Geomorphology 180–181 (2013) 170–179

Contents lists available at SciVerse ScienceDirect

Geomorphology

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

A very efficient $O(n)$, implicit and parallel method to solve the stream power equation governing fluvial incision and landscape evolution

Jean Braun [a,*], Sean D. Willett [b]

[a] Institut des Sciences de la Terre, Université Joseph Fourier and CNRS, BP 53, 38041 Grenoble Cedex 9, France
[b] Department of Earth Sciences, ETH, Zurich, Switzerland

ARTICLE INFO

Article history:
Received 24 April 2012
Received in revised form 29 September 2012
Accepted 9 October 2012
Available online 17 October 2012

Keywords:
Fluvial geomorphology
Numerical modeling
Stream power equation
Routing algorithm
Implicit scheme
Parallel algorithm

ABSTRACT

We present a new algorithm to solve the basic stream power equation, which governs channel incision and landscape evolution in many geomorphic settings. The algorithm is highly efficient because computation time increases linearly with the number of points used to discretize the landscape and is ideally suited to parallelization. It is also unconditionally stable because it uses an implicit scheme for the time integration of the landscape evolution equation, which means that large time steps can be used without sacrificing accuracy. In this paper we describe the algorithm and present results that demonstrate its efficiency and accuracy.

1. Introduction

The evolution of the Earth's surface topography is the result of a large variety of processes acting on a wide range of scales, including fluvial and glacial incision, transport and deposition, aeolian processes, hillslope processes (such as soil creep, landsliding, rain splash or chemical weathering) and karst formation. In many instances, especially where the topographic surface is characterized by relatively large mean slopes, such as in tectonically active areas, the rate of landscape evolution is primarily set by the efficiency of river or channel incision (Whipple, 2004). In such environments, incision at the bottom of well-defined channels generates slopes, a process which destabilizes hillsides and causes mass movement by gravity-driven processes. The sediment resulting from these gravity-driven processes is then deposited in the river channel, which then acts as the main transport agent toward lower elevations (Whipple and Tucker, 1999; Whipple, 2004).

Studying the rate of landscape evolution is thus commonly equated with a better understanding of how channels incise into the bedrock. The processes at play during river incision are numerous and varied. The hydraulic shear stress exerted by the river onto the channel bed is given by

$$\tau_s = k_s \phi^p S^n \qquad (1)$$

where ϕ is discharge (total discharge or discharge per unit channel width), S is local slope, and k_s is a constant (Howard and Kerby, 1983; Whipple and Tucker, 1999). It is commonly accepted that the rate of river/channel incision is directly proportional to this shear stress, which leads to the classical ‘stream power equation’ to describe the rate of river incision:

$$\frac{\partial h}{\partial t} = -KA^m S^n \qquad (2)$$

where K is a factor that depends on lithology and mean precipitation rate (Whipple and Tucker, 1999), channel width, flood frequency, channel hydraulics, and potentially other parameters and processes (Lague and Hovius, 2005), and A is contributing drainage area; parameters ϕ and A are related through net precipitation, which can be uniform or spatially variable. This can be improved upon to incorporate several other important aspects of the processes acting in a river channel, such as the existence of a critical shear stress, potentially representing the protecting effect of bed cover or a dependence on bedload, q_s (Kooi and Beaumont, 1994; Davy and Lague, 2009), which allows us to represent the various states that characterize a river channel from a detachment-limited state, where it is the resistance to incision that limits the rate of incision, to a transport-limited state, where it is the ability of the river to transport and thus evacuate the products of erosion that limits its evolution. Inclusion of a critical shear stress is very important when considering variability in precipitation and runoff (Lague and Hovius, 2005). Other important factors include sediment supply variability (Lague, 2010) or dynamic

* Corresponding author. Tel.: +33 476514074; fax: +33 476514058.
E-mail address: Jean.Braun@ujf-grenoble.fr (J. Braun).

0169-555X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.geomorph.2012.10.008

channel width (Turowski et al., 2009) and much work is currently devoted to document and constrain them. Such complexities are important to fluvial development, but as these can be incorporated into the stream power model without modifying its form or its method of solution, which is the focus of this paper, we, therefore, retain just the standard form.

Eq. (2) has been solved using a wide range of numerical techniques and geometrical representation of landform (Kooi and Beaumont, 1994; Tucker and Slingerland, 1994; Braun and Sambridge, 1997; Howard, 1997; Crave and Davy, 2001) to predict the evolution of landforms in response to fluvial or channel incision. It is also coupled with simple to sophisticated models of precipitation, tectonic uplift, or lithological variations and needs to be associated with a set of boundary conditions, which commonly involves forcing parts of the integration domain (typically along its boundary) to remain at a fixed height that is usually referred to as the 'base level'.

Considering the various elements in Eq. (2), we can conclude that the computation of landscape evolution in situations where river incision is the rate-controlling mechanism is limited by our ability (i) to compute the contributing drainage area or discharge at each point on the landscape and (ii) to integrate in time Eq. (2) with efficiency.

The computation of drainage area is typically an $O(n_p^2)$ problem, i.e. the number of operations needed to find the solution varies as n_p^2 where n_p is the number of points of the mesh/grid on which it is calculated, such that basic ('brute force') algorithms become rapidly intractable for large values of n_p. More efficient algorithms have been proposed to compute drainage area or catchment geometries on a given digital elevation model (DEM) (Wilson et al., 2008); most are $O(n_p log n_p)$. Freeman (1991) proposed a recursive algorithm that is $O(n_p)$. Such an efficient method is essential when studying landscape evolution because computing drainage area or discharge has to be performed at each time step during the evolution of a landform as drainage geometry and connectivity evolve with time. This operation becomes the most intensive part of the computation for most landscape evolution models.

Because slope is the first-order derivative of elevation, h, Eq. (2) is in fact a nonlinear transport or advection equation with a spatially variable transport velocity ($=KA^m$):

$$\frac{\partial h}{\partial t} = -KA^m\left(\frac{\partial h}{\partial x}\right)^n. \tag{3}$$

Such an equation is known to be difficult to solve numerically because its solution potentially involves the propagation of steps or topographic knickpoints, which typically requires small time increments and/or complex (high order) finite difference schemes to insure stability and accuracy (Tucker and Hancock, 2010).

As frequently noted in the literature pertaining to numerical modeling of landscape evolution (see the review by (Tucker and Hancock, 2010), for example), the resolution of the models that are currently used is too low to either capture the details of the landscape (slope and curvature) or to study landscape evolution at the orogenic scale, such that a scaling procedure (homogenization) has to be applied to justify the use of an equation similar to Eq. (2) but at a resolution that does not guarantee that the real slopes are sampled (Tucker and Hancock, 2010). Typically, our current models are limited to a resolution of a few millions of points (n_p), which is equivalent to a numerical mesh of $10^3 \times 10^3$ and thus a resolution of a 100 m to 1 km, if one wishes to study an orogenic area of size 100 by 100 km.

Enhanced numerical resolution is necessary to solve the stream power equation over much larger surface areas than is presently done, for example to study the erosion of topography produced by flow in the Earth's mantle, the so-called 'dynamic topography' that is characterized by horizontal dimensions of 1000 km or more and has an amplitude that rarely exceeds 1000 m (Braun, 2010). Higher resolution also means that we can solve the basic geomorphic equation at a resolution that is much closer to the human scale of observation (1–10 m) and thus can include important processes such as lateral stream migration, meandering and the formation of anastomosing stream topologies, or the complex interactions between hillslopes and channels that lead to the formation and evolution of river terraces. An alternative to higher resolution discretization of the stream power equation was presented by Castelltort et al. (2012) who dealt with the multi-scale problem by using analytical solutions for the low drainage area parts of the landscape. Although this approach increased the accuracy of representation of the water-divides between drainage basins, it did not increase the resolution within river channels.

