NGI Technical note

To:	Norges Vassdrags- og energidirektorat
Attn.:	Aart Verhage
Copy to:	
Date:	2018-12-11
Revision no./Rev.date:	0 /
Document no.:	20170131-10-TN
Project:	SP 4 FoU Snøskred
Project manager:	Dieter Issler
Prepared by:	Dieter Issler, Zhongqiang Liu
Reviewed by:	Malte Vöge

Design Alternatives for a Tool for Probabilistic Run-Out Calculations With MoT-Voellmy

Contents

Review and reference page

NORWEGIAN GEOTECHNICAL INSTITUTE
NGI.NO

Main office
PO Box 3930 Ullevaal St.
NO-0806 Oslo
Norway

Trondheim office
PO Box 5687 Torgarden
NO-7485 Trondheim
Norway

T 22 02 30 00
F 22 23 04 48
NGI@ngi.no

BIC NO. DNBANOKK
IBAN NO26 5096 05 01281
ORGANISATION NO.
958 254 318MVA

ISO 9001/14001
CERTIFIED BY BSI
FS 32989/EMS 612006

1 The problem

As part of WP 2 – Analyses of the joint Natural Hazards GBV project HARM1 (project number 20180069), it was decided to develop a tool for carrying out probabilistic run-out calculations. This task was later moved to FoU Snøskred due to lack of resources. First discussions indicated that simulations with a fast, quasi-3D model like MoT-Voellmy should be feasible if parallelization of the most time-consuming parts is used. Moreover, NAKSIN already contains many features that will be needed in such an application. This suggested writing a Python script that prepares the input for each simulation, then calls MoT-Voellmy, and finally counts the number of hits in each cell of the computational grid.

Closer scrutiny of such a solution reveals quickly, however, that the Python script and MoT-Voellmy interact through ASCII files that must be written by the Python script and read by MoT-Voellmy. Also, MoT-Voellmy computes curvatures for the entire computational domain every time it is run, but in this application the terrain is the same for all simulations. It is therefore of interest to consider alternative approaches that might be more efficient. Below, the originally envisaged approach (Method A below) is compared to two other possible approaches B and C, and finally conclusions are drawn.

2 Standard MoT-Voellmy called from Python script (Method A)

This is the originally envisaged solution: The Python script collects all necessary information about the avalanche to be simulated and about the Monte-Carlo procedure. In parallel threads, it then

- creates command files for MoT-Voellmy,
- prepares the release-depth and friction-coefficient raster files for each simulation according to the specified statistical descriptions (i.e. mean, standard deviation and probability distribution) of these input data,
- runs multiple instances of MoT-Voellmy and stores the calculated results from each simulation,
- blocks the probability-distribution array for other processes and modifies it according to the simulation results from MoT-Voellmy.

When all simulations are done, a simple division of the hit-count raster by the number of years represented by the Monte-Carlo trials produces the final map file showing the spatial distribution of probability. To reduce the number of simulations and the significant time demands on calculations, the Latin Hypercube Sampling (LHS) method could be used,

Advantages:

\+ Minimal programming effort, much can be adapted from NAKSIN.

+ Most of the work can be parallelized in a simple way.
+ Much of the framework can be reused for similar codes using other simulation engines (RAMMS, BingClaw, ...).

Disadvantages:

– Considerable duplicate effort (e.g., computation of curvatures inside MoT-Voellmy for each simulation)
– Parameter passing between Python and C via ASCII files is inefficient.

3 MoT-Voellmy as a shared library called by a Python script (Method B)

1. Strip MoT-Voellmy of all reading of input files and writing of output files. Instead, the parameters read from the command file for the usual version are passed as parameters during the function call and the h_{max} values are returned as an array pointer. Take out the calculation of curvature and use the curvature passed as a parameter instead.
2. Compile stripped-down MoT-Voellmy as a shared library.
3. Create Python code that does the following:
    - Ask the user for the necessary input files and simulation parameters.
    - Ask the user for the mean values and standard deviations of the stochastic parameters as well as the number of Monte Carlo trials (prepared according to the LHS method).
    - Read all input raster files into arrays and compute curvature as an array.
    - Use the ctypes module to convert these parameters and arrays into C-compatible objects.
    - Create the values of the stochastic variables and call the MoT-Voellmy module with the right parameters.
    - Receive the h_{max} output array from all simulation runs and count the number of hits per cell.
    - Output a raster file with the spatial probability distribution.

