water

MDPI

Article

An Optimized and Scalable Algorithm for the Fast Convergence of Steady 1-D Open-Channel Flows

Louis Goffin *, Benjamin Dewals, Sebastien Erpicum, Michel Pirotton and Pierre Archambeau

Hydraulics in Environmental and Civil Engineering (HECE) Research Unit, Urban and Environmental Engineering (UEE) Department, Faculty of Applied Sciences, University of Liege (ULiege), Allée de la Découverte, 9-4000 Liège, Belgium; b.dewals@uliege.be (B.D.); s.erpicum@uliege.be (S.E.); michel.pirotton@uliege.be (M.P.); pierre.archambeau@uliege.be (P.A.)

* Correspondence: l.goffin@uliege.be

Received: 11 October 2020; Accepted: 11 November 2020; Published: 17 November 2020

Abstract: Calculating an open-channel steady flow is of main interest in many situations; this includes defining the initial conditions for the unsteady simulation or the computation of the water level for a given discharge. There are several applications that require a very short computation time in order to envisage a large number of runs, for example, uncertainty analysis or optimization. Here, an optimized algorithm was implemented for the fast and efficient computation of a 1-D steady flow. It merges several techniques: a pseudo-time version of the Saint-Venant equations, an evolutionary domain and the use of a non-linear Krylov accelerator. After validation of this new algorithm, we also showed that it performs well in scalability tests. The computation cost evolves linearly with the number of nodes. This was also corroborated when the execution time was compared to that obtained by the non-linear solver, CasADi. A real-world example using a 9.5 km stretch of river confirmed that the computation times were very short compared to a standard time-dependent computation.

Keywords: shallow water; CasADi; fast computation; 1-D

1. Introduction

Channelized flows can be simulated using 1-D, 2-D or 3-D models depending on the level of flow detail that is required [1]. Results from hydrodynamic numerical models are used in multiple domains including flood risk analysis [2] or real-time control of river facilities [3], for instance.

One-dimensional models are used when a dominant direction can be assumed in the velocity field. This may be the case when the flow is restricted to the main riverbed. A 1-D model can still be used in the case of out-of-bank flooding, although they are unable to represent complex 2-D flow patterns in the floodplain [4]. Several practical cases have shown that flood mapping can be performed using 1-D models [5,6] and they can also be used in other fields, such as flood routing for hydropower plant operations [7] and mixed flows in pipes [8].

In fact, 1-D models are still used extensively even though 2-D and 3-D models are currently widely available. There are various reasons for their use, for example, digital elevation models (DEM) and bathymetry data are not available in some regions of the world. When only cross-section profiles are available to represent the geometry of a riverbed, 1-D models can make direct use of such profiles whereas they have to be interpolated to be used in 2-D or 3-D models. Besides, many applications do not require a detailed description of the flow features in the floodplain, and thus, a 1-D modeling approach is sufficient.

Large-scale hydraulic modeling of river networks [9–13] makes heavy use of 1-D models because of their ability to compute long stretches of the river at a reasonable cost. To initiate a computation, one needs boundary conditions as well as the initial condition. This initial condition is often a steady water profile, which can be obtained by performing a time-dependent simulation with steady boundary conditions over a period of time that is long enough to reach a steady solution [14]. This step may consume a considerable amount of time before the main problem can be addressed. The initial condition should be computed with the same numerical scheme as the one used in the unsteady model in order to ensure the steadiness of the first step of the unsteady model. The ability of these models to quickly obtain a steady initial solution is also of great importance.

Optimization is another field where obtaining a steady result as quickly as possible is important. Indeed, most optimization techniques require a large number of runs in order to figure out the optimal solution. In order to ensure that the overall computation time is as short as possible, techniques that are quick should be utilized including parallelization [15] or the use of fast computing models.

Although 1-D simulations are known to provide results in a short period of time, accelerating the computation of the 1-D steady solution is of great interest in the fields mentioned above. This can be achieved by using two main strategies. The first consists of exploiting the resources of modern computers more efficiently. Such techniques are more frequently applied to 2-D cases, which naturally require more computing resources. Common hardware acceleration strategies include parallelizing codes on several CPU cores [16,17] or on a GPU [18–20]. The second method consists of designing algorithms in order to converge with less effort toward the solution. To the authors' knowledge, there is no such work available in the literature. Both strategies can be combined in order to obtain the best performance.

The purpose of this paper is to propose algorithmic strategies for the fast computation of 1-D steady solutions. First, the equations are presented. Then, we introduce two original strategies in order to reduce the overall computation time. These strategies can easily be implemented in other hydraulic codes. An alternative non-linear solver is introduced for comparison purposes. Finally, the validation results, the optimal parameters for the algorithm, the performance of the model and its application to a real-world problem are presented.

