SMODERP2D SOIL EROSION MODEL ENTERING AN OPEN SOURCE ERA WITH GPU-BASED PARALLELIZATION

M. Landa[1], J. Jeřábek[2], O. Pešek[1], P. Kavka[2]

[1] Dept. of Geomatics, Faculty of Civil Engineering, Czech Technical University in Prague, Czech Republic - (martin.landa, ondrej.pesek)@fsv.cvut.cz
[2] Dept. of Landscape Water Conservation, Faculty of Civil Engineering, Czech Technical University in Prague, Czech Republic - (jakub.jerabek, petr.kavka)@fsv.cvut.cz

Commission IV, WG IV/4

KEY WORDS: Hydrology, Soil Erosion Models, GIS, Open Source, Parallel computing

ABSTRACT:

SMODERP2D is a runoff-soil erosion physically-based distributed episodic model used for calculation and prediction processes at agricultural areas and small watersheds. The core of the model is a raster based cell-by-cell mass balance calculation which includes the key hydrological processes, such as effective precipitation, surface runoff and stream network routing. Effective precipitation, the forcing of the runoff and erosion processes, is reduced by surface retention and infiltration. Surface runoff consists of two components: slower sheet and concentrated rapid rill flow. Stream network routing is performed line-by-line in the user predefined polyline layer.
SMODERP is a long-term project driven by the Department of Landscape Water Conservation at the Czech Technical University in Prague. At the beginning, SMODERP has been developed as a surface runoff simulated by profile model (1D). Later the model has been redesigned using a spatially distributed method. This version is named SMODERP2D. Ongoing development is focused on obtaining parameters of the hydrological models, incorporating new infiltration and flow routing routines, and conceptualization of a rill flow and rill development. The model belongs to a family of so-called GIS-based hydrological models utilizing capabilities of GIS software for geospatial data processing. Importantly, the SMODERP2D project is currently entering an open source world. Originally the model could be run only in proprietary Esri ArcGIS platform. A new version of the model presented by this manuscript adds support for two key open source GIS platforms, GRASS GIS and QGIS. A newly developed GRASS module and QGIS plugin significantly increases the accessibility of the SMODERP2D model for research purposes and also for engineering practice.
Middle scale distributed hydrological models often encounter with high computation costs and long model runtime. Long runtime is caused by high-resolution input data which is easily available nowadays. The project also includes an experimental version of the SMODERP2D model enabling the parallelization of computations. This parallelization is done using TensorFlow, and its goal is to decrease the time needed for its run. It is supported by both CPU and GPU. Parallelization of computations is an important step towards providing SMODERP2D web processing services in order to allow quick and easy integration to highly specialized platforms such as Atlas Ltd.

1. INTRODUCTION

Erosion / hydrological models (EH) are being used for various research or engineering purposes. Results of such models may be used as input information for planning or designing soil conservation measures in the landscape and hydrological units - basins. Runoff water volume and transported soil amounts or discharge time series are being calculated in order to design the protection measures sufficient for a given flood or soil transport event. Another example of a practical application of EH models may be land-use change, build up areas development studies or effect of those on water or soil transport regime. Great use of EH models is also in extreme event forecasting. In research, EH models are being used to proof a new theory or to test hypotheses related to mechanism controlling the runoff and soil transport.

Empirical erosion models are often based on Universal Soil Loss Equation (USLE), (Wischmeier et al., 1978, Renard et al., 1997) and empirical hydrological models on the Curve number method (CN) (Cronshey, 1986), concepts more than 30 years old. Using empirical approaches may introduce limitations in the protection measures design e.g. because mentioned models do not take into account the transient nature of modelled processes. Physically based models are being developed to overcome the empirical models limitations.

Processes taking place in a landscape are spatially distributed, which is the reason why GIS (Geographic Information System) is often deploying in the modelling process taking advantage of ready to use GIS features. EH models have similar structure (although each model is specific in the terms of processes solved, its purpose or coding strategy). Runoff and soil loss are initiated by precipitation which is, especially for larger areas, spatially distributed process. Majority of models include an infiltration routine with spatially distributed parameters, since grassland and parking lot may be presented in a single hydrological model and have vastly distinct infiltration characteristic. Infiltrated water is transported to the soil which has varying transport properties. Ponging water creates overland flow which leads to a soil transport and may cause severe soil and nutrition losses in the landscape. Linear (water courses, streets, ditches)

or points (typically a water pump) features may be presented in the modelled system and affect the water flow or soil transport regime. GIS software has tools to operate with the linear and point features, and geospatial data which simplifies modellers live.

