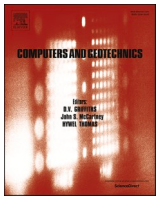
Computers and Geotechnics 153 (2023) 105078

Contents lists available at ScienceDirect

Computers and Geotechnics

journal homepage: www.elsevier.com/locate/compgeo



ORIN-3D – A new model for efficient simulation of landslide motion on a GPU using CUDA

Jordan Aaron [a,b]

[a] Chair of Engineering Geology, Geological Institute, ETH Zurich, Zurich, Switzerland
[b] Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Birmensdorf, Switzerland

ARTICLE INFO

Keywords:
Landslide runout modelling
Rock avalanche
Debris avalanche
Smooth particle hydrodynamics
Graphical processing unit
CUDA

ABSTRACT

Extremely-rapid flowlike landslide are a major hazard in many parts of the world, and managing their risk requires an understanding of the mechanisms that drive motion, as well as reliable predictions of their potential destructiveness. Numerical runout models are one tool that can be used for both these applications, however their utility is presently limited by their computational runtime. In the present work, a new depth-averaged, smooth particle hydrodynamics based model is implemented to run on a graphical processing unit. The new implementation provides a speedup of over two orders of magnitude, compared to a commonly used CPU based runout model. The new model has been validated, and then is used to back-analyse the Johnsons Landing Landslide. It is shown that increasing model resolution results in an accurate simulation of complex topographic interactions between the flowing landslide and the surface topography. The new model runs on commercially available GPU's, and should therefore be useful for both researchers and practitioners seeking to understand and quantify landslide hazard and risk.

1. Introduction

Extremely rapid flow-like landslides, which include rock avalanches and debris flows, are among the most catastrophic geophysical flows. These events threaten people living and traveling in mountainous areas, and effective risk management requires accurate forecasts of their motion. Numerical models are one tool that can be used to forecast flow-like landslide motion, however long model runtimes and limited model resolution restrict the calibration and prediction accuracy of these models.

In recent years, widely available and inexpensive graphical processing units (GPUs) have revolutionized many areas of computing (NVIDIA Corporation, 2021). GPUs, which feature thousands of individual computing cores, can provide orders of magnitude increase in the speed at which certain parallelizable algorithms run (NVIDIA Corporation, 2021). Implementing numerical models for simulating landslide motion on a GPU could dramatically increase the efficiency of these algorithms. The present work implements and tests a numerical runout model on a GPU.

Many numerical models that can simulate landslide motion have been proposed in the literature (e.g.Ho et al., 2018). These models can be either continuum or discontinuum models (Hungr et al., 2007). Discontinuum models have recently gained more prevalence, and some have been implemented on a GPU (Song et al., 2017; Wang et al., 2017; Xu et al., 2021). Continuum models provide an alternative to discontinuum models, and tend to be more computationally efficient, and the use of GPU's has recently enabled 3D runout analysis using continuum models (Li et al., 2020; Peng et al., 2019, 2022). Many continuum runout models have been implemented (Ho et al., 2018; Hungr et al., 2007; McDougall, 2017), and these models differ primarily in the form of the equations of motion (Lagrangian or Eularian) and the numerical solution method, as well as the methods used to simulate internal pressure gradients, entrainment, and centripetal accelerations. Two benchmarking exercises have revealed that simulation results using different continuum models tend to be similar (Ho et al., 2018; Hungr et al., 2007).

In the present work a depth-averaged, Lagrangian runout model is implemented to run on a GPU. This model leverages the highly parallel GPU architecture to decrease model runtime by two orders of

E-mail addresses: jordan.aaron@erdw.ethz.ch, jordan.aaron@wsl.ch.

https://doi.org/10.1016/j.compgeo.2022.105078
Received 25 February 2022; Received in revised form 10 August 2022; Accepted 6 October 2022
Available online 25 October 2022

magnitude, and increase maximum model resolution by two orders of magnitude. Firstly, the implementation of the new model is described, and then the new model is validated. Secondly, a runtime comparison between the new model and a CPU-based runout model is provided. Finally, the new model is applied to back-analysis using simulation configurations that were previously unattainable.

2. Model Overview and CUDA Implementation

As described more fully in McDougall & Hungr (2004), simulation of extremely-rapid flowlike landslides requires that a model can simulate motion over complex topography, while accounting for unique features of flow-like landslide motion, which include centripetal accelerations and strain dependent stresses. Using a Lagrangian form of the equations of motion, combined with meshfree numerical methods, can meet these requirements (e.g. McDougall & Hungr, 2004), and the numerical model presented here is based on Dan3D, a depth-averaged, Lagrangian runout model (Hungr, 1995; McDougall & Hungr, 2004).