Here we propose a new algorithm to compute drainage area (or discharge) that is $O(n_p)$ (computing time varies linearly with the number of points used to discretize the landform) and adapted to parallel computing architecture[1]. It can thus be used to solve the basic equation (and its generalization) on meshes that are several orders of magnitude finer, i.e., on meshes containing 10^8 nodes on a laptop computer and, potentially, up to 10^{10} nodes on a large multi-processor computer. Furthermore, we show how the equation can be integrated using an implicit – and thus far more stable and accurate – method, which guarantees that much larger time steps can be used, further increasing the overall computational efficiency of the method.

In the first sections of this paper, we present the details of this new algorithm and how it can be implemented. In the second part of this paper, we present results obtained with the new algorithm to demonstrate its efficiency.

2. Ordering

Key to the algorithm is to find the order in which one must go through the nodes to compute discharge by adding progressively the contribution of each node to the river discharge (or its contributing area to the total drainage area). To perform this operation, we assume that water goes down the path of the steepest slope (O'Callaghan and Mark, 1984; Gallant and Wilson, 2000). Assuming that the landscape is represented by the height of a series of n_p points (as illustrated in Fig. 1), $h(i)$, with n_b of them being fixed at base level, one can compute, for each node, which of its neighbors defines the steepest descent/slope. This node is called the receiver node of node i and is noted $r(i)$. Our algorithm is thus based on the single-flow-direction approximation (each node has a single receiver) (O'Callaghan and Mark, 1984). Base level nodes have no receiver.

Note also that the algorithm does not depend on the topology/position of the nodes used to describe the landscape. They can be positioned at the corners of a regular rectangular mesh or not. A rectangular mesh makes it easy to find and/or compute the list of neighbors ($N_{i,j}$, for $j=1,\cdots,n_i$), where n_i is the number of neighbors connected to i. For example, on a rectangular grid, the number of neighbors is equal to nine if one takes the convention to include each node in the list of its own neighbors (equivalent of the D8 algorithm of (O'Callaghan and Mark, 1984)). In doing so, finding local minima on the landform becomes relatively simple: they correspond to nodes that are their own receiver ($r(i)=i$).

Before proceeding further, one needs to invert the list of receiver nodes to find the list of so-called 'donor nodes' to node i. The donor nodes form the ensemble of nodes that have i as a receiver. We will note this list $D_{i,j}$ for $j=1,\cdots,d_i$ where d_i is the number of nodes having i as a receiver and thus $r(D_{i,j})=i$. We know that the number of donor nodes to node i is limited by the number of neighbors to node i. Nodes that have no donors are thus the prime candidates to start the computation of the discharge.

[1] An algorithm that is amenable to parallelization requires that each operation on the nodes can be divided into subsets that are independent of each other and can therefore be distributed to different processors.

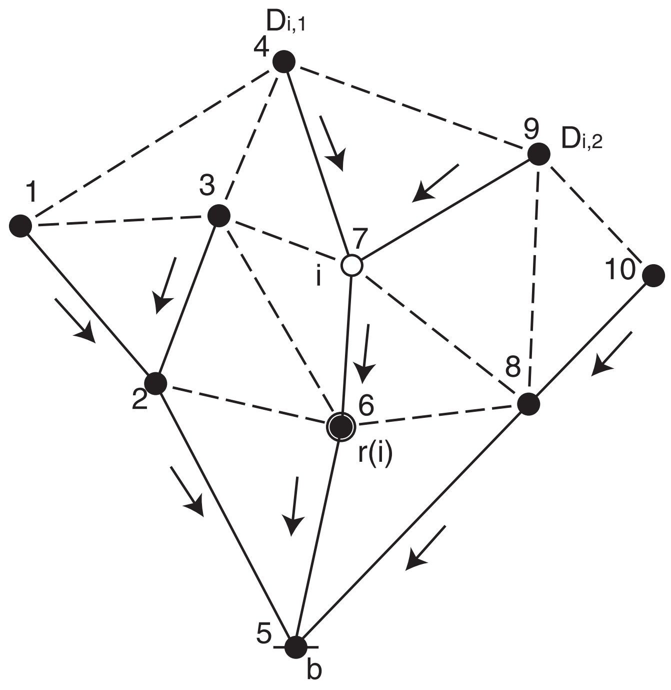

Fig. 1. Nodal representation of the landform. Nodes are indicated as black circles. The lines represent all the possible connections among neighboring nodes. The solid lines indicate the connections following the steepest descent hypothesis (indicated by the arrows). Dashed lines are connections that are not used to construct the network. The node named *b* is assumed to be at base level. Node *i* is white and its receiver has a thin circle around it. The nodes labeled $D_{i,1}$ and $D_{i,2}$ form the list of donor nodes to *i*. The numbers correspond to an arbitrary numbering scheme that is used to illustrate the construction of the stack as described in the text and in Table 1.

One can also show that :

$$\sum_{i=1}^{n_p} d_i = n_p \tag{4}$$

as each node has a single receiver and thus can only be in one list of donor nodes and only appear once in that list. This is a useful property insuring that the list of donor nodes can be stored in an array of length n_p.

To compute the list of donor nodes to each node *i*, we first initialize d_i:

$$d_i = 0 \quad \text{for } i = 1, \cdots, n_p \tag{5}$$

Table 1
Example of the building of the receiver and donor node information based on the nodal connectivity shown in Fig. 1. For each node *i*, the receiver node information *r(i)* can be inverted into donor array $D(i,j)$ and number of donors per node array, d_i. This information can be stored in a single dimension array D_i and an index table/array δ_i.

i:	1	2	3	4	5	6	7	8	9	10	
r(i):	2	5	2	7	5	5	6	5	7	8	
$D(i,j)$:	–	1	–	–	2	7	4	10	–	–	
	–	3	–	–	5	–	9	–	–	–	
	–	–	–	–	6	–	–	–	–	–	
	–	–	–	–	8	–	–	–	–	–	
d_i	0	2	0	0	4	1	2	1	0	0	$\sum d_i = 10 = n_p$
δ_i:	1	1	3	3	3	7	8	10	11	11	11
D_i	1	3	2	5	6	8	7	4	9	10	

and use the receiver information to build d_i:

$$d_{r(i)} = d_{r(i)} + 1 \quad \text{for } i = 1, \cdots, n_p. \tag{6}$$

This procedure is illustrated with a single example made of 10 nodes and shown in Fig. 1 and Table 1. Knowing that the list of donors (for all nodes) can be stored in an array of dimension *n*, we then transform the list d_i into an index array, δ_i, that contains the location of where the list of donors to node *i* is stored:

$$\delta_{n_p+1} = n_p + 1 \tag{7}$$

$$\delta_i = \delta_{i+1} - d_i \quad \text{for } i = n_p, \cdots, 1, -1. \tag{8}$$

We can then build the list of donors, according to :

$$\left.\begin{array}{r} D\left(\delta_{r(i)} + w_{r(i)}\right) = i \\ w_{r(i)} = w_{r(i)} + 1 \end{array}\right\} \quad \text{for } i = 1, \cdots, n_p \tag{9}$$

where w_i is an integer working array that has been initialized to 0 before this operation is performed. This vector, *D*, of length n_p, contains the donor information $D_{i,j}$:

$$D_{i,j} = D(\delta_i + j - 1) \quad \text{for } j = 1, \cdots, \delta_{i+1} - \delta_i. \tag{10}$$

Note that all the operations performed so far are $O(n_p)$, and they only require building vectors of length n_p at most.