The EH model may encounter with some run-time issues which rise from model spatial and temporal discretization. Data availability and larger computation resources lead more often to the use of finer spatial resolution. It was noted in (Molnar, Julien, 2000) that raster grid cell size is interchangeable in the terms of a spatial discretization if the model parameters were calibrated on the model with the same raster grid size. Finer spatial resolution, in some cases, causes problems with a time discretization and the time step size. Time step size is commonly controlled with Courant–Friedrichs–Lewy (CFL) criterion (Courant et al., 1928). CFL criterion forces the time step to decrease if: a) velocity of flow process increases or; b) the spatial discretization becomes finer. Maximum acceptable CFL value, which preserves computation stability, theoretically equals one. For shallow surface processes (processes which take place in the used model) CFL criterion should be even smaller than one as it was noted in (Zhang, Cundy, 1989) or (Esteves et al., 2000). The need for smaller than one CFL criterion is caused mainly by the discrepancy between a solution (surface water height), cell size and surface roughness coefficient or by sharp surface slope changes between adjacent cells.

In the case of EH models, the commonly computed processes are sheet and rill flow. The sheet flow covers the earth's surface evenly, whereas rill flow detaches the soil material and concentrates its flow in the created rill (therefore it is also called concentrated flow). Although the concentrated rill flow is particularly fast (causing the time step size constraint) it usually occupies a small portion of the area. The computation may end up in a situation where a small portion of the computed area demands a shorter time step (due to rill flow presence) whereas the rest of the area allows larger time step. In that case, only a small part of the computed area with developed rills causes a long model run-time.

To summarize and outline the objectives of this manuscript. The advantage of high-resolution geospatial data availability is constrained with an increasing computation demands of a calculation. In the case of this manuscript, the extremely short time steps caused by the needs of CFL criterion is the main concern. Not all computed processes need a shorter time step and processes which are spatially limited (the concentrated flow in rill). In other words, the whole basin computation run-time is being increased due to a small part of the computed basin. One way to overcome this problem is to use GPU or CPU-based parallelization. In this manuscript, TensorFlow Python library (Abadi et al., 2015) was tested to parallelize the EH model. Besides the TensorFlow also a CPU-based parallelization is outlined. The testing was performed with the SMODERP2D EH model. The model calculates the surface runoff and soil loss processes with the use of GIS software for the data pre- and postprocessing. GRASS GIS provider and QGIS plugin were lately implemented in the SMODERP2D project, next to the already existing Esri ArcGIS Toolbox. Those new features and some of the principles used in the SMODERP2D model are also presented in this manuscript.

2. MATERIAL AND METHODS

2.1 SMODERP2D model

The SMODERP2D model has been integrated in open source GIS packages and tested for the GPU/CPU parallelization within presented work. The model, which is now capable of 2D calculation, has been developed from the 1D profile version (Holý, 1984). Description of the model follows.

The model has a simple structure based on the mass balance equation:

$$\frac{Storage}{\Delta t} = Inflow - Outflow \tag{1}$$

where *Storage* represents surface water level h $[L]$ which changes each proceeding time during the computation. $Inflow$ and $Outflow$ terms on the right-hand side of the equation (1) represent the water flowing in and out the storage during the time step Δt and consist of several components. The $Inflow$ and $Outflow$ of i_{th} raster cell are defined as:

$$Inflow_i = es_i + \sum_j^n q_j \tag{2}$$

$$Outflow_i = inf_i - q_i - ret_i \tag{3}$$

where es = effective precipitation $[LT^{-1}]$
q = inflow to resp. outflow from a given cell $[LT^{-1}]$
inf = infiltration $[LT^{-1}]$
ret = surface retention for a given raster cell $[LT^{-1}]$

The sum $\sum_j^n$ in the expression (2) represents sum of all inflows to the cell i. The flow direction and therefore the sum $\sum_j^n$ is controlled by D8 flow direction algorithm (O'Callaghan, Mark, 1984). Effective precipitation es is potential precipitation reduced by interception of the rainfall water on the vegetation.

