –enable-mpi as a configure command line option. To install Titan2D for Hybrid mode (MPI and OpenMP), include –enable-openmp and –enable-mpi as configure command line options. Installing Titan2D for OpenMP mode is preferable at this time. Please see notes in INSTALL for more information.

The Titan2D GUI requires the installation of Java 1.7 or greater. Please see notes in the README.GUI file for more information.

After Titan2D is installed, the Titan2D GUI is invoked by entering <path to root of titan2d>/bin/titan_gui.sh.

Please verify titan_gui.sh environment variable settings before invoking titan_gui.sh for the first time.

For convenience, you can add the location of binaries to the PATH environment variable in your .bashrc file, export PATH=<path to root of titan2d>/bin:$PATH

## Running Titan2D

On VHub, to start the Titan2D GUI, go to [https://vhub.org/resources/titan2d], and click on the Launch Tool button.

On other 64-bit Linux platforms, to start the Titan2D GUI, go to the <path to root of titan2d>/bin directory and enter titan_gui.sh.

To run Titan2D as a standalone application in OpenMP mode (preferred at this time), go to the <path to root of titan2d>/bin directory and enter ./titan [-nt <number of threads>] <python input script> where <python input script> is the name of a Titan2D python input file.

To run Titan2D as a standalone application in MPI mode, go to the <path to root of titan2d>/bin directory and enter mpirun -np <number of processes> ./titan <python input script>.

To run Titan2D as a standalone application in hybrid MPI/OpenMP mode, go to the <path to root of titan2d>/bin directory and enter mpirun -np <number of processes> ./titan [-nt <number of threads>] <python input script>.

Before running Titan2D as a standalone application, it is necessary to set the TITAN_HOME

and LD_LIBRARY_PATH environment variables. To do this, go to the <path to root of titan2d>/bin directory and enter source titanvars.sh.

When running Titan2D via the Titan2D GUI, Titan2D python input files are generated automatically when jobs are submitted via the Job Submission Tab. For advanced users, Titan2D python input files can be created using a text editor. Please see section Titan2D Python Input Files for more information on the format of Titan2D python input files.

When running Titan2D as a standalone application, runtime information is printed to the screen at the completion of each iteration. This information consists of:

1. Time at the end of each timestep in hrs:min:sec
2. Volume of the flow
3. Maximum pile height [m]
4. Average Velocity [m/s]
5. Total number of elements
6. Minimum and maximum X and Y bounds

When the execution is complete, runtime performance and profiling information is displayed.

When running Titan2D via the Titan2D GUI, the runtime information, including performance and profiling information, is returned in Stdout.txt.

If a Titan2D error is encountered, detailed explanations of the error are returned in Stderr.txt.

## Restarting Titan2D

The Titan2D GUI provides support for restarting Titan2D. Please see more information on how to do this via the GUI, in the Run Parameters Tab and Output Options Tab sections.

Here are brief instructions on how to restart a Titan2D simulation when running Titan2D as a standalone application.

To restart Titan2D, you must first save Titan2D restart files by calling the sim.setRestartOutput method in the Titan2D python input file. The restart files are saved to the restart directory specified by the output_prefix argument to the sim.setRestartOutput method. Please see more information on how to do this in the sim.setRestartOutput section. Saved Restart Files have the following naming convention: snapshot_p[MPI Process ID]_i[step].h5, For example: snapshot_p0000_i00003401.h5.

The Titan2D restart command line interface is:

titan [-nt <number of threads>] -restart [-add-iter <iter>] [-add-time <time>] <path to restart snapshot>

For example, consider a case when the previous run of Titan2D runs out of walltime; to continue the simulation enter:

```
titan -nt 4 -restart restart/snapshot_p0000_i00003401.h5
```

As another example, consider the case when a previous simulation finished successfully and it reached requested iteration number but you need more run time, use -add-iter and/or -add-time option to add extra time/iteration. For example, to ask for 10000 more iterations and 3600 more seconds of simulation time, enter:

```
titan -nt 4 -restart -add-iter 10000 -add-time 3600
restart/snapshot_p0000_i00003401.h5
```

## Examples for Running Titan2D

Examples for running Titan2D are contained in the <path to root of titan2d>/share/titan2d_examples folder.

# 4 GIS GRASS and GDAL Interfaces

Titan2D performs flow simulations on a DEM of a user-defined region. The DEM data file, containing elevation data, can be formatted to operate in a GRASS GIS (Geographic Resources Analysis Support System – Geographic Information System) environment or formatted to work in a GDAL (Geospatial Data Abstraction Library) environment.

GRASS GIS is an open source GIS with raster, topological vector, image processing, and graphics production functionality. It operates on various platforms through a graphical user interface. Note: Titan2D runs independently of GRASS. That is, GRASS is NOT REQUIRED to run Titan2D. GRASS GIS is available free of charge under GNU General Public License (GPL) [https://grass.osgeo.org].

GDAL is a translation library for raster and geospatial data formats. GDAL is available free of charge under the X/MIT style Open Source license by the Open Source Geospatial Foundation [http://www.gdal.org]. Please see the GDAL Raster Formats webpage [http://www.gdal.org/formats_list.html] for a list of raster formats supported by GDAL.

Simulation accuracy is highly dependent on the level of DEM resolution and quality. DEMs with higher resolutions (e.g. 5-30m) render more accurate representations of actual geophysical flow events, especially in situations where channelized flow is involved.

The Titan2D GRASS GIS and GDAL interfaces allow the user to adjust the area of the desired initial DEM. This capability decreases output file size thus increasing visualization speed and allows the user to focus attention upon a specific region of interest within a much larger DEM.

For the examples presented in this document, a DEM of the Colima Volcano in Mexico is used and displayed in Figure 4-1 below. WG284 Coordinates: 19°30’45.82”N, 103 37’2.07”W. UTM Coordinates: Zone 13, 645095 2158146. On VHub, this DEM is available in the /apps/titan2d/dev/examples/Colima/grass.data/grass5 directory. This DEM is also available on GitHub:[https://github.com/TITAN2D/DEM].

Figure 4-1 Colima Volcano

# 5 Titan2D Graphical User Interface (GUI)

The Titan2D Graphical User Interface (GUI) provides users with menus and tabulated forms to setup, save and load GIS data specifications, Titan2D computational and run parameters, material properties, flux sources parameters, and discharge planes parameters, and, to submit and monitor Titan2D jobs. The GUI comprises three menus and ten tabulated forms. To select a menu, left click on the desired menu in the upper left corner of the GUI screen and select an item from the drop down menu that appears. To select a form, left click on the desired form's tab at the top of the GUI screen.

The Titan2D GUI translates the parameters entered into the forms and creates Titan2D python input files for running Titan2D jobs.

Figure 5-1 shows the initial screen presented to the user when the Titan2D GUI is launched. As shown, the Load/Save Tab is the first tabulated form displayed. In this section, the menus and tabulated forms are described in detail.

## Options Menu

### Titan Application Version

Set the Titan application binary files to use when launching a Titan model run. The setting allows the user flexibility to use Titan application binary files that may have been modified by the user for testing purposes.

### Use External Titan Binaries

When set to False, the Titan application binary files specified by the user's PATH environment variable will be used when a Titan run is launched from this interface.

When set to True, the Titan application binary files at the location specified by the titan binary directory entered on this panel will be used when a Titan run is launched from this interface.

### Titan Binary Directory

The fully qualified path where the Titan application binary files are located can be entered in this field. The field is not required if the use of external binaries is disabled. The value can be set by using the directory chooser, which is displayed by pushing the "Titan Binary Directory" button, or entering the directory in the text field.

## Help Menu

## Help Me

The Help Menu provides information for running Titan2D.

# Load/Save Tab

Figure 5-1 displays the Load/Save Tab. The Load/Save tab helps you to load a set of previously saved Titan2D input parameters, or save the current set, which can save you a lot of time/effort for frequently used set of run parameters.

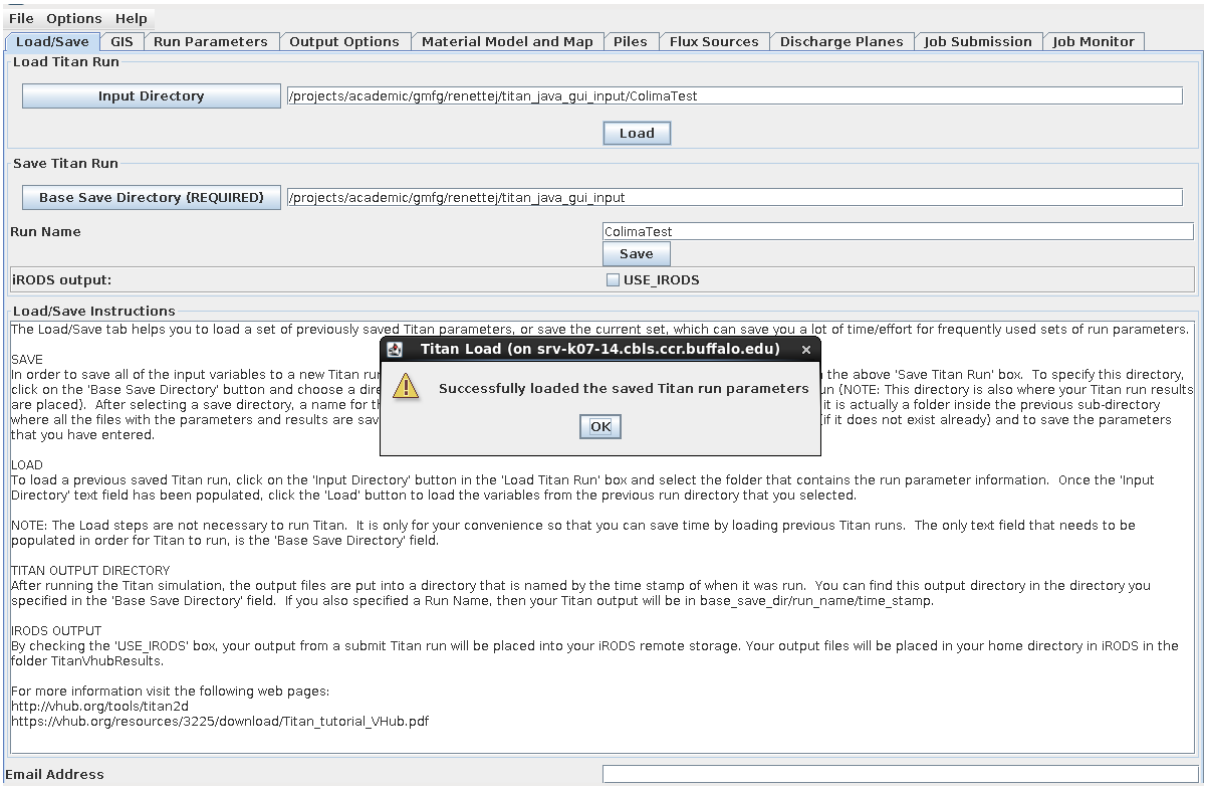


Figure 5-1 Load / Save Tab

## Load Titan Run

(this step is not necessary) To load a previous run, you need to enter the full path name of a previously saved run in the field next to Input Directory. You can also click on the Input Directory button and navigate to that directory within a folder browser that pops up.

## Save Titan Run

Here you have to specify the path (directory) where you want to store your new run. On VHub, the directory name must begin with /home/vhub/yourusername where yourusername corresponds to your personal VHub username (shown at the very top of the web page). You can

also click on the Base Save Directory button and navigate to the folder where you save your Titan2D runs. In order to save all of the input variables for a new Titan2D run, you need to specify a save directory.

### Run Name

The Run Name is actually a folder inside the previous directory where all the files with the parameters and results are saved. You need to press the Save button AFTER entering all other input parameters and just BEFORE submitting the job. The tool creates the folder (if it doesn't exist already) and save the parameters that you have entered in a series of files. To run the simulations, the tool reads the input parameters from these files (not from the GUI).

On, VHub, following the example of the figure, the directory where the input files are saved would be /home/vhub/yourusername/anyname.

NOTE: After running a Titan2D simulation, the output files are put into a directory that is named by the time stamp of when it was run. You can find this output directory inside the directory you specified in the field above, that is /home/vhub/yourusername/anyname.

## GIS Tab

Figure 5-2 displays the GIS Tab. The GIS tab enables users to specify GIS data information as detailed below.

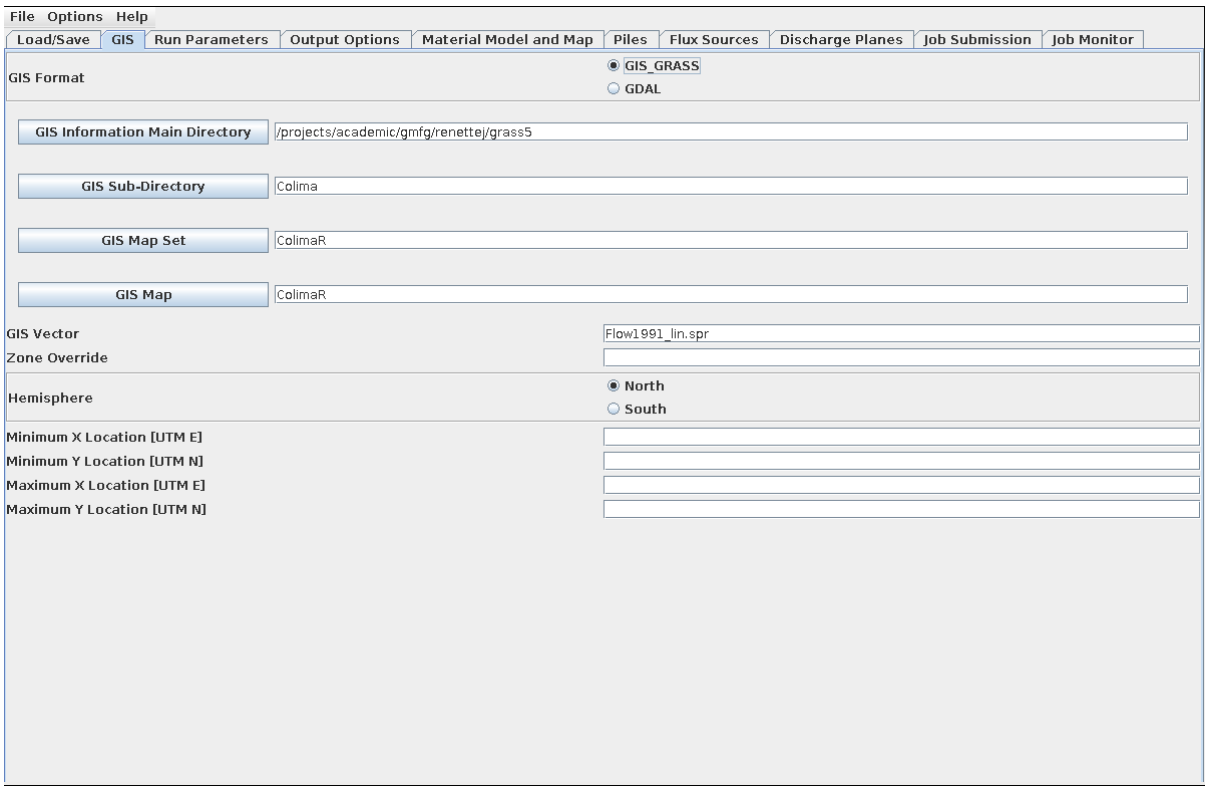


Figure 5-2 GIS Tab

## GIS Format

The value "GIS_GRASS" or "GDAL" can be selected in this field to select the GIS Format. When "GIS_GRASS" is selected, the GIS Information Main Directory, GIS Sub-Directory, GIS Map Set, and GIS Map fields are required; please see details below. When "GDAL" is selected, only the GIS Map field is required; please see details below.