Next, we build a vector *s(i)*, or stack, that contains the list of all nodes (from 1 to n_p) but in a specific order that will be used to perform the computation of the discharge and to solve the evolution equation. To build this stack, we take successively each of the n_b base level nodes b_i (in any order) and perform the following operation after initializing the global variable *j* to 1:

$$k = b_i \tag{11}$$

and:

$$\text{add\_to\_stack}(l,j), \quad \text{for } l = D\left(\delta_k, \delta_k + 1, \cdots, \delta_{k+1} - 1\right) \tag{12}$$

where the add_to_stack (*l*,*j*) operation corresponds to :

$$\begin{array}{c} j = j + 1 \\ s(j) = l \\ \text{add\_to\_stack}(m,j) \quad \text{for } m = D\left(\delta_l, \delta_l + 1, , \cdots, \delta_{l+1} - 1\right). \end{array} \tag{13}$$

Note that the add_to_stack operation is recursive (it calls itself). The recursion stops when the stack reaches a node that has no donors ($\delta_l = \delta_{l+1}$). The resulting stack is shown in Fig. 2 for the simplified example of Fig. 1.

With this procedure, we built a stack containing all the nodes belonging to the catchment of node b_i. In doing so, we can also perform a simple operation that propagates b_i to all the nodes, which is then transformed into a tag identifying which catchment a node belongs to.

Note again that during the construction of the stack each node of the grid is only reached once. Thus, the computation of the stack is performed in $n_p \times$ a set number of simple operations and is thus $O(n_p)$. The stack itself is of length n_p.

The computation of the stack is also amenable to parallelization if one distributes the base level nodes across the processors. Each processor will build a section of the stack (or substack), which will need to be assembled to form the complete stack. Note that because the remaining operations (discharge and sediment flux computation as well as solution of the evolution equation) will be performed on a catchment-per-catchment basis, these operations can also be distributed across the processors; and knowledge of the corresponding substack only is needed in each processor. Consequently, the

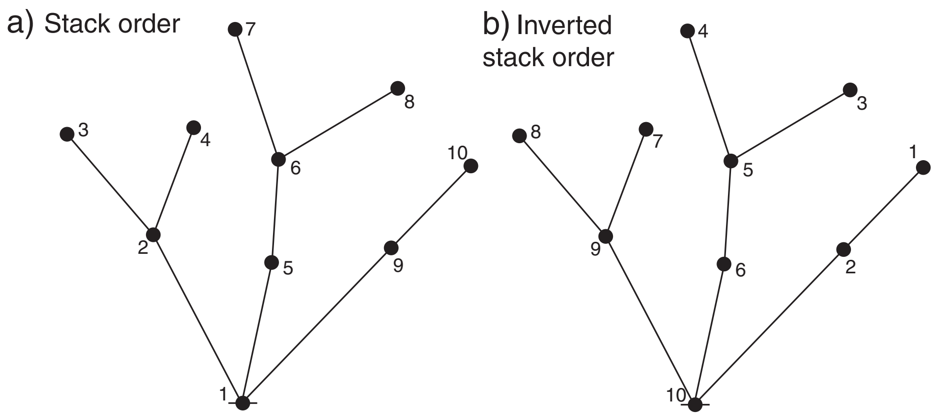

Fig. 2. a) Stack and b) inverted stack order corresponding to the node connectivity and obtained from the donor information described in Fig. 1 and Table 1. At each confluence between two or more channels, the algorithm goes through all donors. After inverting the stack, the resulting order of the nodes is such that, within this catchment, all nodes upstream of a given node are processed before the inverted stack proceeds downstream.

gathering of the substacks into the global stack is not necessary, although it may be needed for other nonessential operations, such as to compute model output.

The order of the nodes on the stack is illustrated in Fig. 2a for a simple catchment made of only 10 nodes. One can see from this example that by inverting the order of the nodes on the stack (Fig. 2b):

$$s(i) \quad \text{for } i = n_p, \cdots, 1, -1 \tag{14}$$

we go through the n_p nodes in an order that is appropriate to compute the contribution of each node to the discharge (or drainage area). Indeed, the inverted stack starts at the top of a hill (a node with no donors) and proceeds downhill following the receiver information. At each confluence, the inverted stack jumps to the top of the 'hill' or subcatchment of each of the tributaries to systematically and progressively compute the contribution to discharge from these subcatchments before proceeding beyond the confluence.

3. Discharge and sediment flux computation

Once the stack is built, the computation of the drainage area or discharge is trivial. It can be expressed in the following manner:

$$\begin{aligned} &\phi(i) = a(i)\nu(i) \quad \text{for } i = 1, \cdots, n_p \\ &\phi(r(s(i))) = \phi(r(s(i))) + \phi(s(i)) \quad \text{for } i = n_p, \cdots, 1, -1 \end{aligned} \tag{15}$$

where $\phi(i)$ is the discharge at node i, and $\nu(i)$ is the net precipitation rate over the area $a(i)$ associated with node i. We see that ν does not need to be uniform, and thus our new algorithm can take into account any spatial variation in rainfall/precipitation such as that resulting from an orographic model. The algorithm could also take into account more complex situations, involving water storage and/or release between adjacent cells.

In the case of uniform rainfall, the precipitation rate can be moved into the constant K and the drainage area only needs to be computed. This is performed in a similar manner:

$$\begin{aligned} A(i) &= a(i) \quad \text{for } i = 1, \cdots, n_p \\ A(r(s(i))) &= A(r(s(i))) + A(s(i)) \quad \text{for } i = n_p, \cdots, 1, -1. \end{aligned} \tag{16}$$

Note that in the case of a rectangular mesh, $a(i) = \Delta x \times \Delta y$, for all i, and the drainage area can be computed by integer summation only.

The same operation can be performed to obtain sediment flux by integrating the net rate of erosion, $\partial h(i)/\partial t$, instead of net precipitation rate:

$$\begin{aligned} q_s(i) &= a(i)\ \phi\ \frac{\partial h(i)}{\partial t} \quad \text{for } i = 1, \cdots, n_p \\ q_s(r(s(i))) &= q_s(r(s(i))) + q_s(s(i)) \quad \text{for } i = n_p, \cdots, 1, -1 \end{aligned} \tag{17}$$

where ϕ is rock porosity and q_s is sediment flux. This flux could be used, in turn, to incorporate sedimentation/transport processes in the stream power equation.

This method is similar to that developed by Freeman (1991) to compute discharge, but the order in which it is performed (starting from the highest nodes) makes it more appropriate for parallelization; furthermore, the memory requirement is lower than for the method proposed by Freeman (1991) as it is bounded by n_p.

4. Solving the equation

The stack can also be used to greatly improve the stability of the solution scheme for the equation. To achieve this, we use a simple first-order finite difference scheme to compute both the spatial derivative (the slope) of the height field and its time derivative (the rate of change of landscape height):

$$\begin{aligned} \frac{\partial h(i)}{\partial x} &= \frac{h(i) - h(r(i))}{\Delta x_i} \\ \frac{\partial h(i)}{\partial t} &= \frac{h^{t+\Delta t}(i) - h^t(i)}{\Delta t} \end{aligned} \tag{18}$$

where Δx_i is the distance between node i and its receiver, $r(i)$, and Δt is the time step in the evolution algorithm.

An explicit first-order finite-difference scheme, where the right-hand side term of the equation (and thus the slope) is computed at time t:

$$\frac{h^{t+\Delta t}(i) - h^t(i)}{\Delta t} = -KA^m \left(\frac{h^t(i) - h^t(r(i))}{\Delta x_i} \right)^n \tag{19}$$

is commonly used for the solution of the evolution Eq. (3) (see Braun and Sambridge (1997) for example); whereas an implicit scheme, where the right-hand side term is computed at time $t + \Delta t$:

$$\frac{h^{t+\Delta t}(i) - h^t(i)}{\Delta t} = -KA^m \left(\frac{h^{t+\Delta t}(i) - h^{t+\Delta t}(r(i))}{\Delta x_i} \right)^n \tag{20}$$

would insure a much greater stability and, consequently, would allow the use of much longer time steps, Δt (Press et al., 1986). However, in the implicit scheme, each equation (for each node i) has two unknowns ($h^{t+\Delta t}(i)$ and $h^{t+\Delta t}(r(i))$), and we are apparently faced with the problem of having to solve a large number of coupled equations as done by Fagherazzi et al. (2002), which is not practical and not $O(n_p)$.