Meshfree methods, such as Smooth Particle Hydrodynamics (SPH), which is the method used in the present work (Monaghan, 1992), can suffer from long model runtimes. This is because of the interpolation procedure which is at the core of SPH. SPH divides the landslide mass

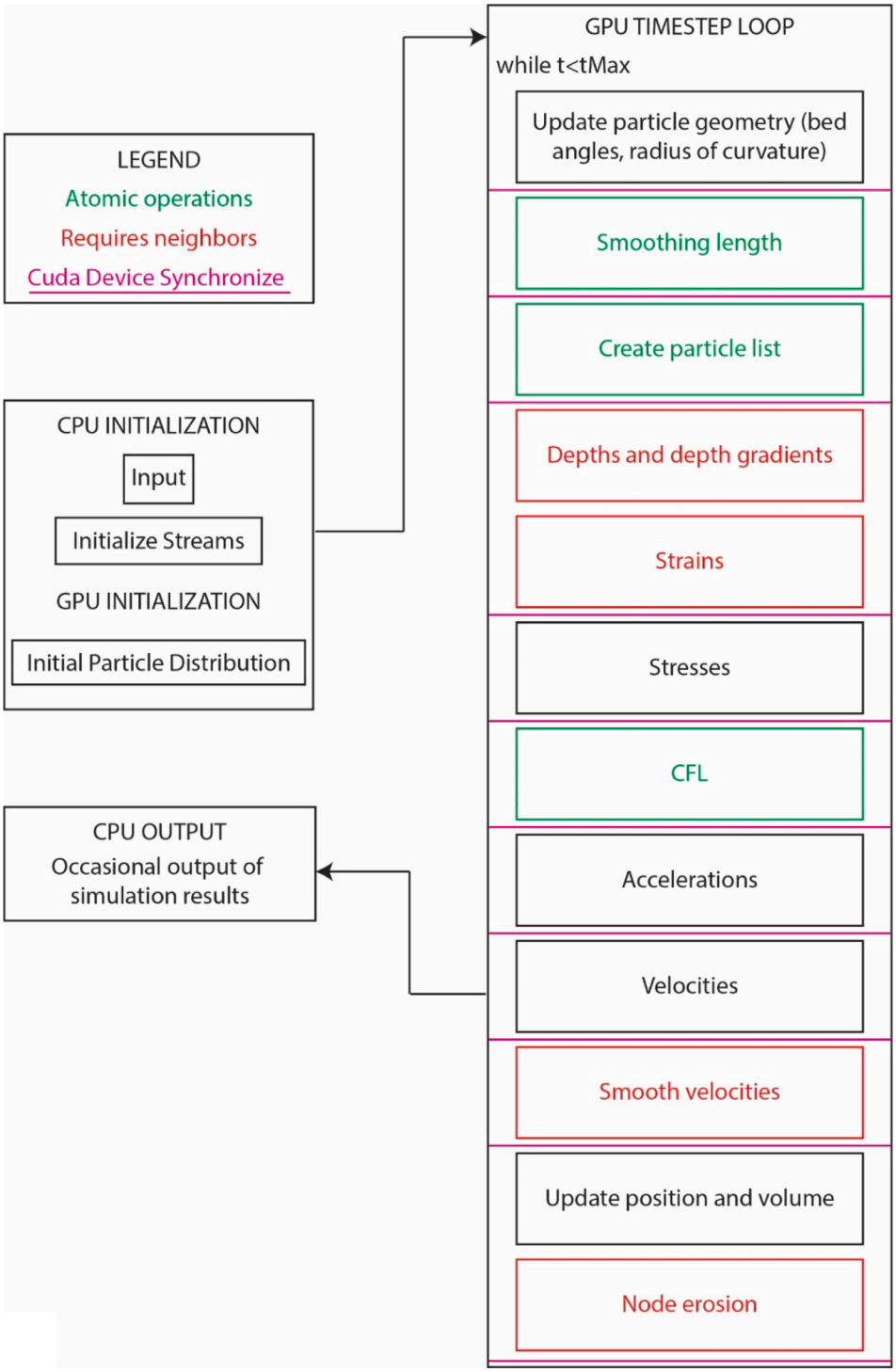


Fig. 1. Flowchart of the algorithm, showing GPU optimizations. Modified after McDougall, (2006).

into a collection of smooth particles, each with a finite volume, and interpolates the quantities of interest (such as flow depth, velocity and erosion depth) based on its value at these particle locations. This interpolation is based on a user selected kernel, whose width is controlled by a 'smoothing length' parameter (Monaghan, 1992), and requires a computationally demanding nearest neighbor search.

Many previous authors (e.g. Domínguez et al., 2013; Goozee & Jacobs, 2003), as well as code profiling of a CPU implementation of an SPH runout model, show that the SPH algorithm spends the vast majority of its computational time performing the nearest neighbor search. Efficient and parallelizable neighbor search algorithms exist, which makes a GPU based SPH runout model a potentially powerful tool for overcoming runtime limitations (e.g. Domínguez et al., 2013; Goozee & Jacobs, 2003; Peng et al., 2019).