The model is forced to satisfy the Courant–Friedrichs–Lewy (CFL) criterion (Courant et al., 1928):

$$CFL = \frac{q\mathrm{d}t}{\mathrm{d}x} < 1.0 \tag{4}$$

where $\mathrm{d}t$ = time step $[T]$
$\mathrm{d}x$ = grid cell size $[L]$

If the flow q is high, the model is forced to decrease the time step in order to satisfy the CFL criterion, since a grid cell size is fixed.

The flow q in the equations (2) and (3) has two components. Slower and spatially extensive sheet flow q_{sh}:

$$q_{sh} = XI^Y h^b \tag{5}$$

where X, Y, b = empirical parameters $[-]$
I = surface slope $[-]$

and faster concentrated rill flow q_{rl} calculated by the Mannings formula:

$$q_{rl} = A\frac{1}{n}R^{2/3}I^{1/2} \tag{6}$$

where A = cross-section area $[L^2]$
n = roughness in the rill $[TL^{-1/3}]$
R = hydraulic radii $[L]$

The resulting flow is a sum of sheet and rill flow:

$$q = q_{rl} + q_{rl} \tag{7}$$

The sheet flow starts when the infiltration capacity is exceeded; when rainfall is higher than infiltration. The rill flow emerges if a critical water level of sheet flow is exceeded. The critical water level is defined based on critical shear stress; when the drag force of the flowing water becomes large than the cohesive forces of the soil particles. From the definition, the sheet flow does not occur all over the basin area. The rill flow is usually presented to even lower extend. However, the CFL criterion is more likely constrained by the rapid rill flow even though its it occupy smaller area compared to sheet flow.

Infiltration is solved with Phillip's infiltration equation (Philip, 1957):

$$inf = 1/2St^{-1/2} + Ks \tag{8}$$

where S = sorptivity $[LT^{1/2}]$
K_s = saturated hydraulic conductivity $[LT^{-1}]$

Parameters of relations (5) (6) and (8) are in the most cases spatially distributed. It is therefore beneficial to incorporate GIS packages in the modeling process.

2.2 SMODERP2D entering an open source world

SMODERP2D is the project with a long history. Over the years its development has been driven by the Department of Landscape Water Conservation at the Czech Technical University in Prague (see SMODERP2D logo in Figure 1). In 2018 SMODERP2D developers started working on a new generation of the model in order to solve or at least to improve various critical issues of the project. This includes most importantly the computation stability and performance, better interoperability, and lack of documentation. Recently SMODERP2D source code has been published on GitHub (SMODERP2D Development Team, 2019) under GNU GPL licence in order to attract a wider audience, new developers and users.

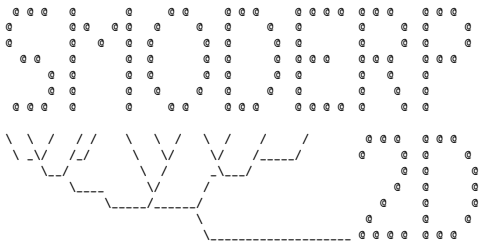

Figure 1. ASCII-art SMODERP2D project logo

The model is implemented in Python programming language using the object-oriented paradigm. The original source code has been designed with a low level of scalability, limited readability and interoperability. Part of the computation phase responsible for a data preprocessing was restricted to the single platform only, Esri ArcGIS. In 2018 the original source code has been completely refactorized. Python classes defining computational steps were re-organized in a hierarchical manner. Major design-related changes have been done in Python classes responsible for data handling and preparation using GIS software tools. Data preparation workflow is handled by a newly-defined a base, partly abstract Python class (`BaseProvider` in Figure 2). Functionality depending on the used GIS package has been separated into new classes. This step was crucial in order to make data preparation workflow GIS package independent. The only supported platform, Esri ArcGIS, has been separated from the base workflow. Based on that, a new concept of so-called *GIS providers* has been introduced, see Figure 2.