However, noting that the nodes at base level have a fixed and thus known height, we can use the implicit scheme to find the height of all the nodes donors to a base level node. This results from the simple fact that the implicit equation corresponding to these nodes only has one unknown. We can then repeat this procedure and use the implicit scheme for all the nodes that donate flow to base level nodes' donors, and so on. Thus, if we proceed through the coupled equations in the appropriate order, we can solve the system of coupled equations resulting from the implicit scheme in n_p operations. An adequate ordering of the node is the stack order, which for each catchment attached to a base level node works through the nodes starting at base level and progressing from each node to its donor nodes and the donors of its donors, etc.

We can thus write this implicit algorithm in the following way:

$$\frac{h^{t+\Delta t}(s(i))-h^t(s(i))}{\Delta t}=-KA^m\left(\frac{h^{t+\Delta t}(s(i))-h^{t+\Delta t}(r(s(i)))}{\Delta x_i}\right)^n \quad \text{for } i=1,\cdots,n_p. \tag{21}$$

For $n=1$, each of these equations is linear and can be readily solved explicitly:

$$h^{t+\Delta t}(s(i))=\frac{h^t(s(i))+h^{t+\Delta t}(r(s(i)))KA^m\Delta t/\Delta x_i}{1+KA^m\Delta t/\Delta x_i}. \tag{22}$$

In this situation, one can show that the stack order provides the way to order the equations in a way that makes the resulting system of algebraic equations upper (or lower) triangular, i.e., the lower (or upper) half of the resulting matrix is nil. Although, we do not suggest that such a matrix should be constructed, it is worth noting that solving a triangular system of equations is $O(n_p)$, where n_p is the rank of the corresponding matrix.

For $n\neq 1$, each of these equations is nonlinear, but for most commonly used values of n, i.e., $0.5<n<4$ (Lague and Hovius, 2005), finding the root, $h^{t+\Delta t}(s(i))$, of the resulting equation :

$$h^{t+\Delta t}(s(i))-h^t(s(i))+KA^m\Delta t\left(\frac{h^{t+\Delta t}(s(i))-h^{t+\Delta t}(r(s(i)))}{\Delta x_i}\right)^n=0 \tag{23}$$

can be performed by a first-order Newton–Raphson method based on the following iterative scheme:

$$h^{t+\Delta t}(s(i))_k=h^{t+\Delta t}(s(i))_{k-1}-\frac{h^{t+\Delta t}(s(i))-h^t(s(i))+KA^m\Delta t\left(\frac{h^{t+\Delta t}(s(i))-h^{t+\Delta t}(r(s(i)))}{\Delta x_i}\right)^n}{1+\frac{nKA^m\Delta t}{\Delta x_i}\left(\frac{h^{t+\Delta t}(s(i))-h^{t+\Delta t}(r(s(i)))}{\Delta x_i}\right)^{n-1}}$$
$$\text{with } h^{t+\Delta t}(s(i))_0=h^t(s(i)). \tag{24}$$

This algorithm is stable and efficient (convergence reached in less than three or four iterations), as the equation/function (23) is monotonic in the vicinity of the root (see Appendix A).

Note that the drainage area, A, that appears in the right-hand side of Eqs. (22) and (24) is calculated at time t. This means that our method is not fully implicit (with respect to drainage area) but relies on the assumption that the rate of change of the geometry of the drainage network is slow in comparison to the rate of change of the elevation. The one exception to this assumption is the case of large drainage captures. Drainage reorganization will lag one time step behind the erosion rate calculation, which imposes a time step limit for accuracy, which is difficult to quantify.

5. Boundary conditions

In addition to the base level nodes that correspond to a fixed height boundary condition ($h(b_i)=h_0(t)$), two other types of boundary conditions are commonly associated with the solution of the fluvial geomorphic equation. The first type represents a symmetrical or reflecting boundary, where the nodes have a limited number of neighbors that they can potentially drain to; this is by far the simplest boundary condition to implement as it does only require us to limit the number of potential neighbors ($N(i,j)$, for $j=1,\cdots,n_i$) for nodes that are on the boundary.

The second type is a so-called periodic boundary condition that only applies when using a rectangular mesh. In this situation, it is customary to assume that two opposite sides of the mesh communicate with each other in such a way that nodes on one boundary have nodes on the other boundary as neighbors. In this way, what flows across one boundary 're-appears' on the other side.

Both types of boundary conditions can be used in our new algorithm because the boundary conditions only affect the computation of the neighboring information and not the ordering scheme (construction of the stack order).

Finally, in cases where base level is imposed as a prescribed function of time, a simple approach consists of using the value of that function at $t+\Delta t$ in Eq. (22). In cases where that function varies over a time scale that is similar to or smaller than the characteristic response time of the system, a more elaborate implicit–explicit scheme can be used:

$$\frac{h^{t+\Delta t}(s(i))-h^t(s(i))}{\Delta t}=-(1-\alpha)KA^m\left(\frac{h^t(s(i))-h^t(r(s(i)))}{\Delta x_i}\right)^n -\alpha KA^m\left(\frac{h^{t+\Delta t}(s(i))-h^{t+\Delta t}(r(s(i)))}{\Delta x_i}\right)^n \tag{25}$$

to prevent the propagation of spurious topographic waves from the boundary. The best value for α is 0.5, which gives, for $n=1$:

$$h^{t+\Delta t}(s(i))=\frac{h^t(s(i))-KA^m\Delta t/2\Delta x_i\left(h^t(s(i))-h^t(r(s(i)))-h^{t+\Delta t}(r(s(i)))\right)}{1+KA^m\Delta t/2\Delta x_i}. \tag{26}$$

6. Local minima

When computing discharge, many interior nodes, i.e. not situated along a boundary, may be local topographic minima — they are lower than any of their neighbors. For each local minimum, a potentially nonnegligible surface area of the modeled landscape drains into a point that is not on a model boundary and, thus, a nonnegligible proportion of the precipitation does not reach base level. Two solutions exist: neglecting it, which is similar to assuming that an endorheic system has developed (a specific hydrological condition characterized by a positive moisture balance — i.e. in which evaporation or infiltration are sufficient to compensate precipitation), or finding a path for the water to flow toward the boundary by forming one or a series of lakes and determining the 'sill' of each lake, i.e. the lowest point along its boundary.

Although both solutions may be used – and in our implementation, we have left this choice to the user – , the second solution is to be preferred in most situations, especially when modeling systems characterized by 'normal' hydrological conditions, i.e. situations where precipitation wins over evaporation. Furthermore, by imposing that the entire modeled precipitation must drain through the model boundaries, we insure that the model behaves in a predictable way and not in a manner that would become highly dependent on the formation of local minima, a situation that is likely to arise in regions of the modeled landform that are flat and where small random perturbations have been added with the purpose of seeding drainage network growth.

Finding the route followed by water in a landscape characterized by randomly distributed local minima is a difficult problem for which we only propose a practical but not optimal (i.e. $O(n_p)$) solution. In a first step, the algorithm described above is applied using both the base level nodes and the local minima as starting nodes for the construction of the stack. During the construction of the stack (ordering phase), we saw that we can 'tag' each node by propagating through the stack the name (or number) corresponding to the base level node to which it drains. We will call this tag the catchment number, $c(i)$. In the case of a local minimum, we make this tag a negative number.