Graphical processing units (GPUs) are hardware units that feature a massively parallel architecture. Many detailed descriptions of this hardware exist (NVIDIA Corporation, 2021), and only a brief overview of a few relevant details are presented here. GPUs feature a large number of computing cores, which can execute instructions in parallel. GPUs attain high performance by computing many tasks in parallel, and it is thus important to keep GPU occupancy high. However, global memory accesses are slow, so algorithms written for the GPU should coalesce memory accesses (NVIDIA Corporation, 2021). This has the advantage of both increasing the cache hit rate, and enabling multiple reads per global memory access. In the present work, an NVIDIA GPU is used, and the algorithm is written in CUDA.

The new model, termed *parallel simulatOR of landslIde motioN over 3D terrain*, or ORIN-3D, runs entirely on the GPU, with the exception of model initialization and data output. A flowchart summarizing the algorithm is presented in Fig. 1. Fig. 1 is modified after McDougall (2006), who presented a CPU implementation of an SPH based runout model. On Fig. 1, functions that require a nearest neighbor search are highlighted in red. A key aspect to note about this nearest neighbor search is that, when Gaussian kernels are used (as is done in the present work), only particles within 3 smoothing lengths will appreciably contribute to the sums so only these particles need be evaluated.

In the present work, three main GPU specific code optimizations have been implemented, summarized in Table 1. The first is a parallel neighbor search, based on that presented by Goswami et al. (2010), Domínguez et al., (2013) and Goozee & Jacobs (2003), which uses Z-order indexing of particle locations (Goswami et al., 2010; Morton, 1966). Z-order indexing is an efficient method to map two-dimensional particle coordinates to one dimensional numbers, while preserving spatial locality. Thus, neighboring particles will have similar Z-index values, which increases the cache hit rate in the present algorithm. An overview of this neighbor search algorithm is shown in Fig. 2. First, a kernel is launched where each thread calculates a Z-index for one of the particles. The particles are then sorted based on Z-index, using Radix sort as implemented by THRUST (Bell and Hoberock, 2012). A kernel is then launched to determine the minimum sorted particle indices in each block, as well as the number of particles in each block. Finally, all particles in a block can be scanned through by starting at the minimum particle, and looping through the number of particles contained in the block.

One advantage of using a Z-Index neighbor search algorithm is that neighboring particles are located nearby each other in memory. To take advantage of this, the second optimization (Table 1) is to sort all particle quantities (such as position, velocity and volume) according to Z-index following the creation of the nearest neighbor arrays. This increases memory coalescence (and therefore cache hit rate), as all particles within a thread block access similar locations in global memory. As part of the second optimization (#2, Table 1), global memory accesses are limited by using the Float4 data type, and grouping variables that are accessed together, such as particle position (x,y,z) and flow depth (h). Thus, one global memory read can access all quantities for a given particle. Additionally, as shown on Fig. 1, some functions can be evaluated in parallel, in particular the functions used to calculate flow depths and particle strains. In the fully optimized version of the code, these functions are called in parallel.

Table 1
Optimizations made in the CUDA runout model.

Optimization #	Description
1	Implementation of Z-order based neighbor search
2	Sorting and grouping of variables into the Float4 data type
3	Maximize streaming multiprocessor occupancy by optimizing the number of blocks and threads used by each kernel call

The final optimization is aimed at maximizing streaming multiprocessor occupancy. The nearest neighbor search requires calculating the distance between each particle and all other particles within nine surrounding cells (Fig. 2). In Optimization #3, the nine cells used for comparison are launched as separate thread blocks, and each of these blocks is given a specified number of threads for each particle within the block. Thus, if the block is meant to evaluate 32 particles, and is given 16 threads per particle, the block is launched as a 2D grid of 512 total threads. In this configuration, 16 threads are assigned to one particle to evaluate all neighbor interactions, and the results are aggregated using atomic operations. At the block level, shared memory is used to aggregate the neighbor results, whereas all information computed by individual blocks (which correspond to cells (Fig. 2)) are accumulated using atomic operations on global memory. The number of particles evaluated by each block, as well as the parallel threads per block, need to be optimized to trade off streaming multiprocessor occupancy, the cache hit rate and serialization resulting from atomic operations.

Certain operations (green boxes on Fig. 1), including calculating the smoothing length, creating the particle list and evaluating the Courant–Friedrichs–Lewy (CFL) condition to determine the timestep, require the minimums, maximums and counts of certain quantities. To avoid race conditions resulting from parallel evaluation of particle quantities, these operations are implemented using atomic operations within CUDA.

3. Model Validation

The new CUDA based runout model has been validated using a subset of the test cases presented in McDougall and Hungr (2004). These include an analytical solution to the dam break problem, generalized to include basal friction and a sloping base (Mangeney et al., 2000), as well as lab experiments performed by Gray et al. (1999). For the dam break problem, and following McDougall (2006), the algorithm was modified to neglect momentum in the y-direction, and a one dimensional Gaussian function was used as the interpolation kernel. Simulations were conducted with basal inclinations of 0° and 30°, and friction angles of 0° and 20°, respectively. 1,000,000 particles were used for both validation cases. Only results from the simulations with a 30° basal inclination and 20° friction angle are presented here.