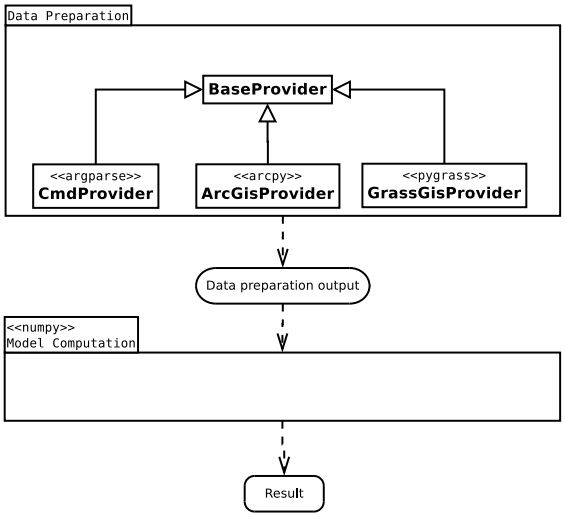

Figure 2. Concept of GIS providers (software dependencies outlined by stereotypes)

Crucial is the separation of GIS functionality related code from the generic workflow defined by the base provider. The base provider depends only on standard builtin Python libraries. Array-like computation is performed by a well-known NumPy library. Using GIS provider prototypes, the SMODERP2D project can be easily extended to support other GIS packages. Currently, the SMODERP2D project comes with three different GIS providers. Support for Esri ArcGIS platform is implemented by `ArcGISProvider`, GRASS GIS is handled by `GrassGisProvider`, see 2.2.1 for details. The `CmdProvider` is triggered only when the model computation is run from a command-line. In this case, it is assumed that the data preparation phase has been already performed by one of the supported GIS platforms.

Example of running computation from a command-line below. Option *–typecomp roff* specifies that only model computation without data preparation phase is triggered. It means that data has been already preprocessed and stored in a pickle file distributed by a *test.ini* configuration file.

```
python ./bin/start-smoderp2d.py --typecomp roff \
 --indata tests/test.ini
```

2.2.1 GRASS GIS integration SMODERP2D supported GIS platforms have been recently extended by a new GRASS-based GIS provider. Introducing an open source GIS platform to SMODERP2D workflow is crucial from the perspective of interoperability. SMODERP2D users can choose between a proprietary Esri ArcGIS platform and an open source GRASS GIS (Neteler et al., 2012). The GRASS GIS provider is designed similarly to ArcGIS provider. From a Python perspective, there is only one difference, GIS functions are accessed by PyGRASS package (Zambelli et al., 2013). Nevertheless, an integration of GRASS tools in the SMODERP2D project required a few improvements in GRASS GIS itself. That was possible since GRASS GIS is an open source project distributed under

GNU GPL licence. These improvements have been integrated into main distribution and will be part of upcoming GRASS GIS version 7.8. A GRASS *v.to.points* module (GRASS Development Team, 2019b) has been extended to extract from lines start or end nodes only. This functionality is used to determine the slope of a polyline stream feature to ensure that its direction will always be downslope. Another improvement is related to a *v.to.db* GRASS module (GRASS Development Team, 2019a). This tool allows uploading geometry-related information into the attribute table. Newly added option *next_edge* allows adding information about the next left and right edge based on the segment orientation determined from surface slope. This functionality is important for SMODERP2D in order to determine stream network correct connectivity as Figure 3 shows.

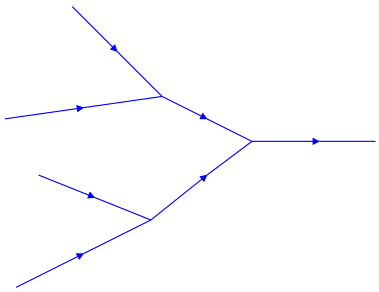

Figure 3. Stream segmentation procedure network connectivity

On the top of the GRASS GIS provider a specialized GRASS *r.smoderp2d* module has been designed. This tool allows the user running SMODERP2D model computation directly from GRASS GIS working environment as demonstrated in Figure 4. The module can be easily installed in GRASS GIS similarly to other extensions (so-called addons modules) by *g.extension* command. By default, the *r.smoderp2d* module performs data preparation phase followed by model computation steps. Data preparation only can be performed by *-d* flag. In this case, the module creates a binary pickle file which can be later used for a subsequent model computation. Note that ArcGIS Toolbox also allows creating a pickle file for later usage. Importantly, such pickle files are platform independent.