In a second step, we go through all the nodes looking for nodes that have a negative catchment number and have, as a neighbor, a node that has a positive catchment number. From all those nodes and per catchment, we take the one that is the lowest (lowest topography) and call its neighbor the sill of the catchment, $\chi(-c(i))$. We then force the sill to become the receiver node of the local minima; and all the nodes in the catchment are given, as a tag or catchment number, the catchment number of the sill. In this way, the local minima drains its water into a neighboring catchment in a way similar to what would happen when a lake forms around a local topographic minimum. This takes the following algorithmic form:

$$\chi(-c(i)) = \min_{\text{catchment}} \left(h(i)\ |c(i){<}0\ \&\&\ (c(N(i,j)) > 0,\ \text{for}\ j = 1, \cdots, n_i)\right) \quad \text{for}\ i = 1, \cdots, n_p. \tag{27}$$

The newly found neighbor (or sill) must have a positive catchment number, otherwise a situation could arise where two local catchments drain into each other, which would not solve the local minima problem. Note that the connection between the sill and the local minimum is only used for the computation of the drainage area. It is not used in the solution of the stream power equation.

Unfortunately, not all negative local minima can be dealt with in this way in a single pass of the algorithm, as there will certainly remain negative catchment nodes that do not have a positive catchment neighbor. However, as we apply the algorithm in successive iterations, the number of negative catchment nodes decreases and the algorithm converges, unconditionally. The rate of convergence is highly dependent on the geometry of the problem and is difficult to predict. The maximum number of iterations is likely bounded by $\sqrt{n_p}$, at least on rectangular grids. Numerous other methods have been proposed to remove the local minima from a DEM (Martz and Garbrecht, 1999; Temme et al., 2006; Zhu et al., 2006; Wilson et al., 2008; Hou et al., 2010); but most are also $O(n_p\sqrt{n_p})$.

Clearly, however, the proper computation of drainage, i.e. based on the principle of water conservation, as proposed here will rapidly lead to sills being efficiently eroded away as they receive the discharge accumulating at a local minimum, or potentially at several interconnected local minima. Alternatively, in surface process models where a sediment transport and a deposition law are incorporated, sedimentation leads also very rapidly to the filling in of local minima and their removal from the landscape (see (Braun and Sambridge, 1997), for example). In most situations, only the first time steps of a typical model run are affected by the existence of local minima. This will be illustrated in the following section in which we present the results of numerical experiments performed with the new algorithm.

7. Numerical results

7.1. Scaling with n_p

To illustrate the efficiency of the new algorithm, we ran a series of numerical experiments in which an originally flat landscape is progressively tilted in one direction and affected by fluvial erosion (i.e. following Eq. (2)). The boundary conditions are of three types. The nodes along the $y=0$ boundary are forced to be at base level ($h(i)=0$); the opposite boundary at $y=y_l$ is a reflecting boundary; while the other two facing sides at $x=0$ and $x=x_l$ are periodic. An initial uniform topography is arbitrarily set to 500 m and an uplift rate, U, given by :

$$U = U_0\ x/xl \tag{28}$$

is imposed from $t=0$ with $U_0=5\times10^{-4}$ m/y. In this first set of experiments, $n=1$ and $m=0.5$, while $K=2\times10^{-4}$ 1/y. The value of K is arbitrary and such that steady-state is reached within 10^5 y. The numerical mesh is rectangular and made of $n_x \times n_y(=n_x)$ points; $\nu=1$ m/y, $x_l=100$ km, and $y_l=100$ km, while $\Delta t=1000$ y.

In Fig. 3 the normalized solution time (for one time step) as a function of $n_p=n_x^2$ is shown, showing a perfect linearity and thus $O(n_p)$ scaling.

7.2. Stability of the implicit algorithm

Fig. 4 shows the results of three experiments with $n_x=1000$ and in which the time step has been varied ($\Delta t=10^4, 10^3$ and 10^2 y). The solution is shown at steady-state (for $t=10^5$ y). The solution is stable for all time steps, and its overall final geometry (maximum height, number of catchments, etc.) is almost identical in all four simulations, showing that the algorithm is not only stable but first-order accurate for long-term (steady-state) solutions.

However, to capture the transient behavior of the solution with accuracy, limits must be placed on the time step (a) due to the explicit nature of the scheme in its dependence on drainage area, A, and (b) to satisfy the courant condition for the propagation of knickpoints – or 'topographic waves' – on the landform (see Appendix B for details).

7.3. Nonlinear slope dependence

In Table 2, the computing time for a single step is shown for several runs where n varies between 0.5 and 2, showing that the computational overhead of the nonlinear slope-dependence of erosion rate is limited to approximately a factor of three. This result is a direct consequence of the monotonic behavior of the equation even in cases where $n\neq1$,

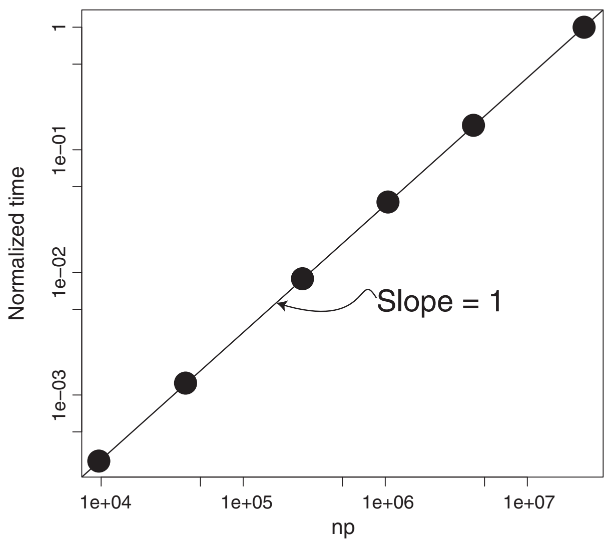

Fig. 3. Solution time on a single processor for one time step as a function of n_p the number of nodes in the mesh, showing that the ordering algorithm and the implicit solution scheme are $O(n_p)$. See Fig. 4 for a description of the problem.