2. Materials and Methods

2.1. Equations

One-dimensional water flow is described by the St-Venant equations, which are as follows [21]:

$$\begin{array}{l} \frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = q_l \\ \frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2}{A}\right) + gA\frac{\partial z_s}{\partial x} = -gAS_f + u_x q_l \end{array} \quad (1)$$

where A is the cross-section area (m^2), Q is the discharge (m^3/s), q_l is a lateral discharge per unit length (m^2/s), g is the gravity acceleration (m/s^2), S_f is the friction slope (-), u_x is the velocity along the x direction of q_l (m/s) and z_s is the free surface elevation (m). The numerical resolution of this set of non-conservative equations has been shown to provide accurate results in many practical cases, including in the presence of discontinuities [22,23].

Assuming a steady flow, temporal derivatives vanish in Equation (1). Since solving the first equation is straightforward, the discharge distribution is known on the entire domain for a given distribution of q_l. We assume that the sign of Q is independent from x. It means that a single equation remains with a single unknown A. In order to keep the same numerical scheme as the one used for the unsteady system (which is important when a steady solution is used as the initial condition of

an unsteady problem and to be able to use a similar algorithmic strategy), Kerger et al. [14] added a pseudo-temporal term (pseudo-time is τ (s)) to the steady form of Equation (1):

$$\beta\frac{\partial A}{\partial \tau}+\frac{\partial}{\partial x}\left(\frac{Q^2}{A}\right)+gA\frac{\partial z_s}{\partial x}=-gAS_f+u_xq_l \tag{2}$$

where $\beta=-\text{sign}(Q)$. Kerger et al. [14] justify this choice for β by analyzing the characteristic velocity of Equation (2):

$$\lambda=\frac{c^2-u^2}{\beta}=\frac{c^2\left(1-\text{Fr}^2\right)}{\beta} \tag{3}$$

where c (m/s) is the wave celerity. In subcritical flows (Fr < 1, Fr $=Q/\left(gA^3/b\right)^{0.5}$ is the Froude number (-), b is the width of the free surface (m)), and the sign of λ is the sign of β. If Fr > 1, then $\text{sign}(\lambda)=-\text{sign}(\beta)$. For critical flows (Fr = 1), the characteristic velocity is 0, independently from β. In order to keep some form of consistency with the general model, if we choose $\beta=-\text{sign}(Q)$, an upstream boundary condition is required when Fr > 1 and a downstream boundary condition is required when Fr < 1. This is equivalent to the position of the water depth boundary condition for the numerical resolution of the full 1-D set of equations.

With a single boundary condition and a discharge distributed in the channel, solving Equation (2) determines the cross-section area (and subsequently, the water depth) all along the stretch. Equation (2) is discretized according to the finite volume method. It is solved according to the same numerical scheme as the one used for the full unsteady model [14,24].

For a node i, Equation (2) is discretized in finite volumes as:

$$\frac{Q_{i+1/2}^2/A_{i+1/2}-Q_{i-1/2}^2/A_{i-1/2}}{\Delta x}+gA_i\left(\frac{z_{s,i+1/2}-z_{s,i-1/2}}{\Delta x}+S_{f,i}\right)-u_{x,i}q_{l,i}\approx-\beta\frac{\partial A}{\partial \tau} \tag{4}$$

where Δx (m) is the spatial discretization step and subscripts refer to the position of variables values.

For the sake of clarity, we consider $Q>0$ and a constant reconstruction of the flux at finite volume boundaries to explicate the numerical scheme. When applying the considered upwinding directions [14] for a node i not located next to a boundary, Equation (4) is equivalent to:

$$\frac{Q_i^2/A_i-Q_{i-1}^2/A_{i-1}}{\Delta x}+gA_i\left(\frac{z_{s,i+1}-z_{s,i}}{\Delta x}+S_{f,i}\right)-u_{x,i}q_{l,i}=\frac{\partial A}{\partial \tau} \tag{5}$$

This flux vector splitting method has been shown to be unconditionally stable [14]. For the node located at the downstream boundary ($i=N-1$), if a weak water level boundary condition $z_{s,BC}$ is imposed at the border, Equation (4) becomes:

$$\frac{Q_{N-1}^2/A_{N-1}-Q_{N-2}^2/A_{N-2}}{\Delta x}+gA_{N-1}\left(\frac{z_{s,BC}-z_{s,N-1}}{\Delta x}+S_{f,N-1}\right)-u_{x,N-1}q_{l,N-1}=\frac{\partial A}{\partial \tau} \tag{6}$$

Without a boundary condition imposed on the value of z_s at the external border, a nil z_s gradient is imposed and Equation (4) becomes:

$$\frac{Q_{N-1}^2/A_{N-1}-Q_{N-2}^2/A_{N-2}}{\Delta x}+gA_{N-1}S_{f,N-1}-u_{x,N-1}q_{l,N-1}=\frac{\partial A}{\partial \tau} \tag{7}$$

At the upstream node ($i=0$), if a weak boundary condition of the water level is imposed at the border, Equation (4) becomes:

$$gA_0\left(\frac{z_{s,1}-z_{s,BC}}{\Delta x}+S_{f,0}\right)-u_{x,0}q_{l,0}=\frac{\partial A}{\partial \tau} \tag{8}$$

Without a boundary condition at the upstream border, Equation (4) becomes:

$$gA_0\left(\frac{z_{s,1}-z_{s,0}}{\Delta x}+S_{f,0}\right)-u_{x,0}q_{l,0}=\frac{\partial A}{\partial \tau} \tag{9}$$

2.2. Original Solving Strategy

Solving Equation (2) instead of Equation (1) decreases the computation time since the number of equations and unknowns is reduced. In order to save even more time, two additional strategies were implemented: (a) a non-linear Krylov accelerator was used to promote fast convergence and (b) the computation was only performed on a sliding part of the full domain.

