SoftwareX 7 (2018) 263–272

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

SPGM: A Scalable PaleoGeomorphology Model

Rakib Hassan [a,*,1], Michael Gurnis [b], Simon E. Williams [a], R. Dietmar Müller [a]

[a] EarthByte Group, School of Geosciences, University of Sydney, Sydney, New South Wales 2006, Australia
[b] Seismological Laboratory, California Institute of Technology, Pasadena, CA 91125, USA

ARTICLE INFO

Article history:
Received 28 January 2016
Received in revised form 31 October 2017
Accepted 17 July 2018

Keywords:
Surface processes
Geomorphology
Paleogeomorphology
Morphodynamics
Landscape evolution model

ABSTRACT

Numerical models of landscape evolution are playing an increasingly important role in providing an improved understanding of geomorphic transport processes shaping Earth's surface topography. Improving theoretical underpinnings coupled with increasing computational capacity has led to the development of several open source codes written in low-level languages. However, adapting these codes to new functionality or introducing greater flexibility often requires significant recoding. Here we present a multi-process, scalable, numerical model of geomorphological evolution, built with a modular structure and geared toward seamless extensibility. We implement recent algorithmic advances that reduce the computational cost of flow routing – a problem that typically scales quadratically with the number of unknowns – to linear in time while allowing for parallel implementations of geomorphic transport processes. Our scalability tests demonstrate that such parallelizations can achieve an order of magnitude speedup on a typical desktop computer, making large-scale simulations more tractable.

Code metadata

Current code version	1.0
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-16-00021
Legal Code License	GNU General Public License, version 2 (GPL-2.0)
Code versioning system used	git
Software code languages, tools, and services used	c++
Compilation requirements, operating environments & dependencies	**gcc** compiler, **scons** build system
If available Link to developer documentation/manual	dev-docs
Support email for questions	rakib.hassan@ga.gov.au

1. Introduction

The interplay of a variety of physical processes acting over a range of space and time scales is responsible for the evolution of Earth's surface topography. These physical processes broadly fall in four categories: (i) river incision that leads to advective transport of material over distances of up to several thousands of km, (ii) diffusive processes such as soil creep and landslides that operate over comparatively shorter length scales, (iii) large-scale surface deformation arising from tectonic forces in the crust and lithosphere, and (iv) transient long-wavelength but small-amplitude topographic variations arising from deep mantle convective forces that operate over timescales of tens of millions of years [1]. In recent decades, numerical models of geomorphological evolution have become an important tool to facilitate a better understanding of these coupled processes that sculpt surface topography [2–9].

Geomorphological (or landscape) evolution models have made rapid advances in recent decades as a result of the increasing computational capacity of computing hardware. However, the numerical resolution necessary to study geomorphological evolution over continental scales, spanning tens of millions of years – particularly aimed at better understanding the influence of long wavelength but small amplitude dynamic topography – still poses significant computational challenges. Wallis et al. [10] and Do et al. [11] presented parallel flow routing algorithms based on the message-passing-interface (MPI) on clusters and implementations on graphics processing units (GPUs) have also been reported [12,13], which

* Corresponding author.
E-mail address: rakib.hassan@ga.gov.au (R. Hassan).
[1] Now at Geoscience Australia, GPO Box 378, Canberra 2601, ACT, Australia.