In the Gray et al. (1999) lab experiments, dry quartz chips were released down a chute inclined at 40°, which then went through a smooth transition before depositing on a flat plane. Due to the smooth transition, centripetal accelerations at the base of the setup approximate what may occur in natural cases, and this experiment thus tests the efficacy of the centripetal acceleration algorithm.

The results of the dam break validation are shown on Fig. 3A. The match between the numerical and analytical solutions is high. The comparison between the Gray et al. (1999) experiments and ORIN-3D results are shown on Fig. 3B. The correspondence is again high. As can be seen in Fig. 3B, the model diverges somewhat from the experiments at t = 1.5 s and 2 s. A similar phenomenon was observed in McDougall & Hungr (2004), and may be related to variations of the dynamic friction angle of the material, as discussed in literature (Gray et al., 1999; Wieland et al., 1999).

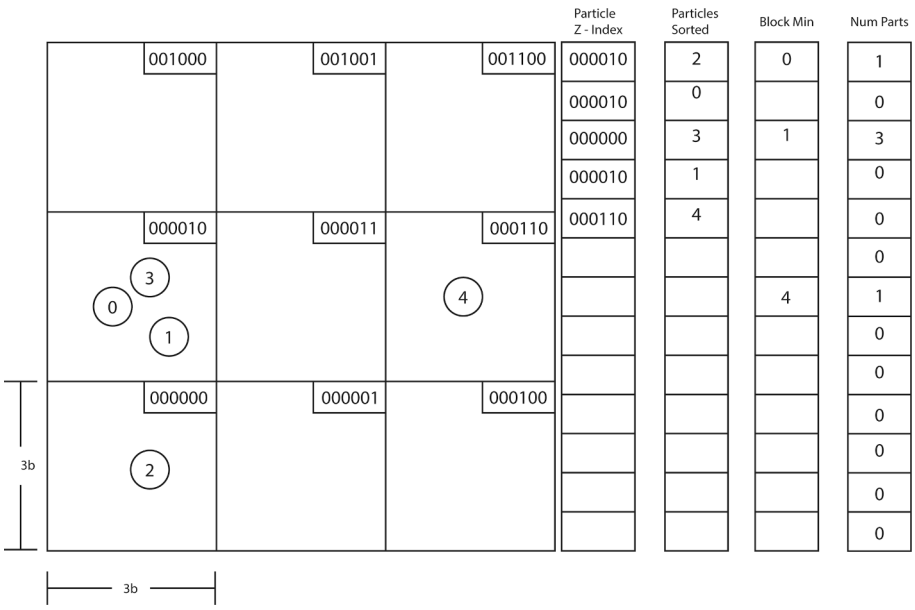


Fig. 2. Explanation of the particle nearest neighbor detection algorithm, modified after (Domínguez et al., 2013).

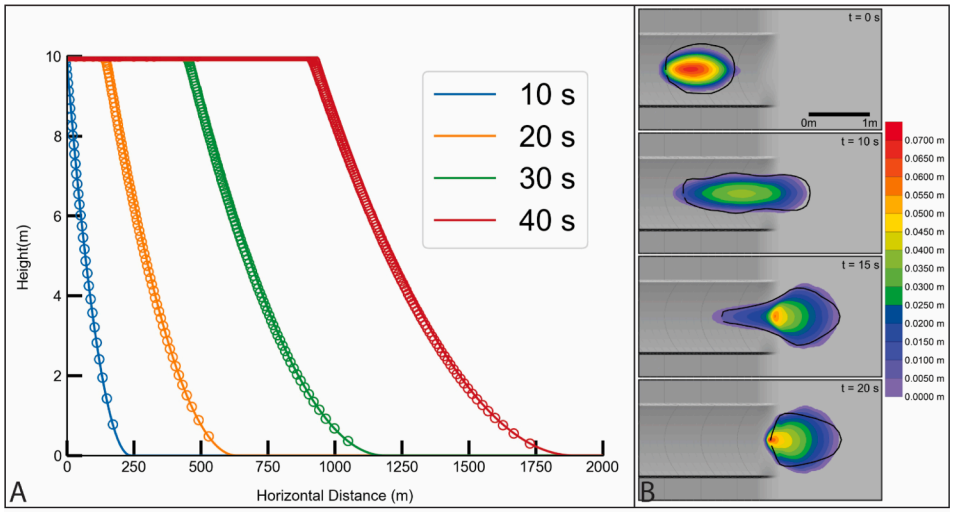


Fig. 3. A) Results of the dam break analysis over a 30° sloping bed with a basal friction angle of 20°. The solution is given at ten second intervals, with the line representing the analytical solution, and the dots the numerical solution. For clarity, only the spreading of the front of the simulated dam break is shown, and not the trailing material present at horizontal distances less than 0. B) Results of simulating the Gray et al. (1999) experiments. The black outline shows the measured extent of the experimental flow, and the contours show the simulated flow depth.