Advantages:

+ Duplication of computational work (e.g., calculation of curvature) is largely avoided.
+ The full functionality of MoT-Voellmy is retained.
+ The new parts should be relatively easy to program in Python.
+ The Python code is easy to parallelize, with one instance of MoT-Voellmy running per process. Little work is left outside the parallelized code section.

Disadvantages:

– Need to learn how to use ctypes.

– Some of the set-up work for each simulation may be less efficient in Python than C-code.

4 A new version of MoT-Voellmy including Monte-Carlo simulations (Method C)

1. Make the computational loop of MoT-Voellmy's main function into a new subroutine.
2. Extend that subroutine to update the shared array that counts hits to each grid cell.
3. Write a new main function that generates multipliers for the friction coefficients and release depths according to the specified statistical descriptions and starts the MoT-Voellmy simulations with the correct parameter values in parallel processes.

Advantages:

+ Is expected to generate the most efficient code.
+ A single, compact executable, no need to install Python.

Disadvantages:

– Significant changes to the MoT-Voellmy code required.
– Author has no experience with parallelizing C-code yet.

5 Preliminary conclusion

Performance-wise, the three variants are clearly ranked as A < B < C. Some conjectures concerning the degree of performance differences and the reasons for them can be made based on experience from NAKSIN runs:

- The Python module subprocesses.py makes it relatively painless to set up parallelized computation, but the bookkeeping needed for this seems to be time-consuming.
- The Python code preparing the input data may consume up to an order of magnitude more time than the MoT-Voellmy run in the case of small avalanches in much larger computational domains.
- In the case of large avalanches running for a long time with short time steps, MoT-Voellmy may use one to two orders of magnitude more time than the Python code. Also, in the first case, the preparatory work within MoT-Voellmy consumes a fair fraction of the total computing time.

Accordingly, in poorly set up problems, an optimized code of type C might require a hundred times less computing time than the type A code. A well-programmed type-B code will probably be closer in performance to a type-C code.

When it comes to development time and uncertainty in the outcome, the ordering is again A < B < C, but this time in favor of approach A. There, MoT-Voellmy is not modified, and all ingredients in the Python code – like writing command and raster input files and running the MoT-Voellmy simulations in parallel – are tested and available in NAKSIN.

The main stumbling block of approach B is the need to correctly apply the ctypes module in both the Python and the C code. It is somewhat unpredictable how difficult this will be – the module is extensively documented, but it is likely that there are some subtleties that need to be figured out, as has been the case when the Python code of NAKSIN was parallelized with subprocess.py. Even disregarding this risk, while not fundamental the changes in MoT-Voellmy are still considerable.

Approach C does not use ctypes, as the entire code would be written in ISO-C with extensions for parallel processing. A search on the Internet arrived at OpenMP as a candidate API or at ISO-C11 as a recent standard for C that has support for multiprocessing. Note that OpenMP supports parallelization on a multiple-core single node, i.e., a single PC or workstation or even a supercomputer, but not a cluster of different machines. A single machine is presently the most realistic scenario for applications at NGI. For distributed nodes, a Message Passing Interface would have to be used.

OpenMP appears to be reasonably easy to use, but the programmer needs to understand precisely how threads that work in parallel, e.g. in a loop, interact with each other indirectly through access to shared resources. The efficient parallel running of MoT-Voellmy simulations in NAKSIN indicates, however, that this should not be a major problem.

At this design stage, approach B appears the least attractive because it achieves only a moderate (albeit significant) speed-up compared to approach A but requires mastery of ctypes. The development effort is probably similar to that of approach C.

With this, the choice is mainly between fast and easy development in approach A on the one hand and a larger and less predictable development effort in approach C that promises a potentially huge gain of speed. In the present situation, this essentially means that approach A must be chosen if the task should be completed within 2018, but approach C holds more promise for a usable tool, provided development can be scheduled for early 2019.