https://doi.org/10.1016/j.softx.2018.07.005

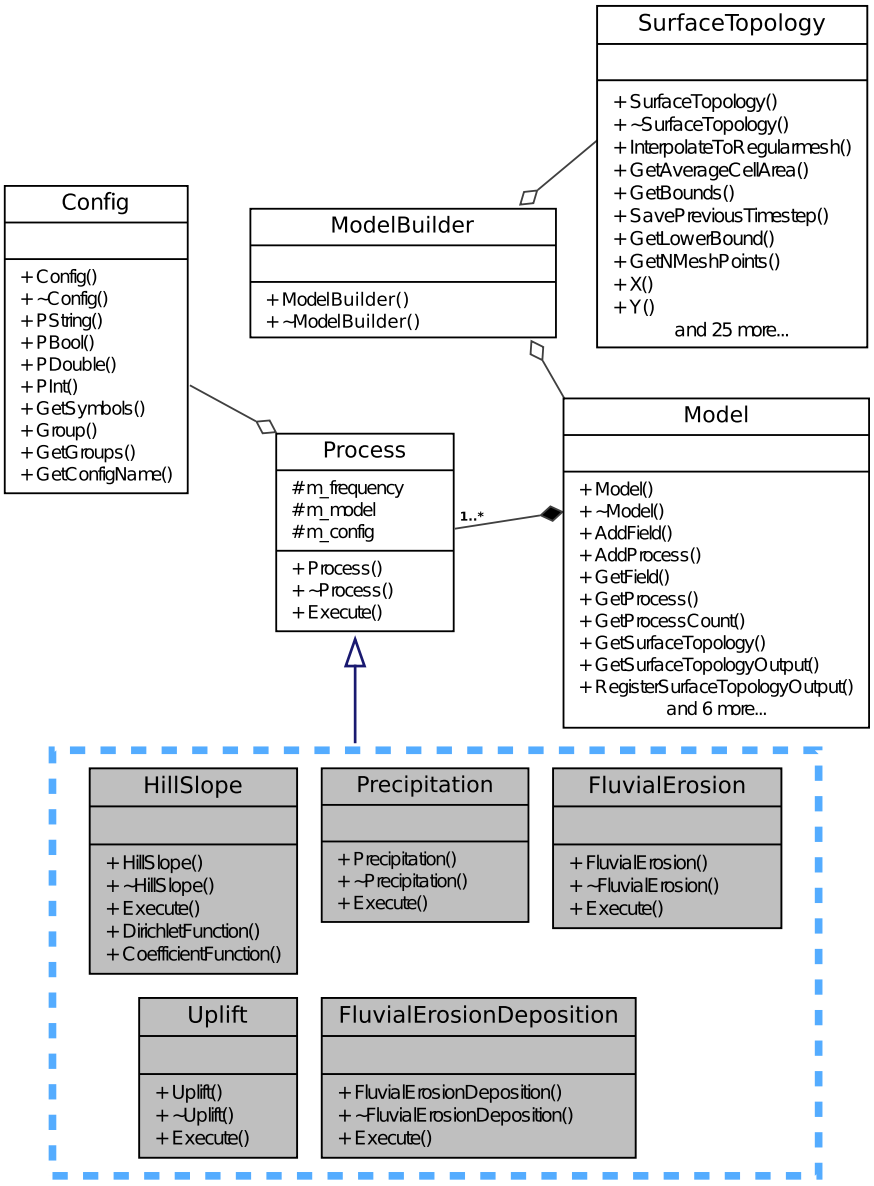

Fig. 1. Class diagrams showing the basic building blocks of SPGM.

show significant speedups. Braun and Willett [9] presented a parallelization approach based on open-multi-processing (OpenMP), producing near-linear scalability on a shared-memory machine. However, both MPI- and GPU-based parallelization strategies involve increased development and debugging times and are likely to compromise ease of code adaptability. We therefore adopt the parallelization approach described in Braun and Willett [9] for its simplicity and ease of implementation.

2. Software framework

SPGM has been implemented using C++ [14] and the basic abstractions utilized are shown in Fig. 1. The *Config* class is responsible for parsing input configuration files and for providing collaborating classes access to named parameters and parameter-groups. Each of the physical processes implemented in SPGM – which we interchangeably refer to as 'modules', e.g. *Precipitation* – inherit from the abstract *Process* class. The *Process* class contains a reference to the *Config* class and its sub-classes (blue rectangle in Fig. 1) implement the *Execute()* function, which contains the core numerical algorithm pertaining to a given physical process. The *ModelBuilder* class instantiates a number of '*Processes*' based on an input configuration file and populates an instance of the *Model* class with them. The *SurfaceTopology* class manages the underlying geometry of the computational mesh and recomputes drainage networks at the beginning of each time step, as shown in the program flowchart in Fig. 2.

In the following sections we describe the parsing of configuration files where general parameter values, as well as time-series data for specific parameters can be specified. We also briefly describe the algorithms used in mesh generation, flow-routing and output generation.

2.1. Configuration files and parameter time series

We use simple, plain-text configuration files to list a number of required parameters, followed by several groups of parameters corresponding to physical processes modeled. Parameter values