4. Runtime Evaluation

The performance of the new numerical model was assessed by comparing the model runtimes of ORIN-3D to Dan3D, which is implemented to run on a CPU. The CPU used in this evaluation was an Intel-i9, and the GPU was an Nvidia 2080Ti. It should be noted that the CPU simulation algorithm has not been optimized in the same manner as the GPU algorithm, so this comparison cannot be used to generally assess the relative performance of CPU vs GPU SPH models. However, the specific CPU model (Dan3D) is commonly used by researchers and practitioners, so the runtime comparison is still relevant in assessing the speedup of the present model compared to a commonly used one.

For this benchmarking exercise, the Nomash River case history was selected, as this case history features entrainment of path material and strongly curving topography (McDougall & Hungr, 2005), so it utilizes all the features of both Dan3D and the newly developed CUDA algorithm. This event, shown in Fig. 4A, occurred in 1999 on Vancouver Island, British Columbia. It involved an irregular failure (collapse) of approximately 300,000 m^3 of crystalline limestone (McDougall & Hungr, 2005). Following failure, the mass overran a thick colluvial deposit, entraining a further 360,000 m^3. The moving fragments of rock and entrained material then entered into a channel, and followed a sinuous path downstream, superelevating three times before deposition (Hungr & Evans, 2004; McDougall & Hungr, 2005). To analyze this case, the same parameterization as that used by McDougall & Hungr (2005) was used. A single Voellmy material was used for the entire travel path, and entrainment was enabled in the zone containing colluvium ('entrainment zone' on Fig. 4A). In this zone, an entrainment rate of $1.9e^{-3}$ m^{-1} was used (McDougall & Hungr, 2005). The simulation output obtained when using these parameters is shown on Fig. 4B.

For the runtime evaluation, a suite of simulations was performed which varied the included code optimizations and number of particles (N). As summarized in Table 1, three main code optimizations have been implemented. The influence of these optimizations is assessed based on the change of simulation runtime achieved for each successive optimization. Two sets of baseline simulations are presented, which include model runtimes for CPU and GPU algorithms that implement an $O(N^2)$ nearest neighbor search, and do not coalesce and minimize global memory access. The next sets of results present simulation runtimes using progressively more optimized code, where the optimization

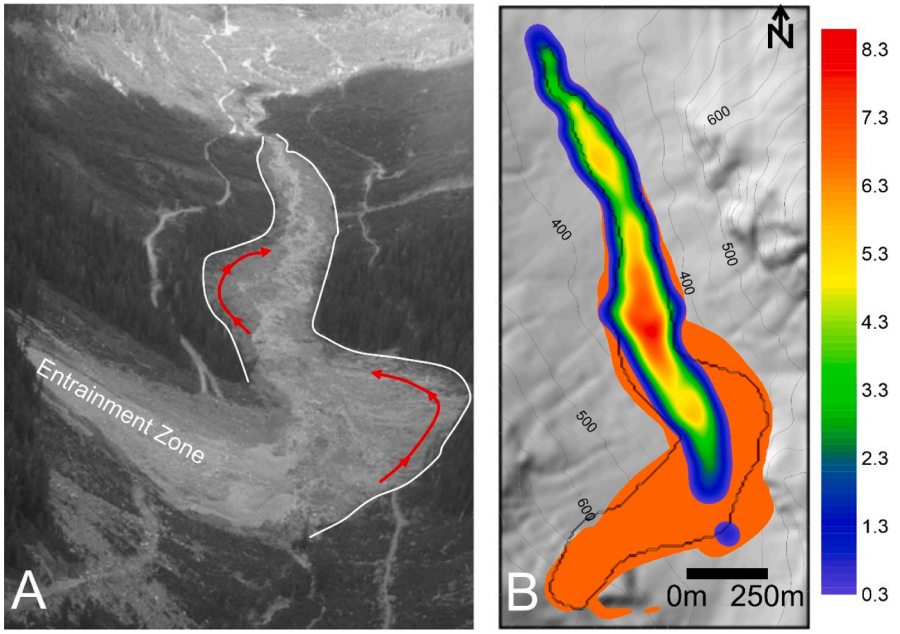


Fig. 4. A) Overview of the Nomash River rock avalanche. Modified after McDougall and Hungr (2005). and Aaron and McDougall (2019), original photo: Dana Ayotte. B) Output of the GPU numerical simulation using the input parameters detailed in McDougall and Hungr (2005). The black outline shows the observed impact area, the orange shape shows the simulated impact area, and the contours show simulated final deposit depth. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