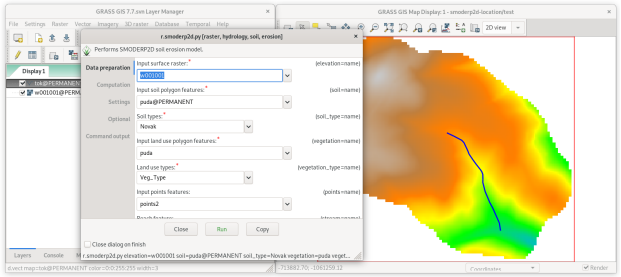

Figure 4. Running r.smoderp2d module from GRASS GIS graphical user interface

r.smoderp2d command-line usage example:

```
r.smoderp2d elevation=w001001 soil=soil_map \
 soil_type=Novak vegetation=soil_map \
 vegetation_type=veg rainfall_file=rainfall.txt \
 points=points2 table_soil_vegetation=tab_sv \
 table_soil_vegetation_code=soilveg \
 table_stream_shape=tab_stream_shape \
 table_stream_shape_code=smoderp stream=stream
```

2.2.2 QGIS plugin Recently the SMODERP2D model has been integrated also into QGIS environment. QGIS[1] is a widely used open source GIS platform which can be easily extended by user-defined plugins. A SMODERP2D QGIS plugin allows performing both data preparation and model computation phases in QGIS native environment, see Figure 5. Data preprocessing is ensured by GRASS GIS provider as described in 2.2.1. Note that QGIS installation normally comes with GRASS GIS included. It means that GRASS dependency is solved by QGIS installation itself. Experimental code of the plugin compatible with the current long term release QGIS version 3.4 is available from the project GitHub repository (SMODERP2D Development Team, 2019).

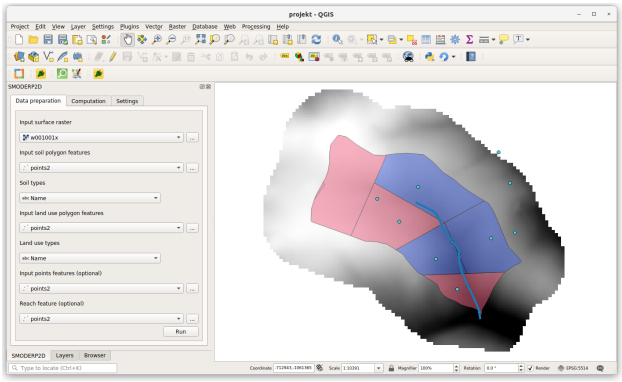

Figure 5. SMODERP2D model implemented as QGIS plugin

2.2.3 Python3 support SMODERP2D project also comes with Python 3 support, but still supporting Python 2. Note that Python versions 2 and 3 are not backwards compatible. Python 3 support is important from various perspectives. Python 2 is slowly reaching the end of life[2], but still used by many GIS platforms such as Esri ArcGIS 10.x. Newly supported GIS platforms by the SMODERP2D project as Esri ArcGIS Pro, (upcoming) GRASS GIS 7.8 and QGIS 3.x are Python 3 based. On the other hand it is still meaningful to support both Python versions, Python 2 mainly because of Esri ArcGIS 10.x platform.

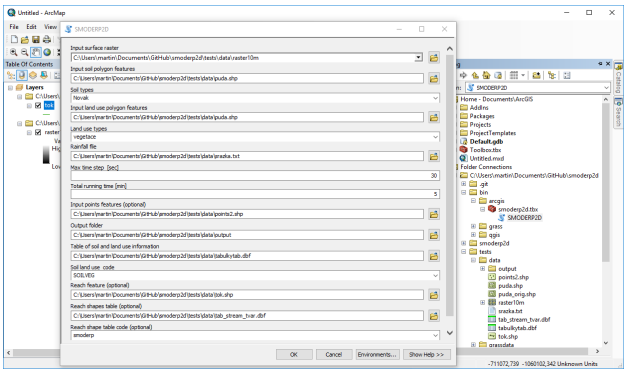

Figure 6. SMODERP2D model available as ArcToolbox for Esri ArcGIS 10.x and Pro platforms

[1] https://www.qgis.org
[2] https://legacy.python.org/dev/peps/pep-0373/

2.3 Parallel computing experiments

Because one of the most crucial points of SMODERP2D computations is the speed, an experimental branch allowing (both CPU and GPU-based) parallelized computations has been developed.