## GIS Format GIS_GRASS

## GIS Information Main Directory

The main directory where the GIS information is stored can be entered in this field. All GIS input data must be in GRASS format to be used. The value can be set by using the directory chooser, which is displayed by pushing the "GIS Information Main Directory" button, or entering the directory in the text field.

## GIS Sub-Directory

The sub-directory where the GIS information is stored can be entered in this field. The value can be set by using the directory chooser, which is displayed by pushing the "GIS Sub-Directory" button, or entering the directory in the text field.

## GIS Map Set

The name of the GIS map set can be entered in this field. The value can be set by using the directory chooser, which is displayed by pushing the "GIS Map Set" button, or entering the directory in the text field.

## GIS Map

The name of the GIS raster map can be entered in this field. The value can be set by using the directory chooser, which is displayed by pushing the "GIS Map" button, or entering the directory in the text field.

## GIS Vector

The name of the GIS vector can be entered in this field. This field may be left blank. The “GIS vector” field allows you to enter the outline of an actual event in a SPRING GIS format [http://www.dpi.inpe.br/spring/english/index.html/]

## GIS Format GDAL

## GIS Map

The full path name of the GIS raster map. The value can be set by using the directory chooser, which is displayed by pushing the "GIS Map" button, or entering the directory in the text field.

## Zone Override

An integer value can be entered in this field to override the zone specified in the GIS Map file. The zone is required for creating KML files. Please see the Job Monitor tab for more information on creating KML files. Leave this field blank to use the zone specified in the GIS Map file for creating KML files.

## Hemisphere

The value "North" or "South" can be selected in this field to set the hemisphere for the GIS Map. The hemisphere is required for creating KML files. Please see the Job Monitor tab for more information on creating KML files.

## Minimum X Location

A decimal value, in UTM E coordinates, can be entered in this text field. If a computational region that is smaller than the GIS region is desired, the user can input the minimum X location of the desired computational region. The field can be left blank indicating to use the GIS region.

### Minimum Y Location

A decimal value, in UTM N coordinates, can be entered in this text field. If a computational region that is smaller than the GIS region is desired, the user can input the minimum Y location of the desired computational region. The field can be left blank indicating to use the GIS region.

### Maximum X Location

A decimal value, in UTM E coordinates, can be entered in this text field. If a computational region that is smaller than the GIS region is desired, the user can input the maximum X location of the desired computational region. The field can be left blank indicating to use the GIS region.

### Maximum Y Location

A decimal value, in UTM N coordinates, can be entered in this text field. If a computational region that is smaller than the GIS region is desired, the user can input the maximum Y location of the desired computational region. The field can be left blank indicating to use the GIS region.

*Note on entering coordinates: The user can look up the boundaries of a particular map by looking in its corresponding header folder. For example, if using the DEM "ColimaSmall", the map boundaries are located in the file by that name in the following directory: /Colima/ColimaSmall/cellhd. (Coordinates are also required for specifying the locations of pile/flux-source centers and the end-points of discharge planes.)

Please see Appendix C for more GIS information.

## Run Parameters Tab

Figure 5-3 displays the Run Parameters Tab. The Run Parameters tab enables users to specify computational parameters.

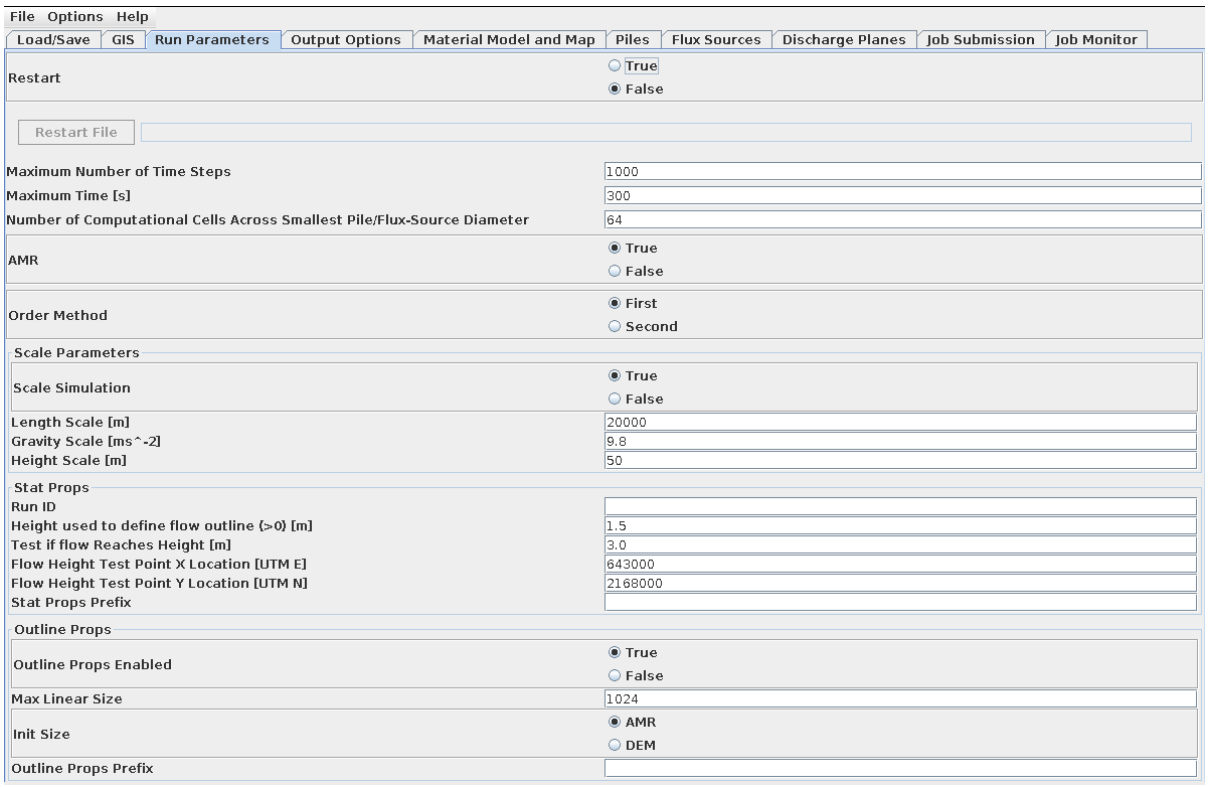


Figure 5-3 Run Parameters Tab

## Restart

A boolean value can be selected in this field. Activate the "True" button to enable running Titan by passing Titan a Restart File as an argument. When a Restart File is passed as an argument to Titan, the simulation will restart at the time specified in the selected Restart File, using parameters specified in the selected Restart File. When Restart is enabled, updated parameter values, accept for the Additional Maximum Number of Time Steps and Additional Maximum Time parameters, will not have an effect. Please see the Output Options tab for more information on creating Restart Files.

## Restart File

When Restart is enabled, a Restart File filename must be selected via this Directory Selector field. Saved Restart Files have the following naming convention: snapshot_p[MPI Process ID]_i[step].h5. For example: snapshot_p0000_i00003401.h5.

## Maximum Number of Time Steps/Maximum Time

The next two input parameters concern the run time of the simulation. The user must specify the Maximum Number of Time Steps (on the order of several thousand) and Maximum Time (in seconds). Fractions of seconds may be used if desired (e.g. 2.5 seconds). When the job is submitted for processing, the simulation will stop when it has reached the specified maximum allowed number of time steps or has simulated the specified amount of time (whichever comes first).

Note: Determining an acceptable stopping criterion for the simulation is difficult when considering general cases. Because of this, the stopping criteria for the simulation must be set by the user. The two criteria that are used are Maximum Time and Maximum Number of Time Steps. Both of these values should be set high enough such that the geologic event being simulated has ended (e.g. the material has come to rest). If either of these values is set too low, the simulation will end before the material has reached static equilibrium. If both of these values are set excessively high, wasted computation will be performed dynamically simulating a pile that has already come to rest. The simulation will end when either the number of time steps computed is equal to the Maximum Number of Time Steps or the simulation time has reached the Maximum Time. The number of computed time steps needed to simulate a geologic event will vary depending on the amount of computational mesh points used, the friction parameters, the use of grid adaptation, the simulation order, and initial pile geometry and location.

## Maximum Number of Time Steps

An integer value can be entered in this text field to set the maximum number of time steps that the simulation will run. For most simulations, this should be in the 1,000s range.

## Maximum Time

A decimal value, in seconds, can be entered in this text field to set the maximum amount of time that the simulation will run.

## Additional Maximum Number of Time Steps

When Restart is enabled, an integer value can be entered in this text field to set the additional maximum number of time steps that the restart simulation will run.

## Additional Maximum Time

When Restart is enabled, a decimal value, in seconds, can be entered in this text field to set the additional maximum amount of time that the restart simulation will run.

## Number of Computational Cells Across Smallest Pile/Flux-Source Diameter

TITAN2D creates a regular grid/mesh on which the computation takes place. The amount of cell refinement within Titan is determined by the smallest pile/flux source diameter and the number of cells the user wants across it. In other words, this value is used by Titan to determine the maximum level of grid-cell refinement allowed throughout the simulation. The default case as an example (a value of 20) can be used. Then, at the beginning of a simulation, the grid-cell containing the centroid of each pile is successively refined (i.e. split into 4 ‘son’ cells) until its size is no larger than 1/20th of the smallest pile or flux source diameter. This is the size of the

smallest computational cell allowed on the map during the course of the simulation. Although (with an adaptive grid) maximally-refined cells will not occur everywhere, but rather only where they are needed (only in close proximity to a pile's boundaries or where ever a flow's mass/momentum fluxes become large). It is important to choose a value here that strikes a balance between computation time and calculation accuracy. If there are many debris piles, they will all have maximally refined cells around their edges throughout the simulation. If the size of the maximally refined cells is very small, then there will be more of them and more computation time will be required to process the data along with more disk space to store it. Please see Figure 5-4 for a basic illustration of this process. Note, in figure 5-4, the minor axis of the ellipse is the smallest pile diameter on the map and is therefore chosen as the diameter that is used in determination of the maximum level of refinement.

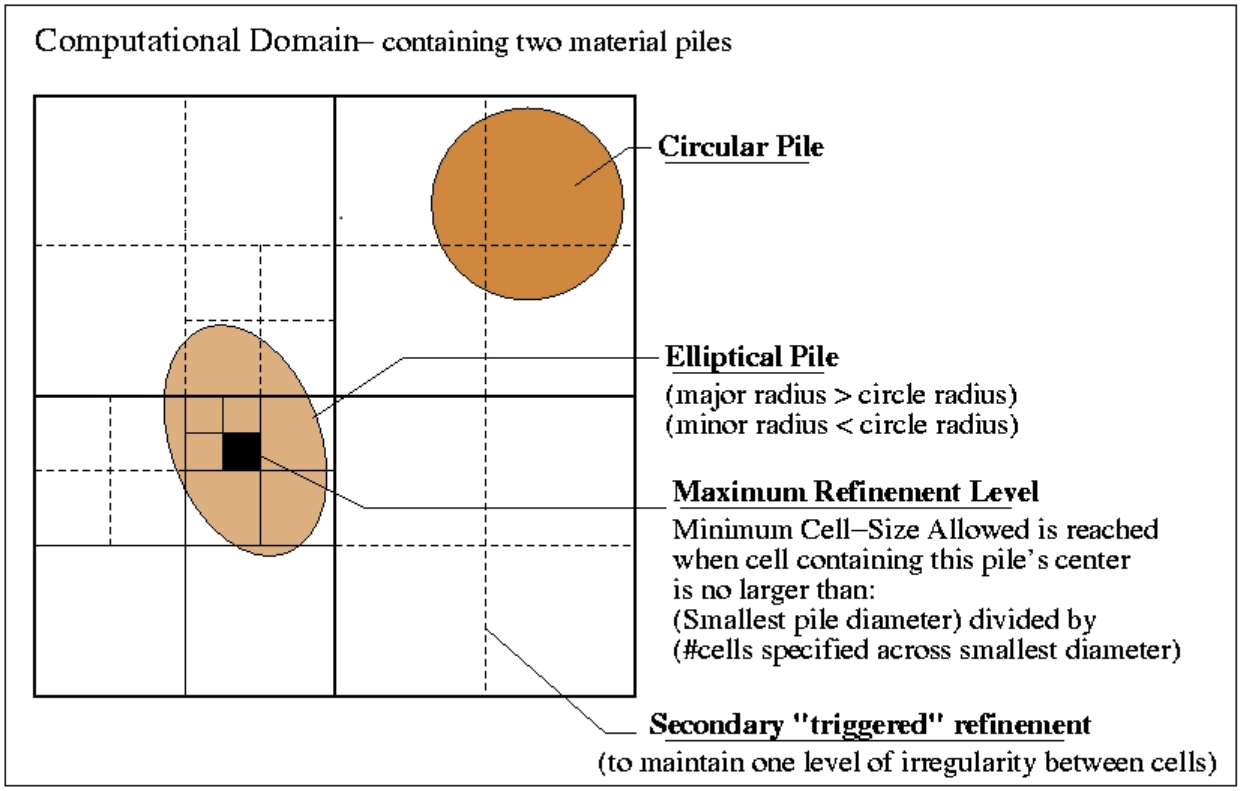


Figure 5-4 Computational Domain

## AMR (Adaptive Mesh Refinement)

This refers to the computational grid and reduces the computational cost while maintaining the simulation's accuracy. However, this can also introduce some instability into the computation. Also note that if AMR is not selected, then the entire computational domain will be a uniform maximally refined grid with a cell-size determined by the user's specification in the Number of Computational Cells Across Smallest Pile/Flux-Source Diameter box above. Due to the major savings in computation time, it is recommended that this be selected unless instability is detected in the output.