Kontroll- og referanseside/ Review and reference page

Dokumentinformasjon/*Document information*		
Dokumenttittel/*Document title* Design Alternatives for a Tool for Probabilistic Run-Out Calculations With MoT-Voellmy		**Dokumentnr./*Document no.*** 20170131-10-TN
Dokumenttype/*Type of document* Teknisk notat / Technical note	**Oppdragsgiver/*Client*** Norges vassdrags- og energidirektorat	**Dato/*Date*** 2018-12-11
Rettigheter til dokumentet iht kontrakt/ *Proprietary rights to the document according to contract* NGI		**Rev.nr.&dato/*Rev.no.&date*** 0 /
Distribusjon/*Distribution* FRI: Kan distribueres av Dokumentsenteret ved henvendelser / FREE: Can be distributed by the Document Centre on request		
Emneord/*Keywords* Probabilistic hazard mapping, MoT-Voellmy, parallel processing, Python, ISO-C		

Stedfesting/*Geographical information*	
Land, fylke/*Country* —	**Havområde/*Offshore area*** —
Kommune/*Municipality* —	**Feltnavn/*Field name*** —
Sted/*Location* —	**Sted/*Location*** —
Kartblad/*Map*	**Felt, blokknr./*Field, Block No.*** —
UTM-koordinater/*UTM-coordinates* **Zone:** — **East:** — **North:** —	**Koordinater/*Coordinates*** **Projection, datum:** — **East:** — **North:** —

Dokumentkontroll/*Document control*
Kvalitetssikring i henhold til/*Quality assurance according to NS-EN ISO9001*

Rev/ *Rev.*	**Revisjonsgrunnlag/*Reason for revision***	**Egenkontroll av/ *Self review by:***	**Sidemanns-kontroll av/ *Colleague review by:***	**Uavhengig kontroll av/ *Independent review by*:**	**Tverrfaglig kontroll av/ *Interdisciplinary review by:***
0	Original document	2018-12-11 Dieter Issler	2018-12-17 Zhongqiang Liu		2018-12-18 Malte Vöge

Dokument godkjent for utsendelse/ *Document approved for release*	**Dato/*Date***	**Prosjektleder/*Project Manager***
	18 December 2018	Dieter Issler

NGI (Norwegian Geotechnical Institute) is a leading international centre for research and consulting within the geosciences. NGI develops optimum solutions for society and offers expertise on the behaviour of soil, rock and snow and their interaction with the natural and built environment.

NGI works within the following sectors: Offshore energy – Building, Construction and Transportation – Natural Hazards – Environmental Engineering.

NGI is a private foundation with office and laboratories in Oslo, a branch office in Trondheim and daughter companies in Houston, Texas, USA and in Perth, Western Australia

www.ngi.no

NGI (Norges Geotekniske Institutt) er et internasjonalt ledende senter for forskning og rådgivning innen ingeniørrelaterte geofag. Vi tilbyr ekspertise om jord, berg og snø og deres påvirkning på miljøet, konstruksjoner og anlegg, og hvordan jord og berg kan benyttes som byggegrunn og byggemateriale.

Vi arbeider i følgende markeder: Offshore energi – Bygg, anlegg og samferdsel – Naturfare – Miljøteknologi.

NGI er en privat næringsdrivende stiftelse med kontor og laboratorier i Oslo, avdelingskontor i Trondheim og datterselskaper i Houston, Texas, USA og i Perth, Western Australia.

www.ngi.no

OSLO
TRONDHEIM
HOUSTON
PERTH

NORWEGIAN GEOTECHNICAL INSTITUTE
NGI.NO

Main office
PO Box 3930 Ullevaal St.
NO-0806 Oslo
Norway

Trondheim office
PO Box 5687 Torgarden
NO-7485 Trondheim
Norway

T (+47)22 02 30 00
F (+47)22 23 04 48
NGI@ngi.no

BIC NO. DNBANOKK
IBAN NO26 5096 0501 281
ORGANISATION NO.
958 254 318MVA