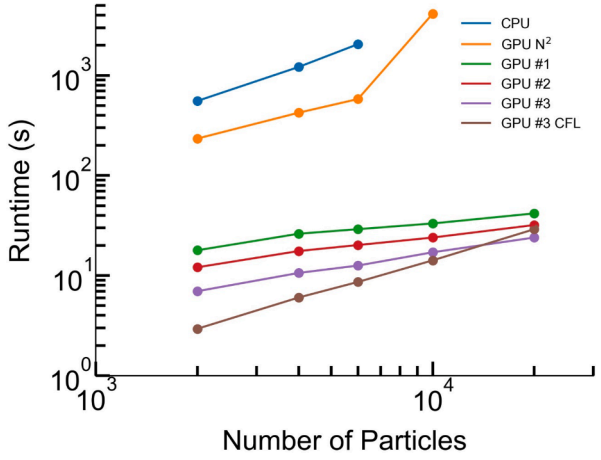


Fig. 5. Runtime comparison for various code optimizations, compared to a CPU runout model. The numbers refer to the optimization numbers given in Table 1, and CFL refers to simulations performed with an adaptive timestep based on the CFL condition.

numbers refer to Table 1.

The results of the performance comparison are shown on Fig. 5. The suite of simulations labelled 'GPU N^{2}' implements an algorithm that is identical to the CPU model, but that does not leverage many of the unique features of the GPU. Nevertheless, Fig. 5 shows an immediate speedup (~4x) is gained by switching to the GPU. Successive code optimizations provide more dramatic speedups, with the fully optimized code running about 115 times faster for a 4,000 particle simulation. Additionally, increasing the number of particles appears to result in a logarithmic increase in runtime for the optimized GPU simulations. It should be noted that the timestep was kept constant for all simulations except 'GPU #3 CFL' on Fig. 5. Comparing this simulation to the others presented on Fig. 5 shows that larger timesteps are used for fewer particles, resulting in a runtime speedup, however for larger numbers of particles, smaller timesteps are required. This small timestep becomes the main runtime constraint when the number of particles are increased.

5. High Resolution Simulation of the Johnsons Landing Landslide

In order to demonstrate the new results that can be achieved with the optimized GPU model, an example high-resolution simulation of the Johnsons Landing landslide has been performed. This landslide occurred in 2012 on the shores of Kootenay Lake, British Columbia, Canada, and involved the failure of about 320,000 m^3 of debris, which bulked to a volume of ~ 375,000 m^3. As shown on Fig. 6, the failed material rapidly traversed the upper reaches of Gar Creek, before reaching a 70° bend in the channel. The failed material avulsed at this bend, flowing onto the Johnsons landing bench and killing four people. A detailed back-analysis of this event was presented in Aaron et al. (2020), who found that the material was likely moving in an undrained condition. The simulations performed by Aaron et al. (2020) were limited to 4,000 particles, and underestimated the large volume of debris that deposited on the bench, and overestimated the volume that went down the channel. This was potentially due to limited model resolution. In the present work, the same parameterization as Aaron et al. (2020) is used, but 1,000,000 particles are used to discretize the failed mass.

The results of the 1,000,000 particle simulation are shown on Fig. 7. Fig. 7 shows the material travelling down the channel and super-elevating twice before encountering the 70° bend. The increased number of particles well resolves all of these complex features. Further, the material avulses from the channel, and deposits on the bench, resulting in deposit depths that are close to that inferred from the field observation (compare inset on Fig. 7 to simulated depths). Finally, the small volume of material that ran down the lower channel and deposited on the Gar Creek Fan (Fig. 6) is well resolved.

6. Conclusions

The GPU based runout model presented here provides an increase in efficiency of over two orders of magnitude, when compared to a commonly used, CPU based runout model. The new model was validated, and then the influence that increasing model resolution has on simulation results was assessed. It was shown that increasing model resolution enables the model to capture complex features, such as superelevation, channel avulsion, and branching flow. The new model thus provides the foundation for more rigorous back-analysis, and efficient probabilistic prediction. As all simulations were run on a desktop

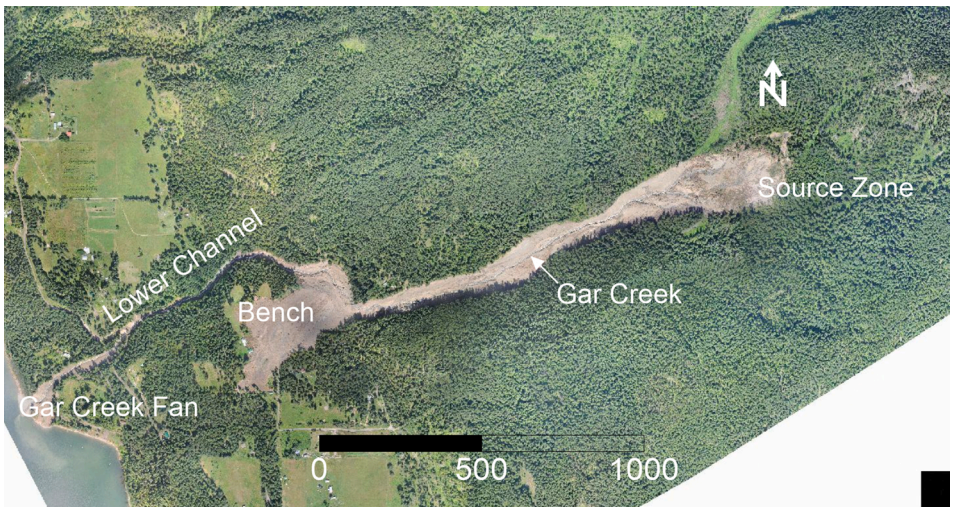


Fig. 6. Overview of the Johnsons Landing landslide showing the main source and deposit features. Figure modified after (Aaron et al., 2020), Image Copyright province of British Columbia.