## Order Method

Clicking on the Second button for this option allows you to select the 2nd order method for calculating the values in a computational grid cell (the 1st order method is the default). Under the first order method, the values for pile height, momentum etc. that are calculated by the model are approximated as constant across the entire cell. This may mean that there is a jump up or down to the value of the same parameter in the neighboring cell. Under the 2nd order method, the values of the parameters are assumed to vary linearly across the cell. The 2nd order method takes into account the value of the neighboring cells and uses those values to calculate the value for the cell in question. For example, if the neighboring cells up-flow of the cell in question have values lower than the cells down-flow of the cell in question, the value of the cell is going to increase (have a slope) in the down-flow direction. If there is no difference in values between the neighboring cells, then the cell in question will also stay constant. Selecting the 2nd order method will produce slightly more accurate results, but may also increase the computation time as the code has to perform more calculations.

## Scale Parameters

## Scale Simulation

Titan2D allows for several properties of the simulation to be scaled. Clicking the 'True' button allows the governing equations to be non-dimensional using pile height, gravity and length scaling factors (among others, such as velocity and time scales, that are secondarily derived from them). By default, the height scale used by Titan2D is the cube root of the total volume from all piles and flux sources. It is calculated at the beginning of the simulation and then is a constant from that point on. The gravity scaling factor is simply taken as 9.80 $ms^{-2}$.

***Note:** Scaling will help reduce the occurrence of round-off error and is critical for the proper control of propagating thin layers (i.e. the boundaries of a flow), which otherwise may have a tendency to develop unrealistically high velocities.

## Length Scale

A decimal value, in meters, can be entered in this text field. This scale factor refers to the expected runout length of the flows and need only be specified if the simulation is to be scaled, however for all real-terrain calculations it is highly recommended that the simulation be scaled. Note: Scaling will help reduce the occurrence of round-off error and is critical for the proper control of propagating thin layers (i.e. the boundaries of a flow), which otherwise may have a tendency to develop unrealistically high velocities. If the simulation is scaled, this field cannot be left blank.

## Gravity Scale

A decimal value, in meters per second squared, can be entered in this text field. The default gravity scale is 9.80 $ms^{-2}$. If the simulation is scaled, this field cannot be left blank.

## Height Scale

A decimal value, in meters, can be entered in this text field. When this field is left blank, Height Scale will be calculated based on the total volume of moving material.

## Stat Props

## Run ID

This provides the user with greater control over the extension naming of the statistical output files. If left blank, the files are named with a .-00001 extension.

## Height used to define flow outline (> 0)

This box sets the "boundary" of the mass flow. The user can define a pile boundary (i.e. ignore material less than a specified pile height) by entering a value greater than zero here. Although thin-layer flow is now controlled, the user is still able to define a pile boundary (i.e. ignore material less than a specified pile height) by entering a value greater than zero here. This will be used to, among other things, compute the spread of the pile in the x and y directions. Spread in each direction is defined as the maximum minus minimum coordinate where the pile height is greater than the value entered here. If no value is entered it defaults to 1/50th of the maximum initial pile height.

## Test if flow reaches height [m] ...: and ... at test point (x and y location)

These options set the criteria to determine if the flow reaches a particular height or location, namely did flow of this depth reach this location at any time during the calculation.

## Output Prefix

This provides the user with greater control over the extension naming of the statistical output files.

## Outline Props

### Outline Props Enabled

A Boolean value can be selected in this field. Activate "True" to enable outline properties calculations. Outline properties include the calculation and output of pileheightrecord.%06d files, where .%06d is the Stat Props' Run ID number left padded by zeros. These files contain spatially varying maps of maximum over (simulated) time flow depth at each East-North location on a uniform rectangular grid.

### Max Linear Size

An integer value can be entered in this field to set the maximum size of the outline grid in the X and Y directions.

### Init Size

The value "AMR" or "DEM" can be selected in this field to select the method for calculating the initial outline grid size. When "AMR" is selected, Titan uses the highest possible resolution. When "DEM" is selected, Titan uses the DEM resolution. In both cases, the outline grid size is decreased until the sizes in the X and Y directions are less than Max Linear Size.

### Outline Props Prefix

A string value can be entered in this text field to select the prefix for outline output files.

## Output Options Tab

Figure 5-5 displays the Output Options Tab. The Output Options Tab enables users to specify the format of and when output files are generated by Titan2D.

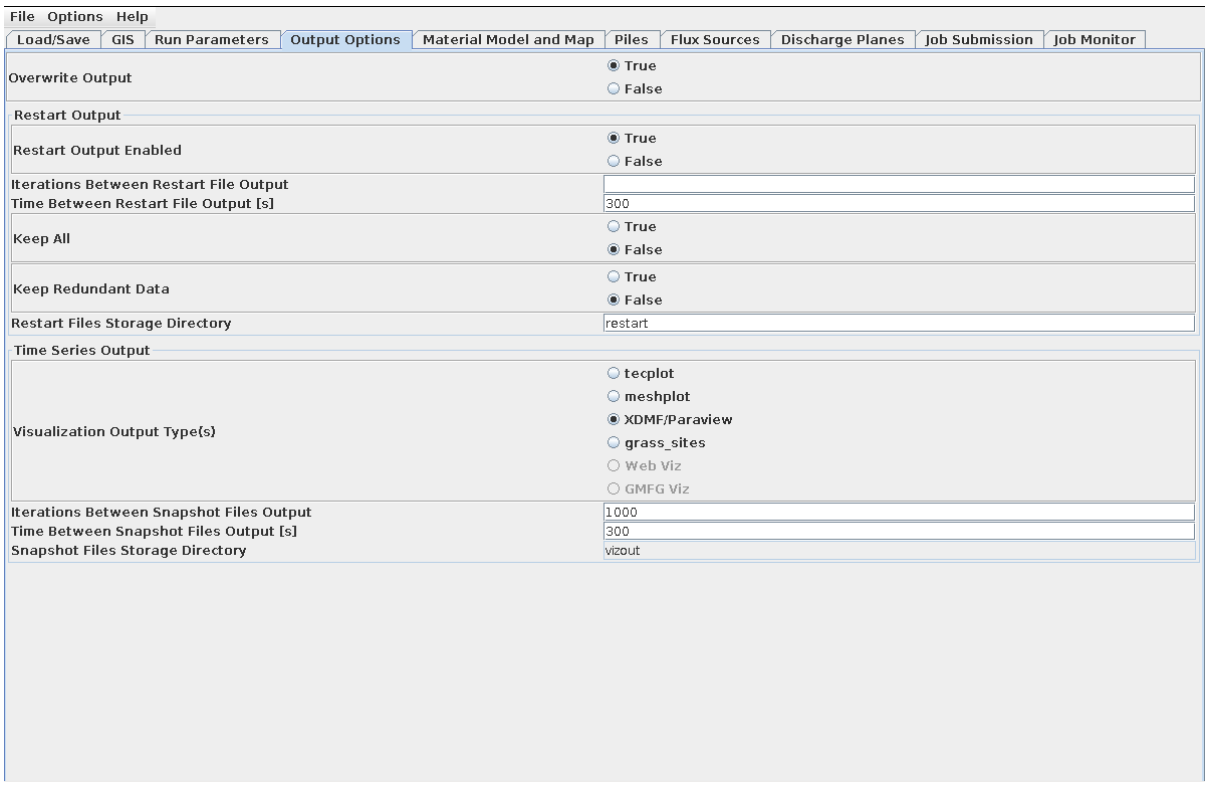


Figure 5-5 Output Options Tab

## Overwrite Output

A boolean value can be selected in this field to enable/disable the results overwrite safety flag. If set to True, overwrite previous output. If set to False, abort simulation if output files with same names exist in the current working directory.

## Restart Output

Define when to save Restart Files required for running Titan when Restart is enabled via the Run Parameters tab.

## Restart Output Enabled

A boolean value can be selected in this field. If set to True, Restart File output is enabled.

## Iterations Between Restart File Output

An integer value can be entered in this text field. This corresponds to how often Restart Files will be saved based on the number of iterations between saves. Dump a Restart File every Iterations Between Restart File Output iterations. These files can become very large and the user may not need to see results for every iteration since some iterations may have little change in the results from the previous iteration. For example, a user may wish to only save a Restart File every 100 iterations.

## Time Between Restart File Output

A decimal value, in seconds, can be entered in this text field. This corresponds to how often Restart Files will be saved based on the time between saves. Dump a Restart File every Time Between Restart File Output seconds. These files can become very large and the user may not need to see results for every time step since some time steps may have little change in the results from the previous time step. For example, a user may wish to only save a Restart File every 300 seconds of simulated time. If desired, the user may choose to use a fraction of a second.

## Keep All

A boolean value can be selected in this field. If set to True, keep all Restart Files. If set to False, keep only the last Restart File.

## Keep Redundant Data

A boolean value can be selected in this field. If set to True, keep redundant data.

## Restart Files Storage Directory

Subdirectory name for storing the Restart Files. The saved Restart Files have the following naming convention: snapshot_p[MPI Process ID]_i[step].h5. For example: snapshot_p0000_i00003401.h5.

## Time Series Output

Define when to save Visualization Output Snapshot Files.

## Visualization Output Type(s)

Formats for Visualization Output Snapshot Files. Choose formats (tecplot, meshplot, XDMF/Paraview, grass_sites, Web Viz and/or GMFG Viz) - Activate the buttons corresponding to the visualization outputs that are desired.

The tecplot (tecpl[step].tec) and meshplot (mshpl[step].tec) formats are tecplot files. The eXtensible Data Model and Format (XDMF)/Paraview(xdmf_p[MPI Process ID]_i[step].h5) is for Graphics Interchange Format (GIF), Paraview and Google Earth visualizations. The Web Viz format is for the web visualization output. The GMFG Viz format is for the TITAN2D visualizer. Note that the user can have no or multiple visualization output formats with each simulation run.Note: To use the Titan2D Viewer tool to see your results within VHub, you need to choose the tecplot format for saving your results..

## Iterations Between Snapshot Files Output

An integer value can be entered in this text field. This corresponds to how often results will be

saved based on the number of iterations between saves. Write snapshot files every Iterations Between Snapshot File Output iterations. These files can become very large and the user may not need to see results for every iteration since some iterations may have little change in the results from the previous iteration. For example, a user may wish to only save the snapshot files every 100 iterations.

### Time Between Snapshot Files Output

A decimal value, in seconds, can be entered in this text field. This corresponds to how often results will be saved based on the time between saves. Write snapshot files every Time Between Snapshot File Output seconds. These files can become very large and the user may not need to see results for every time step since some time steps may have little change in the results from the previous time step. For example, a user may wish to only save the snapshot files every 300 seconds of simulated time. If desired, the user may choose to use a fraction of a second.

### Snapshot Files Storage  Directory

Subdirectory name for storing the Visualization Output Snapshot Files.

## Material Model and Map Tab

Figure 5.6 displays the Material Model and Map Tab. The Material Model and Map Tab enables users to specify material properties.

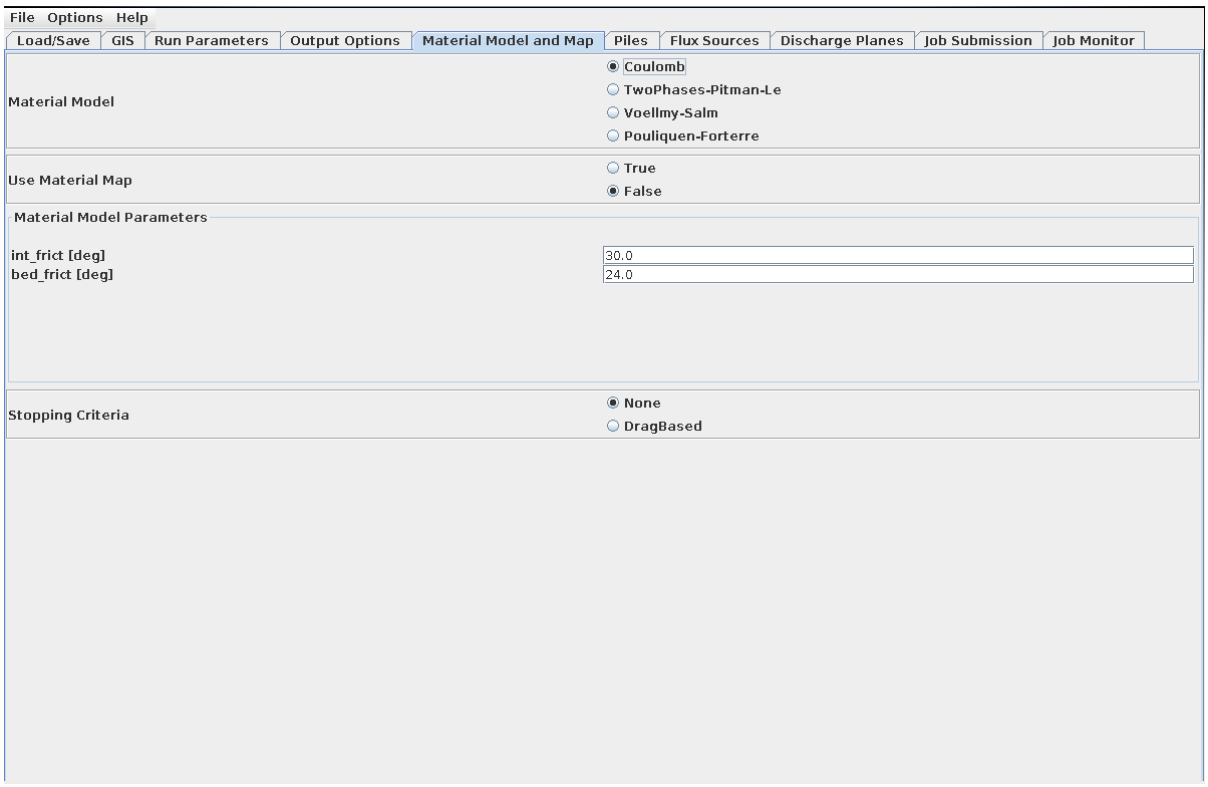


Figure 5-6 Material Model and Map Tab

## Material Model

Select the material model governing the flow of material for the simulation. Current models available are Coulomb, TwoPhases-Pitman-Le, Voellmy-Salm and Pouliquen-Forterre. When a material model is selected, text fields appear which allow model specific input parameters to be entered. Please see Appendix B for more information specific to the Voellmy-Salm and Pouliquen-Forterre material models.