The main step was to rewrite all loop-based computations into matrix-based mathematical operations. To keep matrices as so-called tensors and to perform all the operations, an open source TensorFlow Python library (Abadi et al., 2015) developed by Google Brain Team[3] was used. Even though TensorFlow is most widely used for machine learning and its performance on basic mathematical operations is not always better than the one of NumPy (a quick comparison with NumPy and Numba can be seen in (Puget, 2015)), it had been preferred for its easy switch between CPU and GPU-based core (it depends only on the version of TensorFlow the user has installed, no needs for changes in the code) and therefore support also for users without an access to machines with GPU. Another advantage of TensorFlow is its usage of so-called graphs. A graph is a representation of all operations in dataflow/workflow. Its individual operations are automatically sent to multiple cores in a CPU or multiple threads in a GPU. These nodes are run independently in parallel.

To support further development of TensorFlow and exploit its bleeding edge functionalities, TensorFlow 2.0, which is published currently just as an alpha version, was used in the SMODERP2D experimental branch. Because TensorFlow 2.0 is still not suitable with all the Python acrobatic tricks, NumPy was used for matrix operations in places where TensorFlow could not (on places where loops were still needed; looping through a NumPy array is incomparably faster than through a Tensor).

This experimental SMODERP2D branch is still under development; however, the alpha-version is ready to be used. Table 1 presents the results of different tests made on this version (comparing parallelized GPU computation, parallelized CPU computation and a single CPU one).

As can be seen in the table, the usage of GPUs is not always the right way even when compared with CPUs, both single and parallelized ones. The bottleneck of TensorFlow is its graph initialization; this step is very time-consuming and therefore can last many times longer than the computation itself for extremely small data. Another bottleneck is the memory shift between RAM and GPU virtual memory which concludes into slower processes for weaker GPUs (compared with parallelized computations being run on CPUs). Generally, for data of common size was the process with parallelized computations faster (reaching around 60 per cent of the total computation time on different architectures). Interesting moment is slower run of much stronger $CPU4$ when compared with weaker $CPU2$; this behaviour has to be examined deeper.

2.3.1 Further ideas for a sub-basins based parallel computing Besides the GPU-based parallelization (with TensorFlow (Abadi et al., 2015) or NVIDIA Cuda technology (Kalyanapu et al., 2011, Le et al., 2015)) the pure CPU parallelization may also bring a good improvement in the computation time reduction. The computation domain is separated into sub-domains based on a certain algorithm where each sub-domain computation is loaded to a single CPU core. It is beneficial to incorporate the hydrological behaviour in the parallelization strategy if the domain is a hydrological basin. In (Vivoni et al., 2011) the basin was separated in sub-basins based on stream network. The sub-basins communicated with each other through so-called ghost cell. The strategy aimed to generate as few ghost cells as possible; to reduce the communication between the CPU cores.

Table 1. Results of parallelization tests

RAM	Processing unit	Data 62 KB [s]	Data 197 MB [s]
15 GB	GPU1	4.0	7,560
	CPU1	0.2	12,809
	CPU2	2.1	7,249
251 GB	GPU2	2.5	6,611
	CPU3	0.2	10,637
	CPU4	1.5	8,631

Table 2. Used processing units

ID	Model	Clock speed	Memory
GPU1	GeForce GTX 1060 3GB	33 MHz	3,016 MiB
GPU2	4× GeForce GTX 1080 Ti	33 MHz	11,178 MiB
CPU1	AMD Ryzen 7 1700 Eight Core Processor	1.373 GHz	512 KB
CPU2	16× AMD Ryzen 7 1700 Eight Core Processor	1.373 GHz	512 KB
CPU3	Intel Xeon CPU E5-2630 v4	2.4 GHz	25,600 KB
CPU4	40× Intel Xeon CPU E5-2630 v4	2.4 GHz	25,600 KB

The parallelization strategy outlined in the manuscript is based on the hydrological reality and it is shown in a simplified setup in Figure 7. In this example, the Nučice experimental catchment was chosen to present the parallelization strategy. At this 0.5 km^2 large basin a long-term monitoring of erosion and runoff processes is being conducted by the Dept. of Landscape Water Conservation.