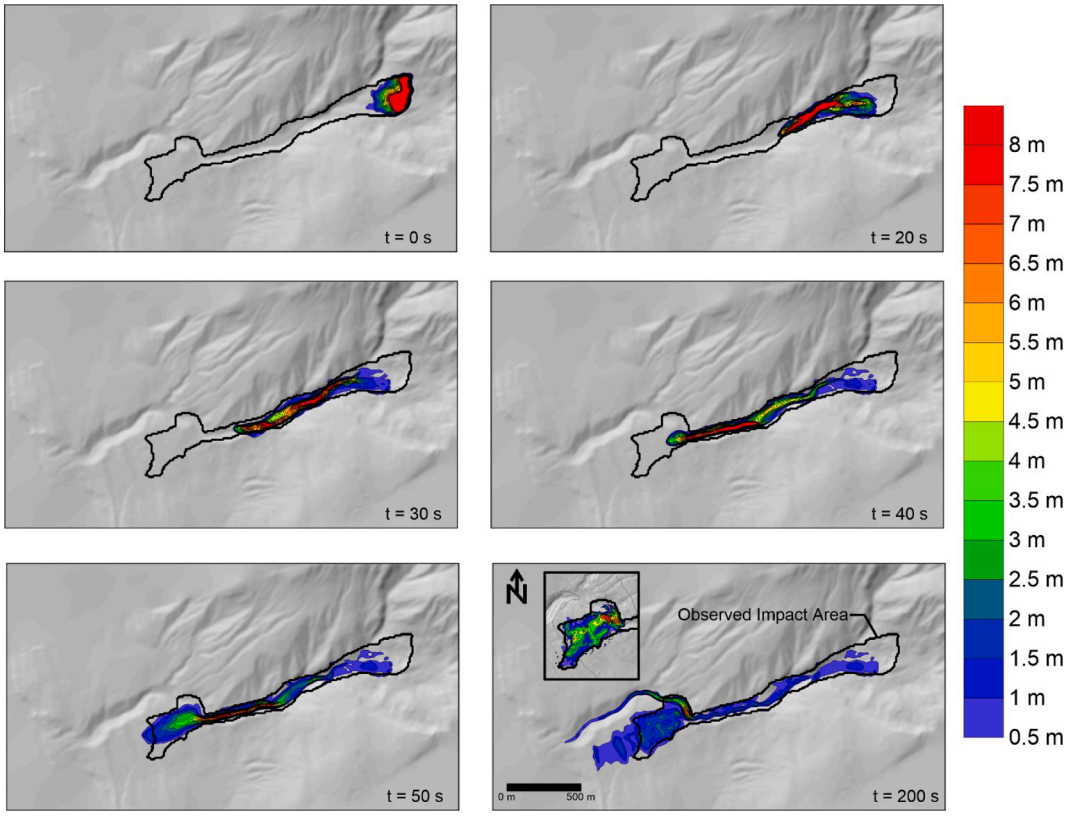


Fig. 7. Simulated landslide depth at various timesteps, using 1,000,000 particles. Inset shows estimated deposit depths based on pre- and post- event topography (further described in Aaron et al., 2020).

computer using a commercial graphics card, the model is useful for both researchers and practitioners.

CRediT authorship contribution statement

Jordan Aaron: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used in this article is already available online

Acknowledgements

Scott McDougall provided excellent feedback on a draft of this manuscript, wrote the original Dan3D source code, and provided many enlightening discussions. The author also thanks Oldrich Hungr for access to the Dan3D source code. The editor and comments from two reviewers helped to improve this manuscript.

References

Aaron, J., McDougall, S., Jordan, P., 2020. Dynamic analysis of the 2012 Johnsons Landing landslide at Kootenay Lake, BC : The importance of undrained flow potential. Canadian Geotechnical Journal 57 (8), 1172–1182. https://doi.org/10.1139/cgj-2018-0623.

NVIDIA Corporation. (2021). *CUDA C ++ Programming Guide* (Issue October).

Domínguez, J.M., Crespo, A.J.C., Gómez-Gesteira, M., 2013. Optimization strategies for CPU and GPU implementations of a smoothed particle hydrodynamics method. Computer Physics Communications 184 (3), 617–627. https://doi.org/10.1016/j.cpc.2012.10.015.