The following briefly describes the material models and their respective input parameters:

## Coulomb

For the Coulomb model, the granular material is assumed to be an incompressible continuum satisfying a Mohr Coulomb law. Material model parameters:

### int_frict

A decimal value, in degrees, can be entered in this text field to set the internal friction angle. This field cannot be left blank.

### bed_frict

A decimal value, in degrees, can be entered in this text field to set the bed friction angle. This field cannot be left blank.

The internal friction angle ϕint, is the steepest angle that the upper surface of a conical pile of dry sand can make with respect to the horizontal plane it is resting on.

The bed (also known as basal) friction angle, ϕbed, is the angle that a plane needs to be inclined so that a block of material will slide downslope at a constant speed. Please figure 5-7 for a basic illustration of this.

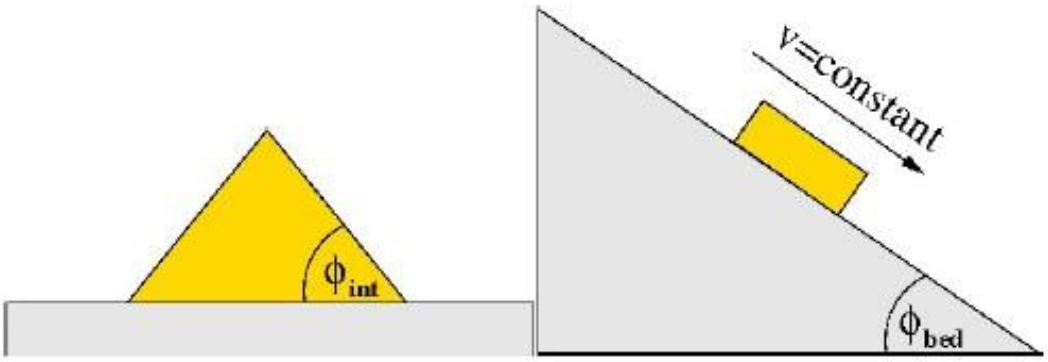


Figure 5-7 Friction Angles

A complete description for this material model, including the model's governing equations, is provided in the following reference papers:

A. K. Patra, C. C. Nichita, A. C. Bauer, E. B. Pitman, M. I. Bursik, M. F. Sheridan. "Parallel adaptive numerical simulation of dry avalanches over natural terrain." Journal of Volcanology and Geothermal Research, 139:1-21 (2005).

E. B. Pitman, C.C. Nichita, A.K. Patra, A.C. Baur, M. Bursik, A. Weber. "Computing granular avalanches and landslides." Physics of Fluids 15:36-38, (2003).

## TwoPhases-Pitman-Le

For the TwoPhases-Pitman-Le model, the granular flow is assumed to consist of a solid phase and a fluid phase. Material model parameters:

### int_frict

A decimal value, in degrees, can be entered in this text field to set the internal friction angle. This field cannot be left blank.

### bed_frict

A decimal value, in degrees, can be entered in this text field to set the bed friction angle. This field cannot be left blank.

## Volume Fraction

The solid phase requires a volume fraction parameter for each pile defined via the Piles tab; set each pile's Volume Fraction field via the Piles tab.

A complete description for this material model, including the model's governing equations, is provided in the following reference paper:

E. B. Pitman, Long Le. "A two-fluid model for avalanche and debris flow." Philosophical Transactions of the Royal Society A, 363:1573-1601, (2005).

## Voellmy-Salm

For the Voellmy-Salm model, the total basal friction is split into a velocity independent dry-Coulomb term, mu, which is proportional to the normal stress at the flow bottom, and, a velocity dependent "viscous" or "turbulent" friction term, xi.

Users are encouraged to apply this material model for snow avalanche simulations. Material model parameters:

### mu

A decimal value, in degrees, can be entered in this text field to set the velocity independent term. This field cannot be left blank.

### xi

A decimal value, in degrees, can be entered in this text field to set the velocity dependent term. This field cannot be left blank.

A complete description for this material model, including the model's governing equations, is provided in the following reference paper:

M. Christen, J. Kowalski, P. Bartelt. "RAMMS: Numerical simulation of dense snow avalanches in three-dimensional terrain." Cold Regions Science and Technology 63:1-14, (2010).

## Pouliquen-Forterre

For the Pouliquen-Forterre model, the basal friction coefficient is computed based on a friction law described in the referenced Pouliquen-Forterre paper. Material model parameters:

### phi1

A decimal value, in degrees, can be entered in this text field. This field cannot be left blank.

## phi2

A decimal value, in degrees, can be entered in this text field. This field cannot be left blank.

## phi3

A decimal value, in degrees, can be entered in this text field. This field cannot be left blank.

## Beta

A decimal value can be entered in this text field to set this constant material property. This field cannot be left blank.

## L_material

A decimal value can be entered in this text field to set this constant material property. This field cannot be left blank.

A complete description for this material model, including the model's governing equations, is provided in the following reference papers:

Pouliquen O., Forterre Y. "Friction law for dense granular flows: application to the motion of a mass down a rough inclined plane." Journal of Fluid Mechanics 453:133-151, (2002).

Forterre Y., Pouliquen O. "Flows of Dense Granular Media." The Annual Review of Fluid Mechanics 40:1-24

## Use Material Map

NOTE: This is available only for users who have a specific material map (comprising files ending in _Mat stored in the GIS Mapset cats, cell and cellhd directories). The “True” check box enables the input of a GIS-based surficial material map that matches the area covered by the DEM. This map will be used to define the zones in the region where changes in the surface morphology results in a change in the basal friction angle. When this function is enabled, subsequent text boxes to enable the selection of basal friction angles for each material represented on the material map. If left “False”, only one basal friction angle is necessary. The use of material maps is currently enabled for the Coulomb material model only.

## Stopping Criteria

A boolean value can be selected in this field to enable/disable the drag based stopping criteria. When enabled, Titan2D evaluates whether the gravitational forces in the X and Y directions are lower than the sum of two other drag forces (the drag due to internal friction and the drag due to

bed friction) or not. If so, assumes that the flow should physically stop by setting the velocities in the X and Y directions to zero. Stopping criteria is currently enabled for the Coulomb material model only.

# Piles Tab

Figure 5.8 displays the Piles Tab. The Piles Tab enables users to specify pile properties. This user-input information is used to characterize the nature of the material in the simulation

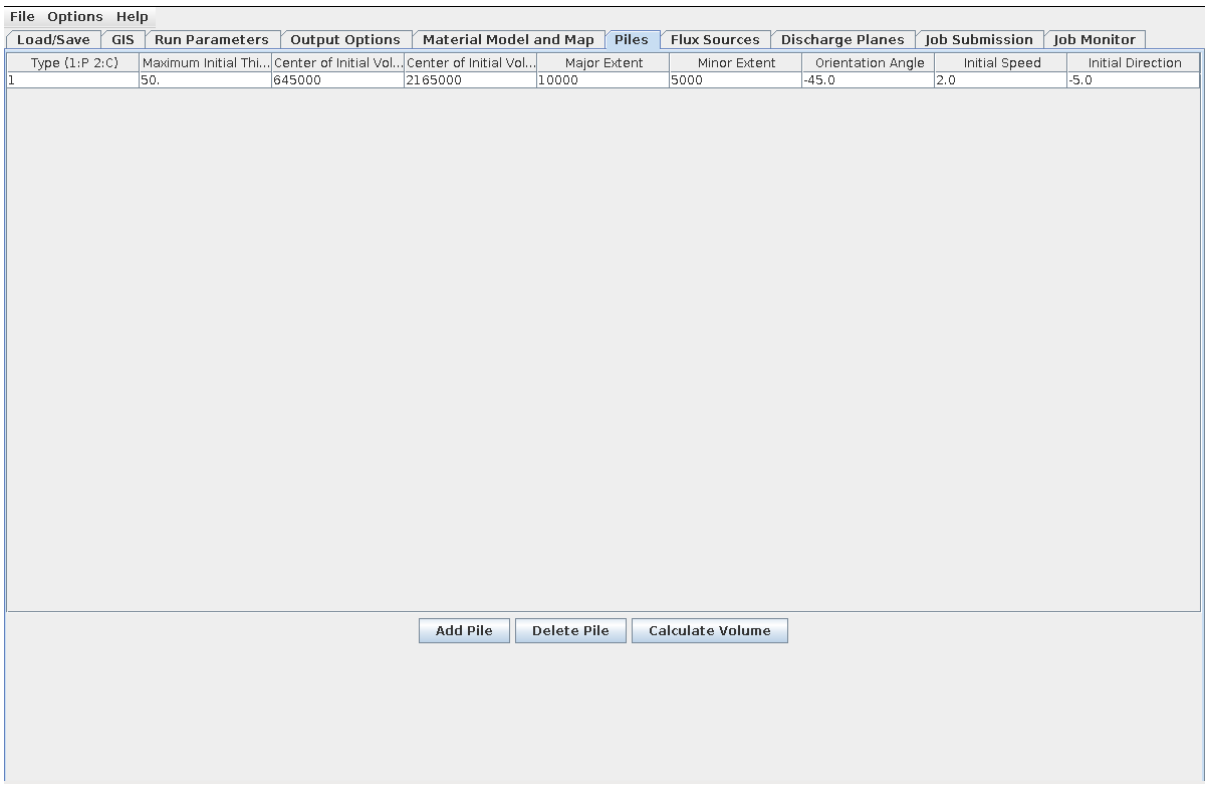


Figure 5-8 Piles Tab

Figure 5-9 displays the Titan2D piles geometry in map view:

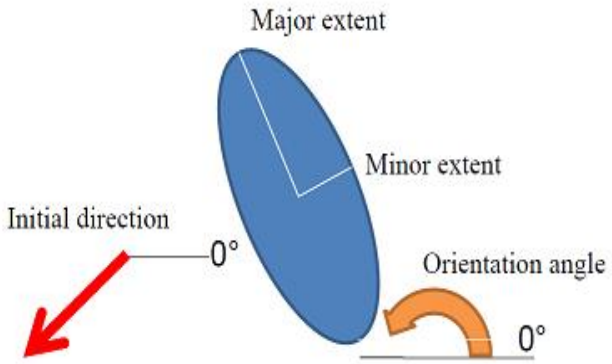


Figure 5-9 Piles Geometry

The data to be entered by the user are: The Pile Type (1: Paraboloid or 2: Cylinder), Maximum Initial Thickness (in meters), Center of Initial Volume (in UTM coordinates), the Major and Minor Extent of the initial pile (in meters), the Orientation angle (degrees from X axis to major axis), its Initial speed (m/s), its Initial direction (degrees from X axis), and, the Volume Fraction (This field only shows when the Material Model is TwoPhases-Pitman-Le). The orientation angle value must range from -180° (clockwise rotation) to +180° (counter-clockwise rotation) from east = 0°.

NOTES: The user must fill EACH box with numbers, even if the corresponding value is 0 (and press Return after typing the values in each field). Initial volumes in Titan2D are in $m^3$ and are calculated from the pile parameters like this:

Volume = 1/2*pi*radius_min*radius_max*h, which for a rmax=100, rmin=75 and h=30 gives you a volume of 353429.2 m3.

## Flux Sources Tab

The flux source shares the same paraboloid profile as a pile source, with the vertical dimension being the rate of mass per unit area (units of meters per second). As with a pile source, the major axis of the paraboloid can be aligned at an arbitrary angle in the East-North plane, and an arbitrary initial East-North velocity (tangential to the terrain) can be assigned to the mass effusing-from or raining-down-on the ground. The flux source starts at the specified value and

linearly decreases to zero at the end of the fluxes duration.

## Discharge Planes Tab

This window lets the user enter the coordinates of the two endpoints of a vertically-oriented discharge-plane. Each point is identified by a UTM E and UTM N coordinate. During the simulation Titan will calculate the amount (cubic meters) of material passing between these points and write the data in "discharge.out" files. Discharge planes have an orientation - meaning that flow passing through them in one direction may be recorded as having a positive volume, whereas flow passing through in the opposite direction will have a negative volume contribution. The orientation of discharge planes is such that if each point of 4 planes is successively laid out in a counter-clockwise manner (where the result is that the planes form a closed box), then any flow leaving the box will have a positive volume contribution. (That is, the material flux through a discharge plane obeys a right-hand rule sign convention.) See Figure 5-10 for further explanation.