The strategy main goal is the reduction of the communication between CPU-cores during the computation as much as possible. The whole basin is divided into several sub-basins based on the digital elevation model and user-defined sub-basin size. Outlet[4] of each sub-basin is depicted with red dots in Figure 7. After the sub-basins are defined, an order in which each sub-basin will be computed is defined as follows. Sub-basins which are hydrologically the farthest from the basin outlet (depicted by the triangle in Figure 7) and therefore have no inflow flow upslope area are calculated at first. Those sub-basins have the rainfall stored in hyetographs as the only input. In the simplified setup shown in Figure 7, the sub-basins 1, 2, 3, and 6 are

[3] https://ai.google/research/teams/brain/

[4] The location in the basin where all water from the basin flows

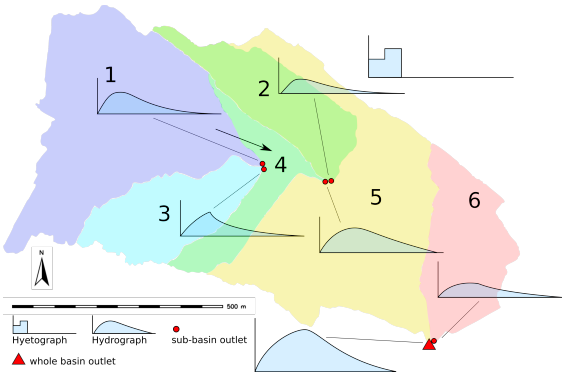

Figure 7. Simple example of possible CPU-based parallelization strategy for the experimental catchment Nučice

calculated at first in parallel. The calculated hydrographs of the sub-basins 1, 2, 3, and 6 are stored in the memory for the later use. Sub-basins which have sub-basins 1, 2, 3, and 6 in its upslope area are calculated next. It this case it is the only the sub-basin 4. The water inputs in the sub-basin 4 are now hyetograph and also hydrographs of upslope sub-basins 1 and 3. Next sub-basin to be calculated is the final sub-basin 5. In this sub-basin is the outlet of the whole area. The water input in the final sub-basin 5 are hyetograph and hydrographs of sub-basins 2, 4 and 6. Once the main outlet hydrograph is obtained the calculation stores the results and stops.

This approach may encounter several limitations. The main one originates from the basin geometry. In the case of a narrow basin situation (each sub-basin has a single upslope and downslope sub-basin), the sub-basins will be computed in a sequence, which loses the advantage of multi-core working station. If this situation happens the user will be forced to create very small sub-basins in order to be able to perform the outlined CPU parallelization. The possibilities of CPU parallelization described in this section will be the subject of further research.

3. CONCLUSION

This manuscript presents SMODERP2D project and related recently triggered development. SMODERP2D computational tools have been successfully integrated into Esri ArcGIS, GRASS GIS and QGIS desktop GIS platforms. On the top of that, the concept of so-called GIS providers has been introduced. Ongoing development is mainly focused on computational routines and parallel computation experiments. Also OGC Web Processing Service providing SMODERP2D functionality is planned to be established. All the tools are currently distributed as experimental ready for testing and user feedback. The official stable release of SMODERP2D model is planned for 2020. This includes also user documentation which is currently under development.

In the case of SMODERP2D model, the run-time is an issue, especially if multiple mid-scale hydrological basins in fine spatial resolution grid computation needs to be undertaken. The code parallelization is a common practice in cases where the reduction of run-time is convenient or even necessary, therefore the existence of the TensorFlow-based branch; and although this branch is still under development, the reduction of the computation costs is already reaching up to 40 per cent depending on the data and architecture. This experiment also shows that the parallelized branch should not be used as the default one, but an ad hoc solution should be chosen depending on the data and available computing power. Even though the SMODERP2D model does not belong in the family of forecasting models (where the short run-time is necessary) the run-time speed up will increase the usability of the model in practice and research applications.

SMODERP2D source code is available on GitHub (SMODERP2D Development Team, 2019) under GNU GPL licence.