Goozee, R.J., Jacobs, P.A., 2003. Distributed and shared memory parallelism with a smoothed particle hydrodynamics code. ANZIAM Journal 44 (April), 202. https://doi.org/10.21914/anziamj.v44i0.679.

Gray, J. M. N. T., Wieland, M., & Hutter, K. (1999). *Gravity-driven free surface flow of granular*. *November*, 1841–1874.

P. Goswami, P. Schlegel, B. Solenthaler, and R. Pajarola. 2010. Interactive SPH simulation and rendering on the GPU. In Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '10). Eurographics Association, Goslar, DEU, 55–64.

Ho, K., Leung, A., Kwan, J., Koo, R., Law, R., Ho, K., Leung, A., Kwan, J., Koo, R., & Law, R. (2018). *Proceedings of the Second JTC1 Workshop Triggering and Propagation of Rapid Flow-like Landslides* (K. Ho, A. Leung, J. Kwan, R. Koo, & R. Law, Eds.; Issue December).

Hungr, O., 1995. A model for the runout analysis of rapid flow slides, debris flows and avalanches. Canadian Geotechnical Journal 32 (4), 610–623.

Hungr, O., Morgenstern, N. R., & Wong, H. N. (2007). Review of benchmarking exercise on landslide debris runout and mobility modelling. *Proceedings of the International Forum on Landslide Disaster Management*, 775–812.

Hungr, O., Evans, S.G., 2004. Entrainment of debris in rock avalanches: An analysis of a long run-out mechanism. Geological Society of America Bulletin 116 (9–10), 1240–1252. https://doi.org/10.1130/B25362.1.

Li, S., Peng, C., Wu, W., Wang, S., Chen, X., Chen, J., Zhou, G.G.D., Chitneedi, B.K., 2020. Role of baffle shape on debris flow impact in step-pool channel: An SPH study. Landslides 17 (9), 2099–2111. https://doi.org/10.1007/s10346-020-01410-w.

Mangeney, A., Heinrich, P., Roche, R., 2000. Analytical solution for testing debris avalanche numerical models. Pure and Applied Geophysics 157 (6–8), 1081–1096. https://doi.org/10.1007/s000240050018.

McDougall, S., 2017. 2014 Canadian Geotechnical Colloquium: Landslide runout analysis—Current practice and challenges. 54 (5), 605–620.

McDougall, S., Hungr, O., 2004. A model for the analysis of rapid landslide motion across three-dimensional terrain. Canadian Geotechnical Journal 41 (6), 1084–1097.

McDougall, S., Hungr, O., 2005. Dynamic modelling of entrainment in rapid landslides. Canadian Geotechnical Journal 42 (5), 1437–1448. https://doi.org/10.1139/t05-064.

McDougall, S. (2006). *A New Continuum Dynamic Model For the Analysis of Extremely Rapid Landslide Motion Across Complex 3D Terrain, PhD Thesis* (Issue August).

Monaghan, J.J., 1992. Smoothed Particle Hydrodynamics. Annual Review of Astronomy and Astrophysics 30 (1), 543–574. https://doi.org/10.1146/annurev.astro.30.1.543.

Morton, G.M., 1966. A Computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. IBM Ltd, Technical Report, Ottawa, Canada.

Peng, C., Wang, S., Wu, W., Yu, H., Wang, C., Chen, J., 2019. LOQUAT: An open-source GPU-accelerated SPH solver for geotechnical modeling. Acta Geotechnica 14 (5), 1269–1287. https://doi.org/10.1007/s11440-019-00839-1.

Peng, C., Li, S., Wu, W., An, H., Chen, X., Ouyang, C., Tang, H., 2022. On three-dimensional SPH modelling of large-scale landslides. Canadian Geotechnical Journal 59 (1), 24–39. https://doi.org/10.1139/cgj-2020-0774.

Song, Y., Huang, D., Zeng, B., 2017. GPU-based parallel computation for discontinuous deformation analysis (DDA) method and its application to modelling earthquake-induced landslide. Computers and Geotechnics 86, 80–94. https://doi.org/10.1016/j.compgeo.2017.01.001.

Wang, W., Zhang, H., Zheng, L., Zhang, Y. bin, Wu, Y. qiang, & Liu, S. guang. (2017). A new approach for modeling landslide movement over 3D topography using 3D discontinuous deformation analysis. *Computers and Geotechnics*, *81*, 87–97. https://doi.org/10.1016/j.compgeo.2016.07.015.

Wieland, M., Gray, J.M.N.T., Hutter, K., 1999. Cohesionless Granular Avalanches in a Chute. J. Fluid Mech. 392, 73–100.

Xu, W., Xu, Q., Liu, G., Xu, H., 2021. A novel parameter inversion method for an improved DEM simulation of a river damming process by a large-scale landslide. Engineering Geology 293 (November 2020), 106282. https://doi.org/10.1016/j.enggeo.2021.106282.