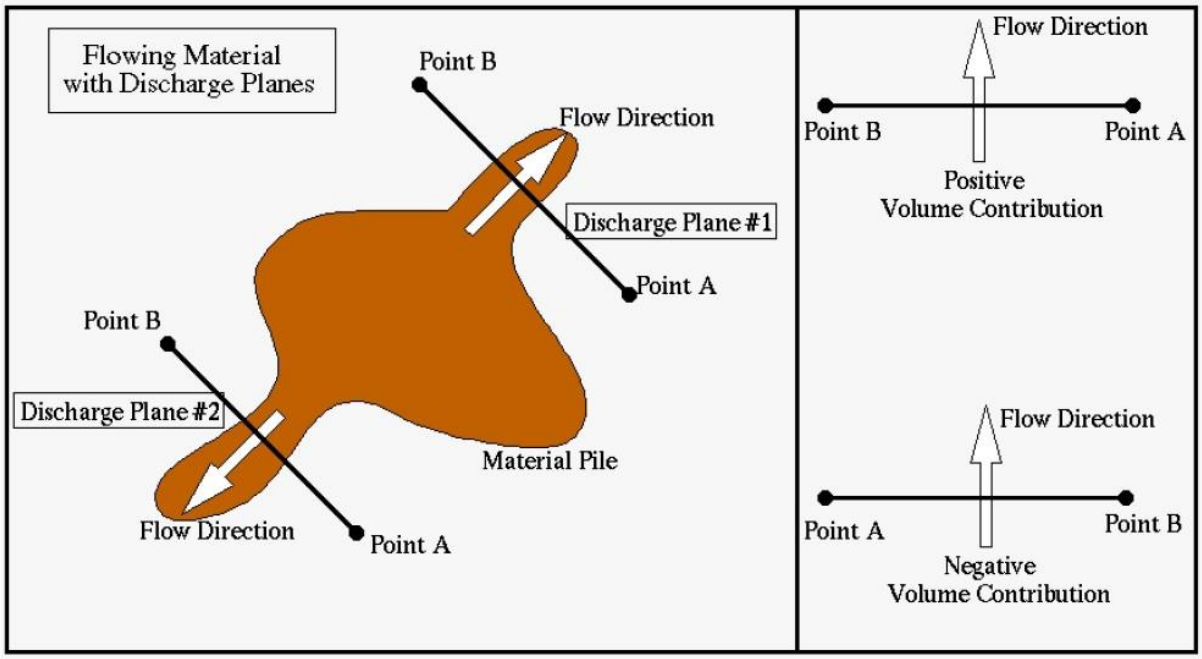


Figure 5-10 Flowing Material with Discharge Planes

## Job Submission Tab

Figure 5.11 displays Job Submission Tab.

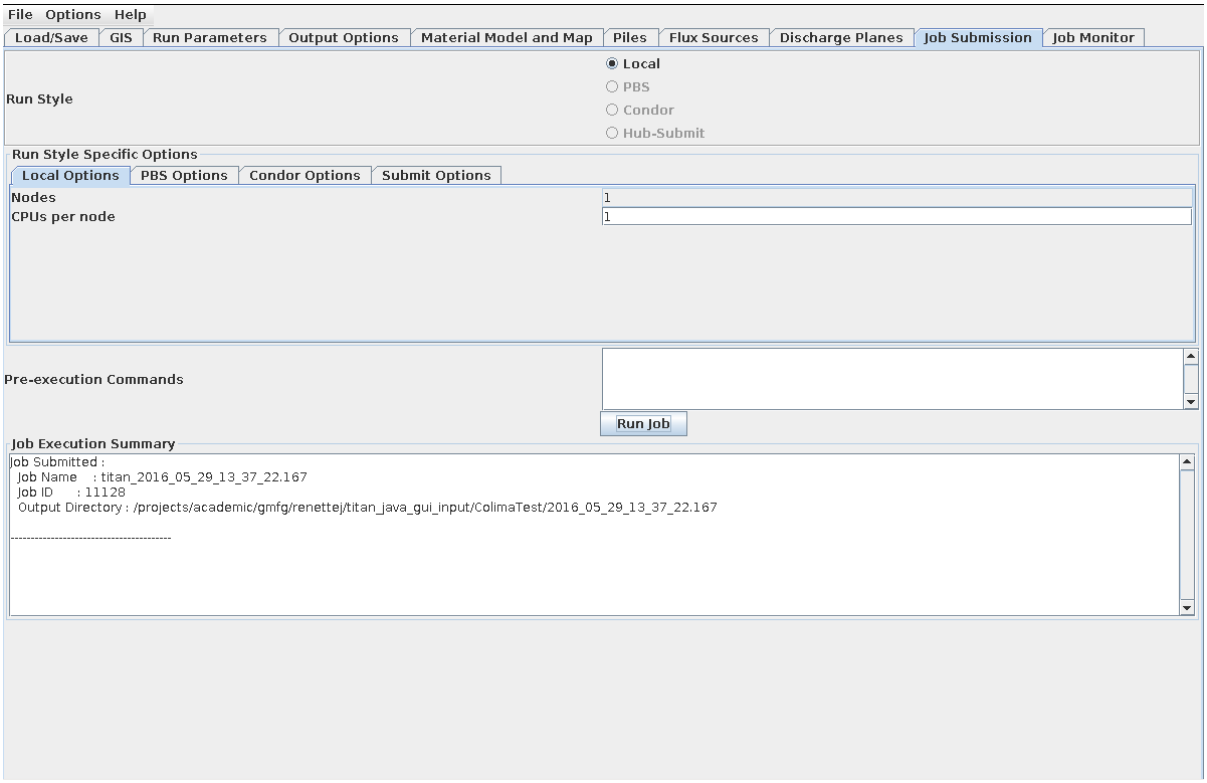


Figure 5-11 Job Submission Tab

Once the GUI tabs have been filled out for the GIS/DEM and mass flow parameters, you reach the Job Submission tab. For execution in the VHub environment you should choose the Local execution (the Hub-submit option can be used to run on a remote cluster, useful for larger jobs, but it takes longer to queue up and wait for the jobs to run) and click on the "Run Job" button. Current parameters are automatically saved when you submit the job. The stock example parameters should take less than 5 minutes to run on a single processing core, and once completed successfully you can use your preferred method for accessing the filesystem to browse the results. Note, for VHUB users, you can find some help in using WebDAV at [https://vhub.org/kb/tips/webdav].

On VHub, please only use the "Local" and "Hub-Submit" Run Style options, the "PBS" and "Condor" options will not work in the VHub environment. The "Hub-Submit" Run Style option is used to submit jobs to CCR's General Compute Cluster.

On other 64-bit Linux platforms, please only use the "Local" Run Style option, the "Hub-Submit", "PBS" and "Condor" options will not work on other 64-bit Linux platforms.

The CPUs per node field corresponds directly with the Titan2D –nt run option.

# Job Monitor Tab

Figure 5.12 displays Job Monitor Tab.

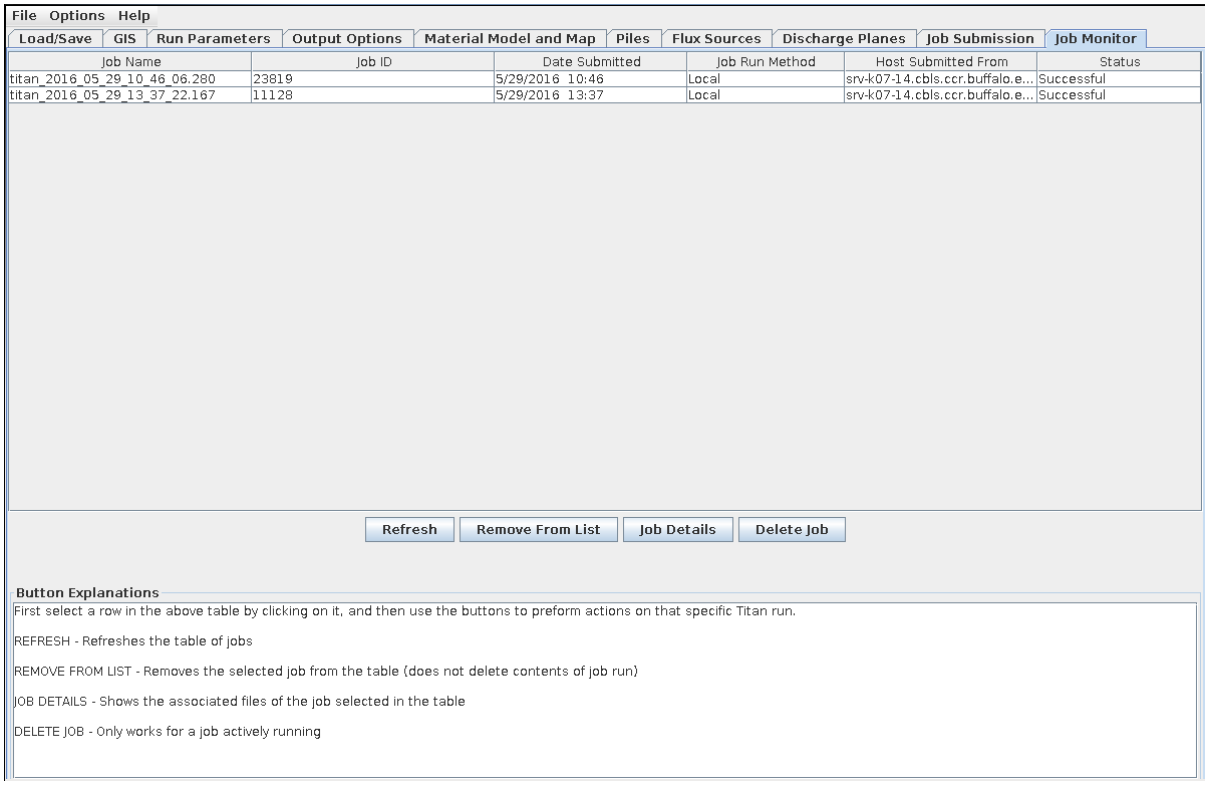


Figure 5-12 Job Monitor Tab

From the Job Monitor Tab, click on a job and then click on the “Job Details” button. A new window will appear that will show the output from the job. Note that you can read here the Output Directory with the full path to your results; you will need this directory when using a Titan2D Viewer tool.

## Job Monitor

Displays information about Titan model runs submitted by the user. It does not display any model run information for jobs submitted by other users. When the tab is initially displayed, it may take a moment to display the job information. The job status displayed is updated every 5 minutes. The job information (not including the status) is retrieved from a database. The job status is determined dynamically. Each user will have a ".titan" directory in their home directory. The database (which is a directory structure of files) is stored there. Should the ".titan" directory be removed, all job details will be lost. The job itself would not be affected.

### Job Summary Table
Not relevant for a local job. For a PBS job, it is the job name.

### Job Name
Not relevant for a local job. For a PBS job, it is the job name.

### Job ID
The job's unique ID. For a local job, it is the jobs process ID. For a PBS job, it is the job id.

### Date Submitted
The date/time that the job was submitted.

### Job Run Method
The method with which the job was submitted (Local, Hub-Submit or PBS).

### Host Submitted From
The host with which the user was on when the job was submitted.

### Status
The current job status.

### Removing an Entry from the list
Display the table item menu popup by right-clicking on any field for the job to be removed from the list. Select "Remove From List" on the menu. The job details will be removed from the list and the job monitor database. The job itself will not be affected.

### Job Details
Figure 5.13 displays Job Details Window.

Display the table item menu popup by right-clicking on any field for which the job details are to be displayed. Select "Job Details" on the menu. Additional information regarding a job will be displayed in a popup window:

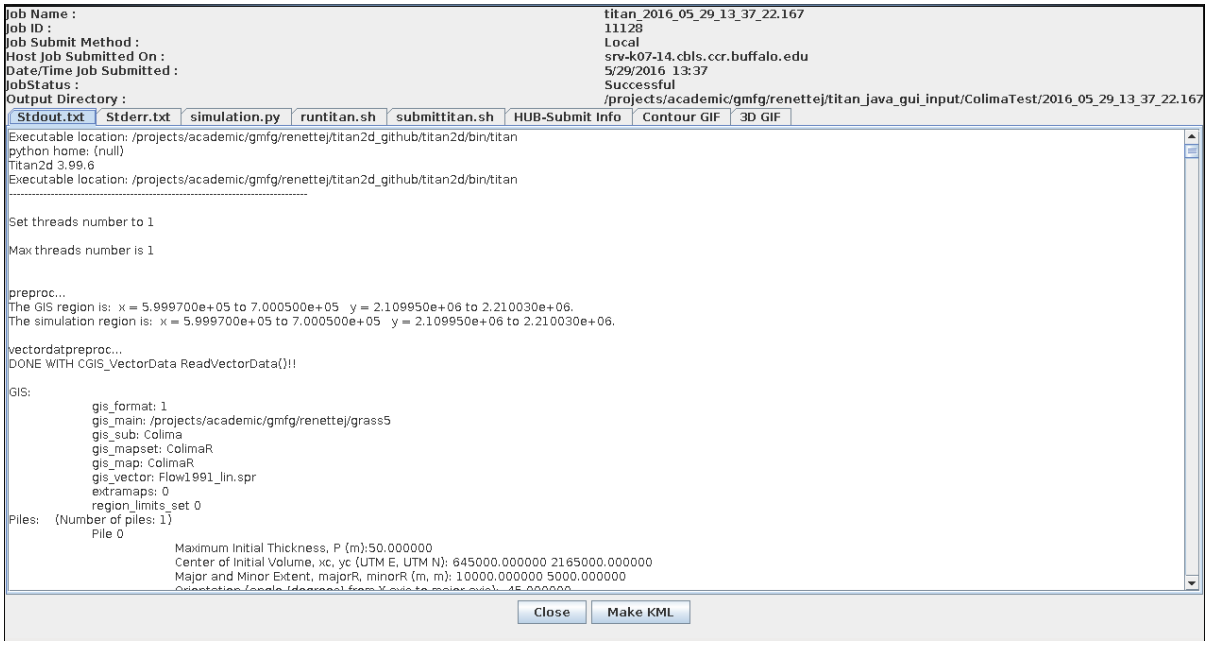


Figure 5-13 Job Details

Information displayed includes the contents of the standard output file, the contents of the standard error file, the contents of the script used to run the Titan2D job, and the contents of the HUB-Submit information file if the job was submitted via HUB-Submit. To view a particular display, left click the tab associated with the display.

The Job Details Window enables the creation and display of Graphics Interchange Format (GIF) formatted files depicting images of the flow of the volcano. To create the GIF formatted files, first select to create XDMF/Paraview visualization output files via the Output Options tab. The Titan2D GUI uses point and pile height information, contained in XDMF/Paraview snapshot output files, to create the GIF formatted files. After the job completes successfully, click on the Contour GIF tab to create and display animation.gif, and/or, the 3D GIF tab to create and display animation3D.gif. The created GIF files are stored to the vizout subdirectory of the timestamped directory created when the job was submitted.