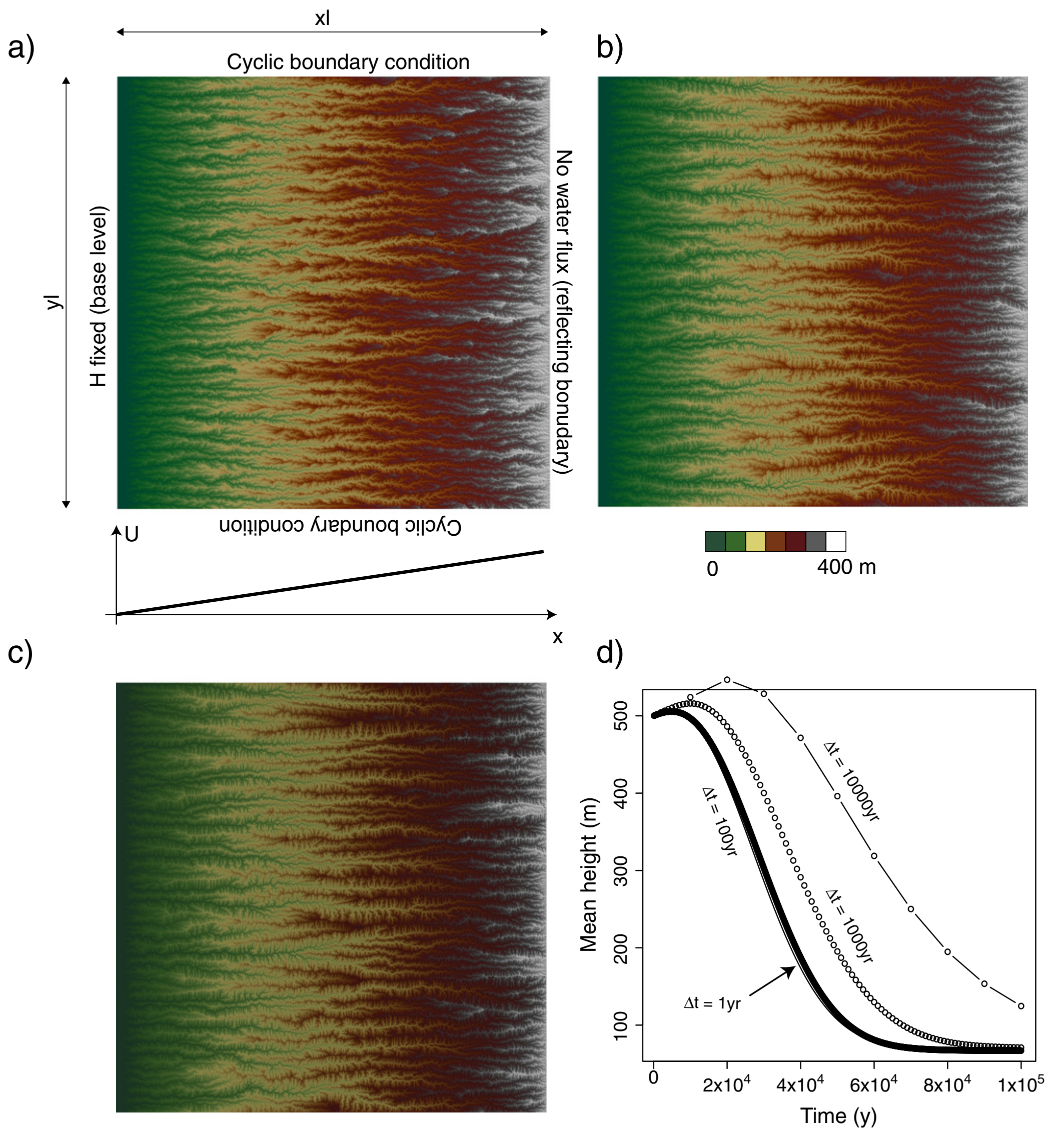

Fig. 4. Three simulations in which the time step length Δt is a) 10,000 y, b) 1000 y and c) 100 y. Results are shown at $t = 100{,}000$ y for all three runs, implying that the number of steps is a) 10, b) 100 and c) 1000. That the results are very similar demonstrates the stability and accuracy of the implicit method. Differences in the details of the solution arise from the explicit nature of the algorithm concerning drainage area A (or discharge). In panel a) are also shown the boundary conditions, and the imposed uplift rate function. d) Evolution of the mean topography for the three model runs and a reference run with $\Delta t = 1y$, demonstrating the accuracy of the scheme (see text for more details).

which insures a fast convergence of the Newton–Raphson scheme (Eq. (24)).

In these experiments, convergence is checked at every point of the landscape such that the height of each node does not vary by more than 1 μm (10^{-6} m) between successive iterations. In most locations, this tolerance is achieved after two iterations, but a third iteration must be performed to verify it.

Table 2
Normalized computation time for a single time step of the algorithm as a function of the slope exponent, n, showing that the Newton–Raphson scheme converges rapidly (1 or 2 iterations at most are sufficient for convergence for the most commonly used values of the exponent n i.e. $0.5 < n < 2$). Even for larger values of $n (= 4)$, the algorithm converges in less than 4 iterations).

$n = 1$	$n = 0.5$	$n = 2$	$n = 4$
1	2.402	2.637	4.117

7.4. Local minima

Fig. 5 shows the evolution of a model run in which the initial topography is uniform, representing a flat plateau, to which a small random perturbation of 10^{-6} m amplitude has been added to seed the growth of the channel network. In this situation, many points on the plateau are local minima. This represents one of the worst-case scenarios that the algorithm can face. Table 3 gives the number of iterations that are required to resolve the local minima and find a route for the water to one of the base level nodes (on the left boundary of the model). Although this number is almost unacceptably high during the first time step (of the order of $\sqrt{n_p}$), it very rapidly decreases to 1 after only 7 time steps, indicating that at this stage in the evolution of the model no local minimum exists on the landscape. This demonstrates that local minima are transient and fast resolving features of any model or natural landscapes and that their overall effect (i.e. over a finite duration model experiment) on computational cost is small.

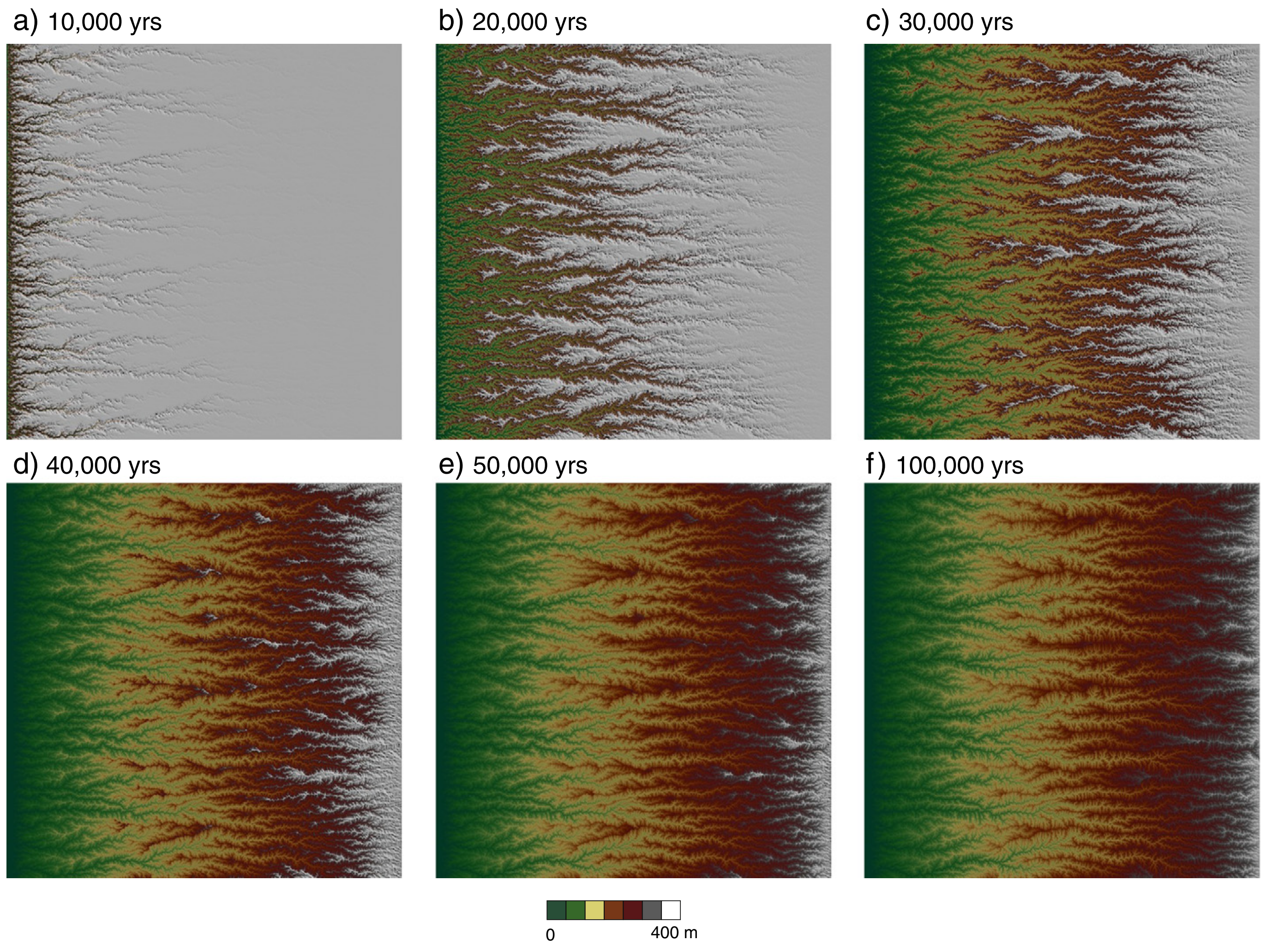

Fig. 5. Time evolution of a model run with an initially flat topography. The solution is shown as a series of snapshots at various time intervals. Note that by 10,000 y in the evolution of the model (panel a), i.e. after 10 time steps, all local minima have been removed.

