Documentation for Iverson (2000) model

Landslide safety factor from transient pore pressure modelling

Simon M. Mudd, Boris Gailleton, School of Geosciences, University of Edinburgh (b.gailleton@sms.ed.ac.uk)

Table of Contents

Chapter 1. Forewords

This documentation is part of a deliverable for the project GSTP project in the WP 4. This document briefly describe the purpose of this model, and detail how to install, run a single simulation and run a monte-carlo analysis to constrain the model. It also details the outputs in order to explore the data.

Chapter 2. Introduction

WP 4 aims to evaluate the performance of using remote sensing data to detect and anticipate landslide activity. Simulating and anticipating slope failure can be a delicate operation. A common approach consists in modelling background pore pressure to generate static safety maps determining area of risks, Iverson (2000) suggested a transient pore pressure modelling approach to consider the time-evolution of the effect of precipitation and soil water content on slope stability. Such a model is highly case-dependent and requires a significant amount of expensive and time-consuming in-situ monitoring of soil hydrolic, mechanic and hydrogeologic features. The aim of this code and WP is to use an already-monitored site to apply Iverson (2000) model and compare the performance of the model constrained with in-situ data with the model constrained with remote-sensing derived data. The following documentation proposes a protocol to run the model from the different set of data and how to visualise and explore the outputs. The core of the modelling code is written in C++ to ensure reasonable performances and most of the automation for multiple simulations and I/O management is written in `python`.

2.1. Parameters

A number of parameters are required to manually enter in the model:

- The model categorically needs precipitation data input to simulate the transient pore pressure evolution in the modeled soil column. This data has to be formated as *precipitation intensity* in length unit per time unit, preferably in seconds.
- Hydraulic parameters from the modelled soil: the hydraulic diffusivity (*D0*), the Hydraulic conductivity (*Ksat*), the steady-state infiltration rates (*Iz/Kz*).
- The depth of the water table and of the substrate.
- The mechanical soil properties, soil cohesion (soil capacity to resist to motion), friction angle, weight of soil and weight of water.
- The landscape property: topographic slope.
- The model resolution.

2.2. Global approach

For a specific site, neither remote-sensed nor in-situ data would constrain discrete values for each parameters. It is therefore proposed to run the model using a Monte Carlo sampling schemes on range of possible parameters. In the case of testing the model against in-situ data, ranges of parameters are determined by the different mechanical and hydraulic tests. Calibration is achieved with field observation of ground motions. In the case of testing the model against remote-sensed constraints, the ranges of parameters are suggested from general ranges in the literature and calibrating failures are recorded from InSAR data recording time of ground motion.

2.3. Model output

The main applicable output of the model is a factor of safety for a soil columns that determines

when the slope is subjected to fail. This factor of safety is an absolute value defined by equation 18 in Iverson (2000). It suggests failure when such an ouput is $\Leftarrow 1$. This factor of safety is calculated for a given time and depth of the modelled soil column in regard to the precipitations time series inputted.

2.3.1. For a single simulation

The models outputs table-like `csv` files allowing the visualisation of time series the factor of safety or the pore pressure for a single simulations. It also allow the generation of safety maps for a given time and a given Digital Elevation Model to identify the area at risk.

2.3.2. For Monte-Carlo Runs

The `python` tools control the Monte-Carlo runs of the simulations. Because of the significant number of run involved by the process, all the `csv` outputs are concatenated into a `hdf5`, easily readable from `python`. Time series of factor of safety for all the simulations can be synthesised into a distribution of detected landslide in time. These can be correlated with calibration data (e.g. observed or measured ground motion from InSAR or in-situ monitoring) to determine "successful" simulations. Once "successful" runs have been defined, visualisation of the parameter sets that leaded to failure detection can be used to constrain the model.

Chapter 3. Installation

All the code has been developed, tested and ran on UNIX platform. It requires uses of a C++ compiler and a python 3 environment.

3.1. The code

The code has been developed within the open-source [LSDTopoTools]https://lsdtopotools.github.io/ research software suite. The deliverable has been packaged into a zip file with the final report and contain all the bits of code required for running the analysis.

3.2. Compiling the model

The C++ code requires compilation into an executable before use. This can be done with the GNU g++ compiler, installed in most UNIX and IOS distributions. The Analysis_driver folder in the root folder contains a make file that automates the compilation. It can be simply ran as follow:

```
make -f iverson_model.make
```

It generates an executable file named iverson_model.exe in the same folder and is ready to be used.

3.3. Installing the python package

The python package can be installed in any python environment satisfying the following dependencies: numpy, scipy, pandas, pytables and matplotlib. The package is downloaded with the github repository in the folder python_tools and can be installed in the python environment using pip install . in the python_tools folder.