The Job Detail Window contains a “Make KML“ button that enables the creation of zipped KML files depicting images of the flow of the volcano. To create the KML files, first select to create XDMF/Paraview visualization output files via the Output Options tab. After the job completes successfully, click the “Make KML” button. The Titan2D GUI uses point and pile height information, contained in XDMF/Paraview snapshot output files, to create the KMZ files. The KML files are zipped as KMZ files to save on disk space. KMZ files can be input into Google Earth for display; please see the Titan2D Viewer section for more information. The created KML files, pileheightAnimation.kmz and pileHeightLast.kmz, are stored to the vizout subdirectory of the timestamped directory created when the job was submitted.

### Delete Job

Display the table item menu popup by right-clicking on any field for the job to halt execution. Select "Delete Job" on the menu. If the job is in a state where it can be removed, the job will be removed. If the job is in a queue, it will be removed from the queue. If the job is running, it will be aborted.

### Refresh Job List

Push the "Refresh" button to get the most up to date information displayed in the list. The list is automatically updated every 5 minutes.

# 6 Titan2D Python Input Files

Titan2D python input files provide input parameters for running Titan2D as high-level python API scripts. When Titan2D is run from the Titan2D GUI, information from the GUI is translated into required python input files when jobs are submitted via the Job Submission Tab. Users may also create the Titan2D python input files using a standard text editor and run Titan2D as a stand-alone application as described in the Running Titan2D section.

In the sections that follow, python input files are explained in more detail. First, the TitanSimulation class, which controls the simulation and contains calls to methods for selecting simulation parameters and running the simulation, is introduced. Next, the various methods are described in more detail.

# 1 TitanSimulation Class

The TitanSimulation class is the main class, which controls the simulation and contains calls to methods for selecting the simulation parameters. Below a typical Titan2D python input file is shown:

```
sim=TitanSimulation(overwrite_output=True)

sim.setGIS(
   gis_format='GIS_GRASS',
   gis_main='/projects/academic/gmfg/renettej/grass5',
   gis_sub='Colima',
   gis_mapset='ColimaR',
   gis_map='ColimaR',
   gis_vector='Flow1991_lin.spr',
   region_limits=None
)

sim.setScale(
   length_scale=20000.0,
   gravity_scale=9.8,
   height_scale=50.0
)

sim.setNumProp(
   AMR=True,
   number of cells across axis=64,
   order='First'
)

sim.setMatModel(
   model='Coulomb',
   use_gis_matmap=False,
   stopping_criteria=None,
   int_frict=30.0,
   bed_frict=24.0
)

sim.setTimeProps(
   max_iter=1000,
   max_time=300.0
)

sim.setRestartOutput(
   dtime=300.0,
   diter=None,
   keep_all=False,
   keep_redundant_data=False,
   output_prefix='restart'
)
```

```
sim.setTimeSeriesOutput(
   vizoutput=('xdmf'),
   dtime=300.0,
   diter=1000,
   output_prefix='vizout'
)

sim.setStatProps(
   runid=-1,
   edge_height=1.5,
   test_height=3.0,
   test_location=(643000.0, 2168000.0),
   output_prefix=''
)

sim.setOutlineProps(
   enabled=True,
   max_linear_size=1024.0,
   init_size='AMR',
   output_prefix=''
)

sim.addPile(
   pile_type='Paraboloid',
   height=50.0,
   center=[645000.0, 2165000.0],
   radii=[10000.0, 5000.0],
   orientation=-45.0,
   Vmagnitude=2.0,
   Vdirection=-5.0

)

#start simulation
sim=sim.run()
```

# 2 TitanSimulation Class Contructor

## TitanSimulation(keywords_arguments) - Constructor

Creates TitanSimulation instance.

| Argument | Description |
|---|---|
| overwrite_output | Results overwrite safety flag. If set to False (default value) would abort simulation if output files with same names exist in current working directory. If set to True overwrite previous output.<br>**Value type:** bool<br>**Default value:** False |

Example:

```
sim=TitanSimulation(
   overwrite_output=True
)
```

# 3 TitanSimulation Class Methods

## setGIS(keywords_arguments) – Method

Specify digital elevation model

| **Argument** | **Description** |
|---|---|
| gis_format | Specify digital elevation model format<br>**Default value:** 'GIS_GRASS' or 'GDAL'<br>No default value |
| region_limits | Limits simulation to specified region<br>**Value type:** None or list of four floats [x_min,y_min,x_max,y_max]<br>**Default value:** None |
| Additional arguments if **gis_format** is 'GIS_GRASS' | |
| gis_main | **Value type:** string<br>**Must be set** |
| gis_sub | **Value type:** string<br>**Must be set** |
| gis_mapset | **Value type:** string<br>**Must be set** |
| gis_map | **Value type:** string<br>**Must be set** |
| gis_vector | **Value type:** None or string<br>**Default value:** None |
| Additional argument if gis_format is 'GDAL' | |
| gis_map | Location of the digital elevation model on file-system.<br>**Value type:** string<br>**Must be set** |

Example:

```
sim.setGIS(
   gis_format='GIS_GRASS',
   gis_main='/projects/academic/gmfg/renettej/grass5',
   gis_sub='Colima',
   gis_mapset='ColimaR',
   gis_map='ColimaR',
   gis_vector='Flow1991_lin.spr',
   region_limits=None
)
```

## setScale(keywords_arguments) – Method

Set simulation scale

| Argument | Description |
|---|---|
| length_scale | Length scale<br>**Value type:** float<br>**Must be set** |
| gravity_scale | Gravity scale<br>**Value type:** float<br>**Must be set** |
| height_scale | Height scale. If set to None will be calculated based on total volume of movable material<br>**Value type:** None or float<br>**Default value:** None |

Example:

```
sim.setScale(
   length_scale=20000.0,
   gravity_scale=9.8,
   height_scale=50.0
)
```

## setNumProp(keywords_arguments) – Method

Specify numerical methods parameters

| Argument | Description |
|---|---|
| AMR | Enable/Disable adaptive mash refinement, recommended to be True<br>**Value type:** bool<br>**Must be set** |
| number_of_cells_across_axis | Number of elements across smallest pile axis in the beginning of the simulation<br>**Value type:** int<br>**Must be set** |
| order | Numerical PDE solver order. Note that not all solver support second order<br>**Value type:** 'First' or 'Second'<br>**Must be set** |
| geoflow_tiny | Use default unless you know what are you doing<br>**Value type:** float<br>**Default value:** False |

| | |
|---|---|
| short_speed | Use default unless you know what are you doing<br>**Value type:** bool<br>**Default value:** False |

Example:

```
sim.setNumProp(
   AMR=True,
   number_of_cells_across_axis=64,
   order='First',
    geoflow_tiny=0.0001,
    short_speed=False
)
```

## setMatModel(keywords_arguments) – Method

Set materials model.

| **Argument** | **Description** |
|---|---|
| model | Moving material model<br>**Value type:** 'Coulomb', 'TwoPhases-Pitman-Le', 'Voellmy-Salm' or 'Pouliquen-Forterre'<br>**Must be set** |
| Additional arguments if **model** is 'Coulomb' | |
| int_frict | Debris material internal friction angle<br>**Value type:** float<br>**Must be set** |
| bed_frict | Bed friction angle<br>**Value type:** float<br>**Must be set** |
| stopping_criteria | Use default unless you know what are you doing<br>**Value type:** None or 'DragBased'<br>**Default value:** None |
| use_gis_matmap | Use default unless you know what are you doing<br>**Value type:** bool<br>**Default value:** False |
| Additional arguments if **model** is 'TwoPhases-Pitman-Le' | |
| int_frict | Debris material internal friction angle<br>**Value type:** float<br>**Must be set** |
| bed_frict | Bed friction angle<br>**Value type:** float<br>**Must be set** |
| Additional arguments if **model** is 'Pouliquen-Forterre' | |

| | |
|---|---|
| phi1 | **Value type:** float<br>**Must be set** |
| phi2 | **Value type:** float<br>**Must be set** |
| phi3 | **Value type:** float<br>**Must be set** |
| Beta | **Value type:** float<br>**Must be set** |
| L_material | **Value type:** float<br>**Must be set** |
| Additional arguments if **model** is 'Voellmy-Salm' | |
| mu | **Value type:** float<br>**Must be set** |
| xi | **Value type:** float<br>**Must be set** |

Example:

```
sim.setMatModel(
   model='Coulomb',
   use_gis_matmap=False,
   stopping_criteria=None,
   int_frict=30.0,
   bed_frict=24.0
)
```

## addPile(keywords_arguments) – Method

Add pile of debris, several piles can be added. This setter must be present only if you want to add pile.

| **Argument** | **Description** |
|---|---|
| height | **Value type:** float<br>**Must be set** |
| center | **Value type:** list of two float [x,y]<br>**Must be set** |
| radii | **Value type:** list of two float [major_radius,minor_radius]<br>**Must be set** |
| orientation | Value type: float<br>Default value: 0.0 |
| Vmagnitude | Value type: float<br>Default value: 0.0 |

| | |
|---|---|
| Vdirection | Value type: float<br>Default value: 0.0 |
| pile_type | Value type: 'Paraboloid' or 'Cylinder'<br>Default value: 'Cylinder' |
| vol_fract | Parameter for case if model is 'TwoPhases-Pitman-Le', otherwise should not be set.<br>Value type: float<br>Must be set if model is 'TwoPhases-Pitman-Le' |

Example:

```
sim.addPile(
   pile_type='Paraboloid',
   height=50.0,
   center=[645000.0, 2165000.0],
   radii=[10000.0, 5000.0],
   orientation=-45.0,
   Vmagnitude=2.0,
   Vdirection=-5.0
)
```

## addFluxSource(keywords_arguments) – Method

Add flux source. This setter must be present only if you want to flux source.

| **Argument** | **Description** |
|---|---|
| influx | **Default value:** float<br>**Must be set** |
| start_time | **Default value:** float<br>**Must be set** |
| end_time | **Default value:** float<br>**Must be set** |
| center | **Value type:** list of two float [x,y]<br>**Must be set** |
| radii | **Value type:** list of two float [major_radius,minor_radius]<br>**Must be set** |
| orientation | **Value type:** float<br>**Default value:** 0.0 |
| Vmagnitude | **Value type:** float<br>**Default value:** 0.0 |
| Vdirection | **Value type:** float<br>**Default value:** 0.0 |

Example:

```
sim.addFluxSource(
   influx=10.0,
   start_time=0.0,
   end_time=10000.0,
   center=[644956.0, 2157970.0],
   radii=[55.0, 55.0],
   orientation=0.0,
   Vmagnitude=0.0,
   Vdirection=0.0
)
```

## addDischargePlane (x_a,y_a,x_b,y_b) – Method

Add discharge plane. This setter must be present only if you want to add discharge plane.

| **Argument** | **Description** |
|---|---|
| x_a | **Value type:** float<br>**Must be set** |
| y_a | **Value type:** float<br>**Must be set** |
| x_b | **Value type:** float<br>**Must be set** |
| y_b | **Value type:** float<br>**Must be set** |

Example:

```
sim.addDischargePlane(637380.0, 2145800.0, 664380.0, 2169800.0)
```

## setTimeProps(keywords_arguments) – Method

Set simulation time and number of iteration. If this setter skipped default values will be used.

| **Argument** | **Description** |
|---|---|
| max_iter | Perform max_iter iteration at maximum, if None don't use number of iteration as exit criteria<br>**Value type:** None or float<br>**Default value:** None |
| max_time | Perform max_time simulation seconds at maximum, if None don't use max_time as exit criteria<br>**Value type:** None or float<br>**Default value:** None |

Example:

```
sim.setTimeProps(
   max_iter=1000,
   max_time=300.0
)
```

## setRestartOutput(keywords_arguments) – Method

Parameter for dumping restart files. If this setter skipped default values will be used.

| **Argument** | **Description** |
|---|---|
| dtime | Dump restart files every dtime simulation seconds. If set to None do not use it as dumping criteria<br>**Value type:** None or float<br>**Default value:** None |
| diter | Dump restart files every diter iterations. If set to None do not use it as dumping criteria<br>**Value type:** None or int<br>**Default value:** 1000 |
| keep_all | Keep all restart files. If set to False will keep only last restart file<br>**Value type:** bool<br>**Default value:** False |
| keep_redundant_data | Keep redundant data<br>**Value type:** bool<br>**Default value:** False |
| output_prefix | Prefix for restart files<br>**Value type:** string<br>**Default value:** 'restart' |

Example:

```
sim.setRestartOutput(
   dtime=300.0,
   diter=None,
   keep_all=False,
   keep_redundant_data=False,
   output_prefix='restart'
)
```

## setTimeSeriesOutput(keywords_arguments) – Method

Set frequency of snapshots output for further visualization. If this setter can be skipped no output will be done, save some disk space.

| **Argument** | **Description** |
|---|---|

| | |
|---|---|
| vizoutput | Snapshot format<br>**Value type:** None or 'tecplot', 'meshplot', 'xdmf', 'grasssites' or list of these values<br>**Must be set** |
| dtime | Write snapshot every dtime simulation seconds. If set to None do not use it as dumping criteria<br>**Value type:** None or float<br>**Default value:** None |
| diter | Write snapshot every diter iterations. If set to None do not use it as dumping criteria<br>**Value type:** None or int<br>**Default value:** 1000 |
| output_prefix | Prefix for vizout files<br>**Value type:** string<br>**Default value:** 'vizout' |

Example:

```
sim.setTimeSeriesOutput(
   vizoutput=('xdmf', 'meshplot'),
   dtime=300.0,
   diter=1000,
   output_prefix='vizout'
)
```

## setStatProps(keywords_arguments) – Method

If this setter skipped default values will be used.

| **Argument** | **Description** |
|---|---|
| edge_height | **Value type:** None or float<br>**Default value:** None |
| test_height | **Value type:** None or float<br>**Default value:** None |
| test_location | **Value type:** None or list of two floats<br>**Default value:** None |
| runid | Integer suffix for statistical and outline output<br>**Value type:** int<br>**Default value:** -1 |
| output_prefix | Prefix for statistical output files<br>**Value type:** string<br>**Default value:** " |

Example:

```
sim.setStatProps(
   runid=-1,
   edge_height=1.5,
   test_height=3.0,
   test_location=(643000.0, 2168000.0),
   output_prefix=''
)
```

## setOutlineProps(keywords_arguments) – Method

If this setter skipped default values will be used.

| **Argument** | **Description** |
|---|---|
| enabled | Enabled/disable outline calculations<br>**Value type:** bool<br>**Default value:** True |
| max_linear_size | Maximal size of outline grid in x or y direction<br>**Value type:** int<br>**Default value:** 1024 |
| init_size | Method for calculation of initial outline grid size. AMR – using highest possible resolution. DEM - use DEM resolution. In both cases grid size is decreased until x and y side will be less than max_linear_size.<br>**Value type:** 'AMR' or 'DEM'<br>**Default value:** 'AMR' |
| output_prefix | Prefix for outline output files<br>**Value type:** string<br>**Default value:** '' |

Example:

```
sim.setOutlineProps(
   enabled=True,
   max_linear_size=1024.0,
   init_size='AMR',
   output_prefix=''
)
```

## run() – Method

Perform simulation.