ACKNOWLEDGEMENTS

The research has been supported by the research grants TJ01000270, QK1910029, and internal CTU grant SGS17/173/OHK1/T3/11.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Courant, R., Friedrichs, K., Lewy, H., 1928. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische annalen*, 100(1), 32–74.

Cronshey, R., 1986. Urban hydrology for small watersheds. Technical report, US Dept. of Agriculture, Soil Conservation Service, Engineering Division.

Esteves, M., Faucher, X., Galle, S., Vauclin, M., 2000. Overland flow and infiltration modelling for small plots during unsteady rain: numerical results versus observed values. *Journal of hydrology*, 228(3-4), 265–282.

GRASS Development Team, 2019a. v.to.db GRASS module. Geographic Resources Analysis Support System (GRASS) Software, Version 7.7. Open Source Geospatial Foundation. https://grass.osgeo.org/grass77/manuals/v.to.db.html (2 June 2019).

GRASS Development Team, 2019b. v.to.points GRASS module. Geographic Resources Analysis Support System (GRASS) Software, Version 7.7. Open Source Geospatial Foundation. https://grass.osgeo.org/grass77/manuals/v.to.points.html (2 June 2019).

Holý, M., 1984. Vztahy mezi povrchovým odtokem a transportem živin v povodí vodárenských nádrží (dílčí zpráva výzkumného úkolu VI 4 15/01 03/) (in czech). Technical report, CTU in Prague, Prague.

Kalyanapu, A. J., Shankar, S., Pardyjak, E. R., Judi, D. R., Burian, S. J., 2011. Assessment of GPU computational enhancement to a 2D flood model. *Environmental Modelling and Software*, 26(8), 1009–1016.

Le, P. V., Kumar, P., Valocchi, A. J., Dang, H. V., 2015. GPU-based high-performance computing for integrated surface-subsurface flow modeling. *Environmental Modelling and Software*, 73, 1–13. http://dx.doi.org/10.1016/j.envsoft.2015.07.015.

Molnar, D., Julien, P., 2000. Grid-size effects on surface runoff modeling. *Journal of Hydrologic Engineering*, 5(1), 8–16.

Neteler, M., Bowman, M., Landa, M., Metz, M., 2012. GRASS GIS: a multi-purpose Open Source GIS. *Environmental Modelling & Software*, 31, 124–130.

O'Callaghan, J. F., Mark, D. M., 1984. The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics, and Image Processing*, 28(3), 323 - 344.

Philip, J., 1957. The theory of infiltration: 1. The infiltration equation and its solution. *Soil science*, 83(5), 345–358.

Puget, J., 2015. How to quickly compute the mandelbrot set in python. IBM Community Blog. https://www.ibm.com/developerworks/community/blogs/jfp/entry/How_To_Compute_Mandelbrodt_Set_Quickly?lang=en (4 June 2019).

Renard, K. G., Foster, G. R., Weesies, G., McCool, D., Yoder, D. et al., 1997. *Predicting soil erosion by water: a guide to conservation planning with the Revised Universal Soil Loss Equation (RUSLE)*. 703, United States Department of Agriculture Washington, DC.

SMODERP2D Development Team, 2019. SMODERP2D GitHub repository. https://github.com/storm-fsv-cvut/smoderp2d (2 June 2019).

Vivoni, E. R., Mascaro, G., Mniszewski, S., Fasel, P., Springer, E. P., Ivanov, V. Y., Bras, R. L., 2011. Real-world hydrologic assessment of a fully-distributed hydrological model in a parallel computing environment. *Journal of Hydrology*, 409(1-2), 483–496. http://dx.doi.org/10.1016/j.jhydrol.2011.08.053.

Wischmeier, W. H., Smith, D. D. et al., 1978. Predicting rainfall erosion losses-a guide to conservation planning. *Predicting rainfall erosion losses-a guide to conservation planning.*

Zambelli, P., Gebbert, S., Ciolli, M., 2013. Pygrass: An Object Oriented Python Application Programming Interface (API) for Geographic Resources Analysis Support System (GRASS) Geographic Information System (GIS). *ISPRS International Journal of Geo-Information*, 2(1), 201–219. https://www.mdpi.com/2220-9964/2/1/201.

Zhang, W., Cundy, T. W., 1989. Modeling of two-dimensional overland flow. *Water Resources Research*, 25(9), 2019–2035.