7.5. Multiprocessor scalability

Fig. 6 shows the computational times obtained to solve one time step as a function of the number of processors or threads used. The algorithm is, in theory, perfectly parallelizable, which should show as a linear relationship between the inverse of computational time and number of processors. As shown in Fig. 6, the relationship is not perfectly linear, as the processors compete among each other to access the memory through the bus. This is not a consequence of the algorithm presented here but is due to hardware limitations that will certainly improve in the not too distant future.

The results shown in Fig. 6 were obtained on an Intel Sandybridge 8 core Xeon processor. For information, a 10,000 × 10,000 node ($n_p = 10^8$) run took 2,7 CPU seconds per time step using the eight cores of the processor.

Table 3
Number of iterations (*niter*) required to resolve all the local minima for the first 7 time steps of the run shown in Fig. 5.

Step	1	2	3	4	5	6	7
niter	249	211	168	105	46	12	1

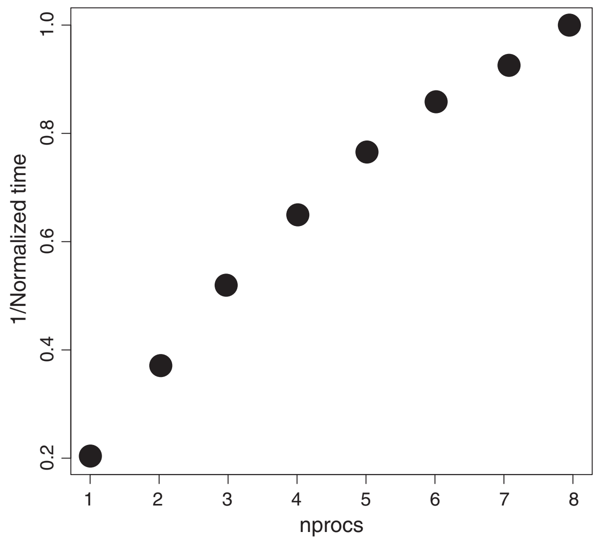

Fig. 6. Inverse of computation time for a single time step (normalized by the time taken using 8 processors) as a function of the number of processors (threads) demonstrating the scalability of the algorithm.

8. Conclusions and future developments

The algorithm presented here solves very efficiently the stream power equation, regardless of problem geometry or boundary conditions. Its main advantages are that the computational time is linearly dependent on the number of nodes (n_p) used to discretize the landform. In a recent review paper on the subject, (Tucker and Hancock, 2010) mentioned under the title 'Computing challenges':

> Computer models of drainage basin evolution have advanced rapidly since the 1980s, but there remain two fundamental technical challenges: speed and geometry. The speed barrier is obvious, and arises from well-known limits on the stability and accuracy of numerical solutions to partial differential equations. Although there has been some exploration of fast solution algorithms (Fagherazzi et al., 2002), the geomorphic equivalent of the fast Fourier transform has yet to be discovered.

The FFT algorithm transformed an $O(n_p^2)$ problem into an $O(n_p log_2 n_p)$ problem (Press et al., 1986). The algorithm presented here has transformed an $O(n_p^2)$ problem into an $O(n_p)$ problem. It could be regarded as an improvement comparable to that brought about by the FFT algorithm in the field of spectral analysis. Furthermore, the stability issues raised by Tucker and Hancock (2010) in his comment have also been addressed by the implicit nature of the new algorithm presented here, which, as we demonstrated, can be used for geomorphic equations that are nonlinear as well as linear function of slopes.

We also demonstrated that the algorithm can be coded to take full advantage of rapidly developing parallel computer architectures. It is, however, limited to SMP (symmetrical multiprocessing) architectures in which memory is shared among processors. Current development in computing technology is, fortunately, heading in this direction with much emphasis being put on multicore processors and multiprocessor motherboards, necessitating the improvement of very fast (or direct) access to memory for each processor/core. The new algorithm is ideally suited and thus ready for this type of technology as demonstrated in Fig. 6. It should also be adaptable for computation on graphics processing units (GPU), a technology that might be better suited to the computation of flow routing (Cheng-Zhi and Lijun, 2012).

A small bottleneck remains, however, in solving the basic stream power equation (or any geomorphic equation that relies on the computation of a direction of water flow): how to solve the routing problem that arises from local minima on the landscape. Here we proposed a solution that is relatively efficient but is not $O(n)$ and will be potentially quite penalizing for very large problems. More work needs to be done to adapt our algorithm with respect to this issue, although, as demonstrated above, the nature of the stream power equation (erosion rate proportional to discharge or drainage area) is such that it tends to naturally remove the local minima by erosion of the sills connecting them to adjacent drainage basins.

Another improvement of the algorithm presented here is critical to solve other types of geomorphic equations, i.e. those that rely on a multirouting of water flow and for which our basic assumption of a single receiver node is not valid. Work is under way to generalize the algorithm presented here to multirouting flow models by adapting the multirouting algorithm of Freeman (1991) to ours. We are also in the process of developing the algorithm to solve more complex fluvial or channelized geomorphic equations, such as those that include thresholds or also predict the effect of sedimentation (e.g. Davy and Lague, 2009). A final, important consideration in modeling landscape evolution involves the parts of the landscape where the stream power equation is not valid, namely, hillslopes, channel heads and upper channels where non-fluvial processes such as debris flows dominate incision (Stock and Dietrich, 2006). Most processes present no issues with the algorithm presented here. Provided the erosion or transport process is local and depends only on the local slope and/or the upstream drainage area, other erosion laws can be implemented directly. For example, a diffusion operator can be applied for local soil creep (e.g. Braun et al., 2001). Perhaps the largest uncertainty in landscape behavior comes from discrete changes in the upstream drainage area, as occur, for example, by river capture. As described here, our method is effectively explicit in drainage area, that is, drainage area is calculated from the previous time step. Both slow migration of water divides and discrete capture events thus lag the erosion calculation. Although direct calculation of discrete capture events and drainage reorganization is possible (Castelltort et al., 2012), this would degrade the efficiency of the scheme.

Acknowledgments

The work described here benefited from support from the Canadian Institute for Advanced Studies. The authors wish to thank G. Tucker and two anonymous reviewers for their very constructive comments on an earlier draft. B. Kaus suggested corrections on an earlier version of the manuscript.