Example:

```
sim.run()
```

# 7 Probabilistic Mass Flow Simulations

In addition to the probabilistic mass flow simulation procedures presented in this section, please see the VHub Titan2D Hazard Map Emulator Workflow tool which also extends Titan2D mass-flow simulation capability, by producing hazard maps which display the probability of a Titan2D flow depth reaching a critical height over a period of time.

What is LHS? LHS stands for Latin Hypercube Sampling. It is a constrained sampling method that can converge in far fewer samples than Monte Carlo. It works by dividing each random dimension into N equally probable bins. A sample point within each bin is randomly chosen. Each bin is divided into 2 equally probable parts, and a sample point is generated in each of the new bins that do not already have one. This refinement should be repeated until the desired level of accuracy is obtained.

Titan2D has the capability to perform LHS simulations with uncertain bed friction (either normally or uniformly distributed) or uncertain volumes (uniformly distributed).

To perform a Titan2D LHS stochastic simulation, follow the instructions below. Remember each Titan2D run in an LHS simulation run must run on a single processor (you must have entered ”1” for the number of processors through the GUI).

## Step 1: Generating the LHS sample points

Decide whether you want to perform a simulation with an uncertain/random/stochastic bed friction angle OR (not both) an uncertain/random/stochastic initial pile volume. If you choose the former type “./lhsbed” (no quotes) and answer the questions it asks you. If you choose the later type “./lhsvol” and answer the questions it asks you. These will produce a file named ”stat ctl.bed” or ”stat ctl.vol” respectively.

## Step 2: Starting the LHS Simulation

Follow the instructions here in place of what you find in Running Titan2D when running a stochastic simulation.

If submitting a batch job, edit the pbs script ″pbslhs″ don′t forget to pass the proper ″stat ctl.″ file (generated in Step 1) to the ″dist-stats.pl″ script. then type ″qsub pbslhs″ at the command prompt.

It is also possible to run the ″dist-stats.pl″ script from the command line on a single computer, however, this will take quite a long time. To launch an lhs simulation without using pbs, type

″perl dist-stats.pl –ctlfile=stat ctl.<extension>″ at the command line. The perl script auto detects the number of CPU′s (on computers running Linux) and performs that many runs simultaneously.

## Step 3: Computing Statistics

After the LHS simulation is completed type ”./lhstitanstats” at the command line. This will produce a file named ”statout.plot” which can be used to make convergence plots. There are 20 entries on each line, the first entry is the number of sample points used to generate the statistics on this line, the second is the estimated probability (0-1) that the flow reached the test height at the test point (which are entered through the Titan2D GUI), the rest are the mean, standard deviation, and skewness of the end state properties

a) volume averaged velocity

b) maximum height

c) x coordinate of the centroid

d) y coordinate of the centroid

e) x direction spread

f) y direction spread

The spread in each dimension is defined as the difference between the maximum and minimum coordinates at which the pile height is greater than or equal to the edge height. The edge height is set through the Titan2D GUI.

## Step 4: Plotting Convergence Studies of Statistics

We have provided a perl script ”plot stats.pl” that uses gnuplot to make convergence plots from the data in ”statout.plot”. Simply type perl plot stats.pl from within the simulation directory to generate 3 ”.ps” files suitable for printing on printers setup for UNIX, or viewing through programs such as ghostview or gimp. If you prefer not to use the ”plot stats.pl” script, the data file ”statout.plot” (see Step 3) is straight forward enough that you can easily use any other plotting tool, such as a spreadsheet package or matlab, that you are familiar with.

# 8 Titan2D Viewers

## 1. Paraview visualization application

Paraview is an open-source data visualization software package. It is available from [http://www.paraview.org/] for no cost. Titan can output data in Paraview readable e**X**tensible **D**ata **M**odel and **F**ormat (XDMF [http://www.arl.hpc.mil/ice/]). Although not required, Xdmf format utilizes HDF5 library to store actual data (computational grid, pile heights, etc). We strongly recommend building Titan2D with HDF5 support. HDF5 is a free-software, downloadable from HDF website [http://hdf.ncsa.uiuc.edu/HDF5/].

When selected, Titan2D will generate data files named xdmf*.xmf and xdmf*.h5, if titan was compiled with HDF5 support. Otherwise only xdmf*.xmf files will be created, all the data being in ASCII format. These files can take considerably larger disc space in comparison to HDF5 files. We have tested only Paraview-2.6 for titan. But we don't foresee any problem with other versions. Following are some quick steps to:

View Titan2D output in Paraview:

1. Open Paraview using paraview command on Linux
2. Open xdmfxxxxxxx.xmf file, using **Open Data** in **File** menu
3. Click on **Accept** button on the left-hand panel (Map should appear in the main window)
4. Adjust view using mouse buttons
   - hold down left–button and move mouse to rotate the view
   - hold down middle–button and move mouse to pan the view
   - hold down right–button and move mouse to zoom-in or zoom-out
5. Change to **Display** tab on left-panel
6. Select or deselect flow properties in the color section
7. You can change the contour levels using **Edit Color Map**

Create animations:

1. Open xdmfxxxxx.xmf file representing 1st time-step using steps 1-7
2. Add files for subsequent time-steps using **Timesteps** under **Parameters** tab
3. From **View** menu, select **Keyframe Animation**, left panel will start showing animation controls
4. Change no. of frames to no. of files added in step 2
5. Select **Source** to 1st filename (e.g. xdmf0000000000.xmf)
6. Click **Add Frame** once, select **Value** as 1st filename
7. Click **Add Frame** again and select **Value** as last filename
8. Adjust time to simulation time
9. Click on VCR style play (✄) button to view animation
10. To save animation click on right-most button on VCR controls (the one with film symbol on it)

# 8. Google Earth

KMZ files created by the Titan2D GUI, pileheightAnimation.kmz and pileHeightLast.kmz, can be opened with Google Earth for geo-referenced displays. Please see the Job Details section for more information on how to create KMZ files.

As of this writing, instructions for installing Google Earth on PC, Mac or Linux systems can be found at [https://www.google.com/earth/download/ge/agree.html].

On VHub, to start Google Earth, open a workspace terminal window and enter: google-earth. Use the Google Earth File Open menu option to open a .kmz file.

Displayed in Figure 8.1, is an example Google Earth display of Titan2D simulation results.

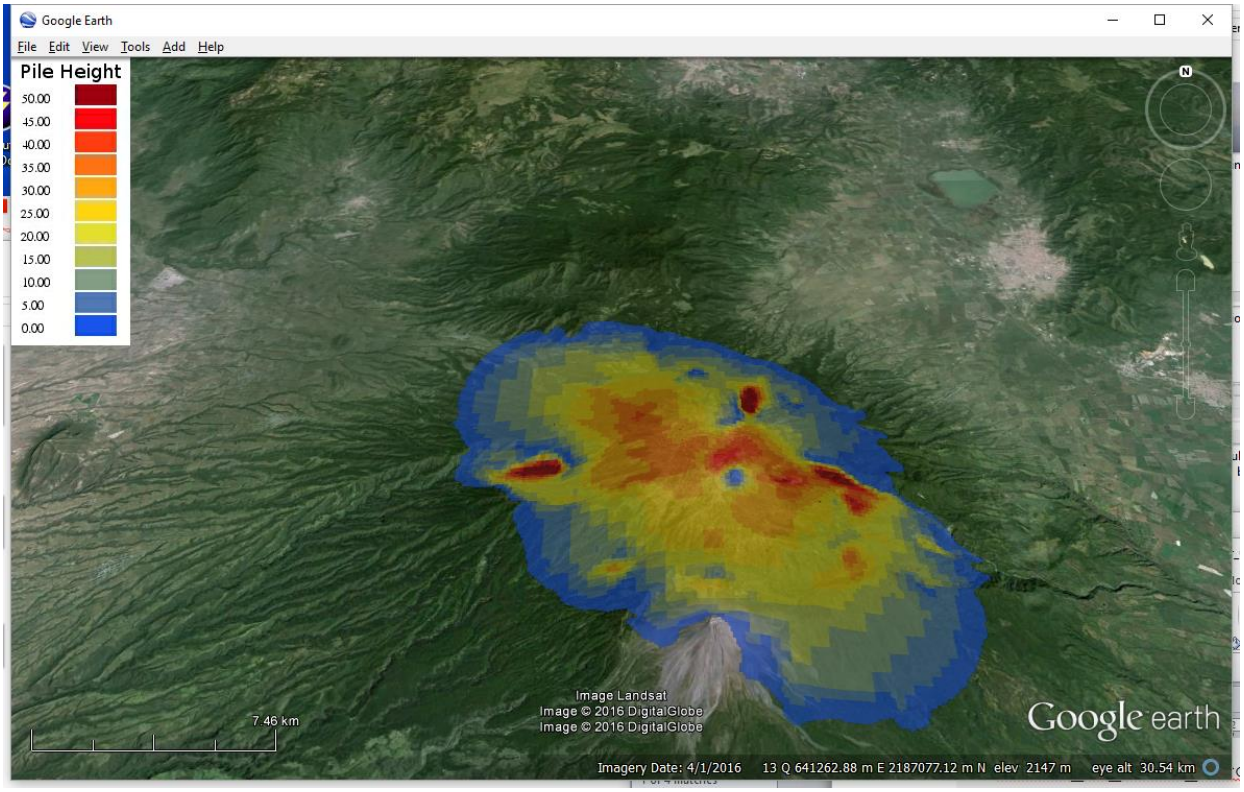


Figure 8-1 Example Google Earth Display of Simulation Results

# 9 Appendix

## A. Governing "Shallow Water" Equations

The shallow-water model conservation equations1 solved by Titan are given as:

$$\frac{\partial}{\partial t}(\vec{U}) + \frac{\partial}{\partial x}(\vec{F}(\vec{U})) + \frac{\partial}{\partial y}(\vec{G}(\vec{U})) = \vec{S}(\vec{U})$$

where:

$$\vec{U} = \begin{bmatrix} h \\ hV_x \\ hV_y \end{bmatrix}$$

is the vector of conserved state variables

(with h = flow depth, hVx = x-momentum, hVy = y-momentum).

$$\vec{F}(\vec{U}) = \begin{bmatrix} hV_x \\ hV_x^2 + \frac{1}{2}k_{ap}g_z h^2 \\ hV_x V_y \end{bmatrix}$$

is the mass and momentum fluxes in the x-direction

(with hVx = mass flux in x-direction, $hV_x^2 + \frac{1}{2}k_{ap}g_z h^2$= x-momentum flux in x-direction,

hVxVy = y-momentum flux in x-direction).

---

1For complete derivation, see: Denlinger, R.P., and Iverson, R.M. (2004), Granular avalanches across irregular three-dimensional terrain: 1. Theory and computation, J. Geophys. Res., 109, F01014, doi:10.1029/2003JF000085.

$$\vec{G}(\vec{U}) = \begin{bmatrix} hV_y \\ hV_xV_y \\ hV_y^2 + \frac{1}{2}k_{ap}g_z h^2 \end{bmatrix}$$

is the mass and momentum fluxes in the y-direction

(with hVy = mass flux in y-direction, hVxVy = x-momentum flux in y-direction, $hV_y^2 + \frac{1}{2}k_{ap}g_z h^2$ = y-momentum flux in y-direction).

$$\vec{S}(\vec{U}) = \begin{bmatrix} 0 \\ g_x h - hk_{ap}\text{sign}\left(\frac{\partial V_x}{\partial y}\right)\frac{\partial}{\partial y}(g_z h)\sin\phi_{int} - \frac{V_x}{\sqrt{V_x^2 + V_y^2}}\max(g_z + \frac{V_x^2}{r_x}, 0)h\tan\phi_{bed} \\ g_y h - hk_{ap}\text{sign}\left(\frac{\partial V_y}{\partial x}\right)\frac{\partial}{\partial x}(g_z h)\sin\phi_{int} - \frac{V_y}{\sqrt{V_x^2 + V_y^2}}\max(g_z + \frac{V_y^2}{r_y}, 0)h\tan\phi_{bed} \end{bmatrix}$$

is the vector of driving and dissipative source terms

(with gxh = driving gravitational force in x-direction, $-\text{hk}_{ap}\text{sign}\left(\frac{\partial V_x}{\partial y}\right)\frac{\partial}{\partial y}(\text{g}_z\text{h})\sin\phi_{\text{int}}$ = dissipative internal frictional force in x-direction, $-\frac{V_x}{\sqrt{V_x^2+V_y^2}}\max(g_z + \frac{V_x^2}{r_x}, 0)h\tan\phi_{bed}$= dissipative basal frictional force in x-direction; similar terms for y-direction).

Note Also:

$k_{ap}$ = active/passive lateral stress coefficient term, where "active" $k_{ap}$ assumes a smaller value in a diverging flow, while $k_{ap}$ "passive" takes on a larger value and means that the flow is converging.

## B. Voellmy-Salm and Pouliquen-Forterre Rheologies

Support for several alternate rheologies to the Mohr-Coulomb model have been added to Titan2D, namely the Voellmy-Salm and Pouliquen-Forterre models.

In the following two sections we will briefly introduce these recently added models.