3.3.1. Optional: Getting a python environment with conda

If it is required to install a python environment from scratch, miniconda environment manager is an easy, clean and open-source solution. It can be downloaded from [here]https://repo.continuum.io/miniconda/. Choose any python 3 installer compatible with your operating system. Once installed it offers a clean method to manage different python environments with different combinations of packages installed. Once install, run the following command to create an environment:

```
conda create -n iverson_python
```

This creates a python environment named iverson_python that need to be activate at each boot of a terminal using the python plotting tools using:

```
conda activate iverson_python
```

The environment has been successfully activated if (iverson_python) is stated at the beginning of

the command line. After the first activation, the dependencies can be installed with:

```
conda install -c conda-forge numpy scipy pandas pytables matplotlib
```

This only need to be run once. All installed packages will remain available at the reactivation of the environment.

Chapter 4. Preprocessing the precipitation data

The most sensible parameter inputted in the model is the precipitation data. This can be from in-situ measurements, global dataset or simulations. The data needs to be in a `csv` (Comma-separated-values) files containing the following columns:

```
duration_s,intensity_mm_sec
...,...
```

Where `duration_s` represent a duration in seconds and `intensity_mm_sec` is the corresponding precipitation intensity in mm per seconds. This last usually requires conversion from common dataset. For example, `15 mm` of rain in 3 hours becomes `15/10800=0.001389 mm/seconds`.

Chapter 5. Run the model for a single soil column

The following instructions describe how to run the code and visualise the outputs for a single soil column simulated.

5.1. Simulation

Once compiled, the C++ executable can be run from the command-line and need to be directed to a parameter file (a text file containing a series of values telling the C++ model how to simulate the soil command). The parameter files can be written as follow in a text file:

```
# This is a parameter file for the Iverson model
# Lines beginning with # are comments and are not read by the softwares

# The read path represent the path where the data will be read
read path: /path/to/file/
# The write path represent the path where the data will be written (Can be the same
than the read path)
write path: /path/to/file/
# Write fname is the prefix of the written files
write fname: 20150711_20160809_filtered

# Name of the csv file containing the preprocessed precipitation data
rainfall_csv: preprocessed_data.csv

# Parameters for the models
D_0: 0.000005
K_sat: 0.00000005
d: 2
Iz_over_K_steady: 0.2
alpha: 0.51
friction_angle: 0.38
cohesion: 12000
weight_of_soil: 19000
weight_of_water: 9800
depth_spacing: 0.1
n_depths: 35


#end of file
```

The executable needs to know the path and the name of the parameter file to read them correctly. From a terminal in the folder containing the compiled executable, the following command will run the model:

```
./iverson_model.exe /path/to/the/parameter/file/ name_of_the_parameter_file.param
```

Note that the parameter file is a text file and can be saved with any extensions (e.g., `.param`, `.driver`). The following parameters can be provided to the model.

Parameter	*Description*
rainfall_csv	the name of the preprocessed `csv` file with the rainfall time series.
D_0	Hydraulic diffusivity
K_sat	Hydraulic conductivity
d	Depth of wate table
Iz_over_K_steady	Steady-state infiltration rate
alpha	Topographic slope
friction_angle	Critical friction angle for the soil
cohesion	Soil cohesion
weight_of_soil	Volumic weight of the soil
weight_of_water	Columic weight of the water
depth_spacing	Model resolution
n_depths	Number of discretisation

5.2. Model outputs

The C++ model outputs 6 `csv` files containing the evolution of various characteristics of the soil column. Four csv files contain the time series of the different components of the Factor of Safety (FS) functions of time and depth. The independent components of FS are described in equations 28 (a,b and c) and their time series are in files `XXX_time_series_depth_Fcomp.csv` , where `XXX` is the `write fname` in the parameter file and `Fcomp` the corresponding component. It also output the final FS as well as the evolution of the transient pore pressure (Psi) in the file `time_series_depth_Psi.csv`. All of these files have the same structure:

- The first column is Depth
- All the following columns are the corresponding values for each simulated time, where the first row is time.

  === Spatial analysis

Each model run therefore allow the tracing of time at risk for a single soil column through time. Once a time has been identify, a raster can be inputed to the model to generate a risk map. The raster, assumed to be represented by the simulations, will be downsampled, the slope will be calculated and factor of safety is calculated for each soil column. The following lines can be added to the parameter file to enable the spatial analysis:

```
full_1D_output: false
spatial_analysis: true
reload_alpha: false
resample_slope: true
resample_slope_res: 6
topo_raster: insar_area_PP
# alpha_to_reload:
polyfit_window_radius: 1
n_threads: 4
time_of_spatial_analysis: 19818000
```

Where `full_1D_output: false` disables other ouputs, `spatial_analysis: true` enables the analysis, `reload_alpha: false` enable/disable the reloading of previously calculated slope map, `resample_slope` determines if the raster is down-sampled before slope calculation (for speed purposes) with a new resolution detailed with `resample_slope_res`, `topo_raster` is the name of the raster witout its extension (it has to be an ENVI bil file), `polyfit_window_radius` is the precision of slope calculation, `n_threads` is the number of threads to use for multithreading and `time_of_spatial_analysis` is the identify time on the precipitation time series. It produces a raster with the factor of safety for each soil column.