Appendix A. Convergence of Newton–Raphson scheme

We have shown that, using the proper ordering of the nodes, the local solution of the implicit form of the basic landscape evolution equation (Eq. (20)) can be transformed into finding the root of the following function:

$$f\left(h^{t+\Delta t}(s(i))\right) = h^{t+\Delta t}(s(i)) - h^t(s(i)) + KA^m\Delta t\left(\frac{h^{t+\Delta t}(s(i)) - h^{t+\Delta t}(r(s(i)))}{\Delta x_i}\right)^n. \tag{A.1}$$

This is equivalent to finding the root of :

$$f(x) = x - 1 + \alpha x^n \tag{A.2}$$

where :

$$x = \frac{h^{t+\Delta t}(s(i)) - h^{t+\Delta t}(r(s(i)))}{h^t(s(i)) - h^{t+\Delta t}(r(s(i)))} \tag{A.3}$$

and $\alpha = KA^m\Delta t(h^t(s(i)) - h^{t+\Delta t}(r(s(i))))^{n-1}/\Delta x_i^n$. To find the root, we use a Newton–Raphson iterative scheme (Press et al., 1986) taking $h^t(s(i))$ as an initial guess for $h^{t+\Delta t}(s(i))$ or $x = 1$. As the predicted height has to decrease (the equation we solve only applies to erosion) and the final topography cannot be lower than that of the node's receiver, $h^{t+\Delta t}(r(s(i)))$, this implies that $0 < x < 1$. In this range, the function *f(x)* is monotonic, which implies fast convergence. To ensure convergence, one must formally have that :

$$|1-x| < 2\frac{min\frac{\partial f}{\partial x}}{max\frac{\partial^2 f}{\partial x^2}} \quad \text{for} \quad x \in [0,1] \tag{A.4}$$

or :

$$\alpha < \frac{2}{n(n-1)}. \tag{A.5}$$

Appendix B. Limit on time step

The Courant condition has the following form:

$$\frac{u\Delta t}{\Delta x} < C_{max} \tag{B.1}$$

where u is the velocity at which information (here local perturbations in elevation) propagates on the landform. From Eq. (2), we can write :

$$u \approx KA^m \tag{B.2}$$

and, using Hack's Law relating drainage area to distance to the divide, which in our case is the x-coordinate :

$$A = k_s x^p \tag{B.3}$$

with $p \approx 2$ and $k_s \approx 1/4$, we can write :

$$u \approx K k_s^m x^{pm} \tag{B.4}$$

and the Courant condition becomes:

$$\Delta t < C_{max} \frac{\Delta x}{K k_s^m x^{pm}} < C_{max} \frac{\Delta x}{K k_s^m L^{pm}} = C_{max} \frac{10^4}{n_x} \tag{B.5}$$

for $m = 0.5$. Using an explicit scheme, $C_{max} \leq 1$ and the Courant condition is very restrictive, imposing that the time step be less than 10 y. Using the implicit scheme described here, a larger value of C_{max} is acceptable and using a time step of 1000 y produces an accurate solution.

This is illustrated in Fig. 4d where the evolution of the mean topography with time is shown for the three model runs, indicating that the transient solution depends on the time step length. The solution with a very small time step ($\Delta t = 1$ y) is also shown for reference, demonstrating that the transient solution converges for values of $\Delta t \leq 1000$ y.

Appendix C. Code availability

A working version of this algorithm has been incorporated into the *FastScape* fluvial geomorphic model, developed by the authors. It is a user-friendly code that is designed to be used by any geomorphologist interested in testing or quantifying scenarios of landscape evolution. The model is written in Fortran and includes a range of model parameters that can be adapted to the user needs. The code can be obtained by contacting the first author (JB).

References

Braun, J., 2010. The many surface expressions of mantle dynamics. Nature Geoscience 3 http://dx.doi.org/10.1038/ngeo1020.

Braun, J., Sambridge, M., 1997. Modelling landscape evolution on geological time scales: a new method based on irregular spatial discretization. Basin Research 9, 27–52.

Braun, J., Heimsath, A., Chappell, J., 2001. Sediment transport mechanisms on soil-mantled hillslopes. Geology 29, 683–686.

Castelltort, S., Goren, L., Willett, S.D., Champagnac, J.-D., Herman, F., Braun, J., 2012. River drainage patterns in the New Zealand Alps primarily controlled by plate tectonic strain. Nature Geoscience 5 http://dx.doi.org/10.1038/ngeo1582.

Cheng-Zhi, Q., Lijun, Z., 2012. Parallelizing flow-accumulation calculations on graphics processing units—from iterative DEM preprocessing algorithm to recursive multiple-flow-direction algorithm. Computers & Geosciences 43, 1–50.

Crave, A., Davy, P., 2001. A stochastic precipiton model for simulating erosion/sedimentation dynamics. Computers & Geosciences 27, 815–827.

Davy, P., Lague, D., 2009. Fluvial erosion/transport equation of landscape evolution models revisited. Journal of Geophysical Research 114 http://dx.doi.org/10.1029/2008JF001146.

Fagherazzi, S., Howard, A., Wiberg, P., 2002. An implicit finite difference method for drainage basin evolution. Water Resources Research 38, 11–16.

Freeman, T., 1991. Calculating catchment area with divergent flow based on a regular grid. Computers & Geosciences 17, 413–422.

Gallant, J., Wilson, J., 2000. Primary terrain attributes. In: Wilson, J., Gallant, J. (Eds.), Terrain Analysis: Principles and Applications. John Wiley and Sons, New York, pp. 51–85.

Hou, K., Sun, J., Yang, W., Sun, T., Wang, Y., Ma, S., 2010. Extraction algorithms for using regular grid DEMs. The Second International Symposium on Networking and Network Security. Academy Publisher, Jinggangshan, P. R. China, pp. 112–115.

Howard, A.D., 1997. Badland morphology and evolution: interpretation using a simulation model. Earth Surface Processes and Landforms 22, 211–227.

Howard, A., Kerby, G., 1983. Channel changes in badlands. Geological Society of America Bulletin 94, 739–752.

Kooi, H., Beaumont, C., 1994. Escarpment evolution on high-elevation rifted margins: insights derived from a surface processes model that combines diffusion, advection and reaction. Journal of Geophysical Research 99, 12,191–12,209.

Lague, D., 2010. Reduction of long-term bedrock incision efficiency by short-term alluvial cover intermittency. Journal of Geophysical Research 115 http://dx.doi.org/10.1029/2008JF001210.

Lague, D., Hovius, N., 2005. Discharge, discharge variability and the bedrock channel profile. Journal of Geophysical Research 110 http://dx.doi.org/10.1029/2004JF000259.

Martz, L., Garbrecht, J., 1999. An outlet breaching algorithm for the treatment of closed depressions in a raster DEM. Computers & Geosciences 25, 835–844.

O'Callaghan, J., Mark, D., 1984. The extraction of drainage networks from digital elevation data. Computer Vision Graphics 28, 328–344.

Press, W., Flannery, B., Teukolky, S., Vetterling, W., 1986. Numerical Recipes, The Art of Scientific Computing, 1st edition. Cambridge University Press, Cambridge, UK.

Stock, J.D., Dietrich, W.E., 2006. Erosion of steepland valleys by debris flows. Geological Society of America Bulletin 118, 1125–1148.

Temme, A.J.A.M., Schoorl, J.M., Veldkamp, A., 2006. Algorithm for dealing with depressions in dynamic landscape evolution models. Computers & Geosciences 32, 452–461.

Tucker, G., Hancock, G., 2010. State of science: modelling landscape evolution. Earth Surface Processes and Landforms 35, 28–50.

Tucker, G., Slingerland, R., 1994. Erosional dynamics, flexural isostacy, and long-lived escarpments: a numerical modeling study. Journal of Geophysical Research 99, 12,229–12,243.

Turowski, J.M., Lague, D., Hovius, N., 2009. Response of bedrock channel width to tectonic forcing: insights from a numerical model, theoretical considerations and comparison with field data. Journal of Geophysical Research 114 http://dx.doi.org/10.1029/2008JF001133.

Whipple, K.X., 2004. Bedrock rivers and the geomorphology of active orogens. Annual Review of Earth and Planetary Science 32, 151–185.

Whipple, K., Tucker, G., 1999. Dynamics of the stream-power incision model: implications for height limits of mountain ranges, landscape response timescales and research needs. Journal of Geophysical Research 104, 17,661–17,674.

Wilson, J., Aggett, G., Yongxin, D., Lam, C., 2008. Water in the landscape: a review of contemporary flow routing algorithms. Advances in Digital Terrain Analysis 213–236.

Zhu, Q., Tian, Y., Zhao, J., 2006. An efficient depression processing algorithm for hydrologic analysis. Computers & Geosciences 32, 615–623.