## Voellmy-Salm Model

The Voellmy-Salm model mostly is used for simulation of snow avalanches on real terrains; however, it can be used for debris avalanches also. The depth-averaged equation modified for the Voellmy-Salm model is the following [1]:

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}(h\overline{u}) + \frac{\partial}{\partial y}(h\overline{v}) = 0$$

$$\frac{\partial}{\partial t}(h\overline{u}) + \frac{\partial}{\partial x}\left(h\overline{u}^2 + \frac{1}{2}g_z h^2\right) + \frac{\partial}{\partial y}(h\overline{uv}) = S_x$$

$$\frac{\partial}{\partial t}(h\overline{v}) + \frac{\partial}{\partial x}(h\overline{uv}) + \frac{\partial}{\partial y}\left(h\overline{v}^2 + \frac{1}{2}g_z h^2\right) = S_y \qquad (1)$$

Here the source terms are defined as follows:

$$S_x = g_x h - \frac{\overline{u}}{\| \underset{\sim}{\overline{U}} \|}\left[\mu h\left(g_z + \frac{\| \underset{\sim}{\overline{U}} \|^2}{r_x}\right) + \frac{\| \underset{\sim}{g} \|}{\xi} \| \underset{\sim}{\overline{U}} \|^2\right]$$

$$S_y = g_y h - \frac{\overline{v}}{\| \underset{\sim}{\overline{U}} \|}\left[\mu h\left(g_z + \frac{\| \underset{\sim}{\overline{U}} \|^2}{r_y}\right) + \frac{\| \underset{\sim}{g} \|}{\xi} \| \underset{\sim}{\overline{U}} \|^2\right] \qquad (2)$$

Where $\underset{\sim}{\overline{U}} = (\overline{u}, \overline{v})$ and $\underset{\sim}{g} = (g_x, g_y, g_z)$.

The Voellmy-Salm approach splits the total basal friction into a velocity independent dry-Coulomb term which is proportional to the normal stress at the flow bottom (friction coefficient $\mu$) and a velocity dependent viscous or turbulent friction term (friction coefficient $\xi$). In addition, the effects of local curvatures are taking into account [2].

The division of the total basal friction into velocity independent and dependent parts allows the modelling of avalanche behavior when the avalanche is flowing with a high velocity in the acceleration zone and close to stopping in the runout zone.

A fundamental assumption of the Voellmy-Salm model is that shear deformations are concentrated near the basal flow surface. This behavior is also observed for flows of various particle types and bed roughness conditions.

## Pouliquen-Forterre Model

For the Pouliquen-Forterre model, also known as the Pouliquen-Forterre basal friction model, Pouliquen & Forterre [4] found two critical slope inclination angles as functions of the flow thickness, namely $\phi_{start}(h)$ and $\phi_{stop}(h)$. The function $\phi_{stop}(h)$ gives the slope angle at which a steady uniform flow leaves a deposit of thickness $h$, while $\phi_{start}(h)$ is the angle at which a layer of thickness $h$ is mobilized. Basically, the knowledge of these two functions $\mu_{start}(h) = \tan(\phi_{start}(h))$ and $\mu_{stop}(h) = \tan(\phi_{stop}(h))$ is sufficient to define the empirical friction law $\mu_b(\| \underset{\sim}{\overline{U}} \|, h)$ in the whole range of velocity and thickness. The thickness of a deposit left by a steady uniform flow at an inclination angle $\phi$ is denoted by $h_{stop}(\phi)$, which is the inverse function of $\phi_{stop}(h)$. An empirical dependence was found by Pouliquen [3] between the ratio of the flow thickness $h$ to $h_{stop}(\phi)$ and the Froude number, $Fr = \| \underset{\sim}{\overline{U}} \| / \sqrt{h g_z}$, which is given as:

$$h_{stop}(\phi) = \beta \frac{h}{Fr} \tag{3}$$

Where $\beta$ is a property of the granular flowing material. The expression (3) is only valid for the $Fr > \beta$ which indicates the steady uniform flow or dynamic regime. As a result,

for the basal friction coefficient in the dynamic friction regime where Fr ≥ β:

$$\mu(h, Fr) = \mu_{stop}(h\beta/Fr) \tag{4}$$

In the intermediate friction regime when 0 < Fr < β, the friction coefficient is given by a power law extrapolation between the friction laws in the static and dynamic friction regimes as:

$$\mu(h, Fr) = \left(\frac{Fr}{\beta}\right)^{\gamma} (\mu_{stop}(h) - \mu_{start}(h)) + \mu_{start}(h) \tag{5}$$

Where $\gamma = 10^{-3}$ is the power of the chosen extrapolation 4.

Finally, for the static friction coefficient, which holds for Fr = 0:

$$\mu(h, 0) = \mu_{start}(h) \tag{6}$$

The functions μstop and μstart are given by fits to experimental measurements as transitions between the relevant critical angles. Therefore, they are written as:

$$\mu_{stop}(h) = \tan\phi_1 + \frac{\tan\phi_2 - \tan\phi_1}{1 + h/L_{material}} \tag{7}$$

and

$$\mu_{start}(h) = \tan\phi_3 + \frac{\tan\phi_2 - \tan\phi_1}{1 + h/L_{material}} \tag{8}$$

The critical angles $\phi 1$, $\phi 2$ and $\phi 3$ and the parameter Lmaterial (the characteristic depth of the flow over which a transition between the angles $\phi 1$ and $\phi 2$ occurs) in addition to the β are the material properties.

Savage & Hutter 5 introduced depth-averaged or Saint-Venant equations in the context of granular flows. Pouliquen & Forterre assumed that for a flow down a slope making an angle $\phi$ with the horizontal plane, the source terms of the depth-averaged equations take the following form:

$$S_x = g_x h - g_z h \left( \frac{\overline{u}}{\| \underset{\sim}{\overline{U}} \|} \mu_b \left( \| \underset{\sim}{\overline{U}} \|, h \right) + \frac{\partial h}{\partial x} \right)$$

$$S_y = g_y h - g_z h \left( \frac{\overline{v}}{\| \underset{\sim}{\overline{U}} \|} \mu_b \left( \| \underset{\sim}{\overline{U}} \|, h \right) + \frac{\partial h}{\partial \mathrm{y}} \right) \tag{9}$$

Note That the last terms in equations (9) represent the pressure force related to the gradient of the pile height.

---

1 M. Christen, J. Kowalski, and P. Bartelt. RAMMS: Numerical simulation of dense snow avalanches in three-dimensional terrain. *Cold Regions Science and Technology*, 63:1–14, 2010. doi: 10.1016/j. coldregions.2010.04.005

2 J. Fischer, J. Kowalski, and S. P. Pudasaini. Topographic curvature effects in applied avalanche modeling. *Cold Regions Science and Technology*, 74-75:21–30, 2012. doi: 10.1016/j.coldregions.2012. 01.005

3 Oliver Pouliquen. Scaling laws in granular flows down rough inclined planes. *Physics of Fluids*, 11 (3):542–548, 1999

4 Oliver Pouliquen and Yoël Forterre. Friction law for dense granular flows: application to the motion of a mass down a rough inclined plane. *Journal of Fluid Mechanics*, 453:133–151, 2002. doi: 10.1017/ S0022112001006796

5 S. B. Savage and K. Hutter. The motion of a finite mass of granular material down a rough incline. *Journal of Fluid Mechanics*, 199:177, 1989. ISSN 0022-1120. doi: 10.1017/S0022112089000340

# C. GIS Information

## GIS Header File

The header file contains information about the projection, elevation data bounding box coordinates, number of columns and rows, and resolution. Additional information is stored to allow the correct interpretation of the binary data file. For a GIS Map defined in the Titan2D GUI, the corresponding header file is named GIS Map and should be located at: GIS Information Main Directory/GIS Sub-Directory/GIS Map Set/cellhd/GIS Map. This information can be used with the GIS of your choosing,including GRASS.

The header-file is an ASCII file, with the following lines:

proj: projection code
zone: UTM projection zone
north: upper Y-direction coordinate
south: lower Y-direction coordinate
east: right X-direction coordinate
west: left X-direction coordinate cols: number of columns
rows: number of rows
e-w resol: resolution in X-direction
n-s resol: resolution in Y-direction
format: binary data format
compressed: compression flag

Projection code and the UTM projection zone are not used, therefore one can use any value, such as 1 for projection code (in GRASS, 1 corresponds to UTM projection). The UTM projection zone can also be any number. If desired, the correct UTM zone number should be used in UTM projection zone.

The number of columns multiplied by resolution in X-direction must be equal to the difference between right X-direction coordinate and left X-direction coordinate. The number of rows multiplied by resolution in Y-direction must be equal to the difference between the upper Y-direction coordinate and the lower Y-direction coordinate.

Binary data format must be -1 indicating that the data are IEEE float values. Compression flag must be 1 if is compressed, 0 otherwise.

More information can be found in the GRASS 5.0 Programmer's Manual.

Example header file:

proj: 1
zone: 13
north: 2210030
south: 2109950
east: 700050
west: 599970
cols: 1667
rows: 1667
e-w resol: 60.0359928
n-s resol: 60.0359928
format: -1
compressed: 1

## GIS Data File

Elevation data is stored in the data file, a binary file stored at the fcell directory. For a GIS Map defined in the Titan2D GUI, the corresponding data file is named GIS Map and should be located at: GIS Information Main Directory/GIS Sub-Directory/GIS Map Set/fcell/GIS Map The data in the file are IEEE float values, stored in Big-Endian byte order. Data can be compressed using zlib. In a uncompressed file, elevation values are stored sequentially from the first column of the first row to the last column of the last row.

An example uncompressed file for a 4 row by 5 column data would be:

| | | | | |
|---|---|---|---|---|
| row[0]column[0] | row[0]column[1] | row[0]column[2] | row[0]column[3] | row[0]column[4] |
| row[1]column[0] | row[1]column[1] | row[1]column[2] | row[1]column[3] | row[1]column[4] |
| row[2]column[0] | row[2]column[1] | row[2]column[2] | row[2]column[3] | row[2]column[4] |
| row[3]column[0] | row[3]column[1] | row[3]column[2] | row[3]column[3] | row[3]column[4] |

where row[ ]column[ ] are float values in IEEE format.

If the file is compressed, the first byte of the file indicates the number of bytes used for each elevation data and must be 4 (0x04). The next four bytes indicate the initial position of the first row of data, and the sequence continues with the initial position of each row for all rows. A compressed row is flagged by the first byte, which must be 0x31. The difference between the

row to be read and the next row is used to read the set of compressed values, which are compressed using zlib compress function. The uncompressed data will correspond to the elevation float values.

## GIS Material Header File

The header file contains information about the projection, material data bounding box coordinates, number of columns and rows, and resolution. Note that this information should be the same as the elevation data header file. Additional information is stored to allow the correct interpretation of the binary data file. For a GISMap defined in the Titan2D GUI, the corresponding header file is named GISMap Mat and should be located at: GIS Information Main Directory/GIS Sub-Directory/GIS Map Set/cellhd/. This information can be used with the GIS of your choosing, including GRASS. The header-file is an ASCII file, with the following lines:

proj: projection code zone: UTM projection zone
north: upper Y-direction coordinate
south: lower Y-direction coordinate
east: right X-direction coordinate
west: left X-direction coordinate
cols: number of columns
rows: number of rows
e-w resol: resolution in X-direction
n-s resol: resolution in Y-direction
format: binary data format
compressed: compression flag

Projection code and the UTM projection zone are not used, therefore one can use any value, such as 1 for projection code (in GRASS, 1 corresponds to UTM projection). The UTM projection zone can also be any number. If desired, the correct UTM zone number should be used in UTM projection zone.

The number of columns multiplied by resolution in X-direction must be equal to the difference between right X-direction coordinate and left X-direction coordinate. The number of rows multiplied by resolution in Y-direction must be equal to the difference between the upper Y-direction coordinate and the lower Y-direction coordinate. Binary data format must be 0 indicating that the data are byte (8 bits) values. Compression flag must be 1 if is compressed, 0 otherwise.

Example header file:

proj: 1
zone: 13
north: 2210030
south: 2109950
east: 700050
west: 599970
cols: 1667
rows: 1667
e-w resol: 60.0359928
n-s resol: 60.0359928
format: 0
compressed: 1

## GIS Material File

Material data is stored in the data file, a binary file stored at the cell directory. For a GISMap defined in the Titan2D GUI, the corresponding material data file is named GISMap Mat and should be located at: GIS Information Main Directory/GIS Sub-Directory/GIS Map Set/cell/GISMap Mat. The data in the file are byte values (8 bits). Data can be compressed using zlib. In a uncompressed file, material index values are stored sequentially from the first column of the first row to the last column of the last row. An example uncompressed file for a 4 row by 5 column data would be:

| | | | | |
|---|---|---|---|---|
| row[0]column[0] | row[0]column[1] | row[0]column[2] | row[0]column[3] | row[0]column[4] |
| row[1]column[0] | row[1]column[1] | row[1]column[2] | row[1]column[3] | row[1]column[4] |
| row[2]column[0] | row[2]column[1] | row[2]column[2] | row[2]column[3] | row[2]column[4] |
| row[3]column[0] | row[3]column[1] | row[3]column[2] | row[3]column[3] | row[3]column[4] |

where row[ ]column[ ] are byte (8 bits) index values.

If the file is compressed, the first byte of the file indicates the number of bytes used for each elevation data and must be 1 (0x01). The next four bytes indicates the initial position of the first row of data, and the sequence continues with the initial position of each row for all rows. A compressed row is flagged by the first byte, which must be 0x31. The difference between the row to be read and the next row is used to read the set of compressed values, which are compressed using zlib compress function. The uncompressed data will correspond to the material index values.

## GIS Material Categories File

Material categories file describes the material name associated to the index stored in the data file. For each integer value present in the data file, the name of the material is defined. The categories file is stored at the cats directory. For a GISMap defined in the Titan2D GUI, the corresponding material categories data file is named GISMap.Mat and should be located at: GIS Information Main Directory/GIS Sub-Directory/GIS Map Set/cats/GISMap.Mat. The categories file is an ASCII file, with the following lines:

# <max cat> categories Vector map: <some name> 0.00 0.00 0.00 0.00

<ind a>:<Mat a>

<ind b>:<Mat b>

<ind n>:<Mat n>

<ind max cat>:<Mat max cat>

<max cat> is the biggest index of material; <some name> is a name (not relevant here); <ind a> is the index of material A; <Mat a> is the name of material A; <ind b> is the index of material B; <Mat b> is the name of material B; ); <ind n> is the index of material N; <Mat n> is the name of material N; ); <ind max cat> is the biggest index and corresponding to material M; and < Mat max cat> is the name of material M. An example categories file with 5 categories and biggest index 30 data would be: # 30 categories

Vector map: FrictionMaterial 0.00 0.00 0.00 0.00

4:MaterialA 15:MaterialB 18:MaterialC 20:MaterialD 30:MaterialE