2.2.1. Non-Linear Krylov Acceleration

Numerically solving a non-linear system can be performed by different means, including Newton's method and Broyden's method [25]. More sophisticated methods exist in order to solve non-linear systems faster, such as the Jacobian-free Newton–Krylov method [26] and Anderson acceleration [27]. The Anderson acceleration method uses the results from successive iterations in order to adapt the new approximation. Walker and Ni [28] showed that this method can be considered equivalent to the well-known GMRES method [29] when applied to linear systems. The nonlinear Krylov acceleration (NKA) [30,31], which is similar to Anderson acceleration, has been found to be more efficient in some applications than more recent methods such as the Jacobian-free Newton–Krylov method [32]. NKA was used for a faster convergence of our pseudo-time model.

Since NKA only relies on the results directly produced by the hydraulic model, it can be easily applied to other algorithms or other domains. Indeed, no gradient evaluation (nor other prerequisite) is required before calling on the NKA algorithm.

The NKA algorithm records N ($N \in \mathbb{N}_{>0}$) previous moves of the root finding process. Based on these previous moves, NKA adapts its guess for the new root. One of the main assumptions is that the Jacobian matrix remains constant within the scope of N moves. We briefly explain the method here and extended details can be found in [30–33].

A Newton–Raphson iteration process computes the $n+1$st guess of the root based on the nth guess $\mathbf{x}_n$, on the value of the function f at $\mathbf{x}_n$ and on the invert of the Jacobian matrix J:

$$\mathbf{x}_{n+1}=\mathbf{x}_n-\mathrm{J}^{-1}f(\mathbf{x}_n) \tag{10}$$

Instead of evaluating J^{-1} at each iteration, NKA evaluates it from N previous moves or sets it to the identity matrix, in which case the method degenerates to the fixed-point method.

NKA takes advantage of a history of N corrections of $\mathbf{x}$ (denoted $\mathbf{v}$) and N evolutions of $f(\mathbf{x})$ (denoted $\mathbf{w}$) at iterate n:

$$\begin{matrix}\mathbf{v}_i=\mathbf{x}_i-\mathbf{x}_{i-1}\\ \mathbf{w}_i=f(\mathbf{x}_i)-f(\mathbf{x}_{i-1})\end{matrix},i=n-N+1,\ldots,n \tag{11}$$

The method assumes that J is constant and invertible within the scope of the N previous iterations, which is written as follows:

$$\begin{matrix}\mathrm{J}\mathbf{v}_i=\mathbf{w}_i\\ \mathbf{v}_i=\mathrm{J}^{-1}\mathbf{w}_i\end{matrix} \tag{12}$$

Mathematical developments described in the references cited above lead to the expression:

$$\begin{matrix}\mathbf{v}_{n+1}=\sum\limits_{i=n-N+1}^{n}z_i\mathbf{v}_i+\left(f(\mathbf{x}_n)-\sum\limits_{i=n-N+1}^{n}z_i\mathbf{w}_i\right)\\ \mathbf{x}_{n+1}=\mathbf{x}_n+\mathbf{v}_{n+1}\end{matrix} \tag{13}$$

where the coefficients z_i are the solution of the projection:

$$\mathbf{z} = \text{argmin}_{a \in \mathbb{R}_0} \left\| f(\mathbf{x}_n) - \sum_{i=n-N+1}^{n} a_i \mathbf{w}_i \right\| \tag{14}$$

Equation (13) shows that the correction of the variable $\mathbf{x}$ is decomposed into two components:

1. The first term depicts the correction as a linear combination of previous corrections.
2. The second term is similar to the second term of a fixed-point iteration that takes into account previous evolutions of function f.

Note that the Jacobian matrix is not used in this iterative process.

2.2.2. Evolutionary Domain

The second strategy was designed to reduce computational cost. It consists in reducing the size of the computation domain and sliding it along the river stretch in order to evaluate the cross-section areas from downstream to upstream.

The development of this strategy has arisen from the long history of numerical hydrodynamics in various flow regimes. Indeed, many flows that are solved for rivers are subcritical (Fr < 1) at downstream and upstream boundary conditions. In such a situation, the cross-section area information is propagated from downstream to upstream. For a fixed flow direction, the upwinding direction of the scheme takes all unknowns and properties (except those linked to the discharge) downstream. This means that when a new node is computed, it depends only on the downstream nodes (Figure 1). It should be noticed that when a node i is added, the upstream border of node $i + 1$ produces a change in the upwinding direction of property Q and the unknown A_{vel} (cross-section area used to compute the velocity). The node $i + 1$, which was supposed to be converged, undergoes a new convergence process that indirectly affects nodes $i + 2$, $i + 3$, ... Theoretically, all the nodes located downstream of node i should be kept in the computational domain.

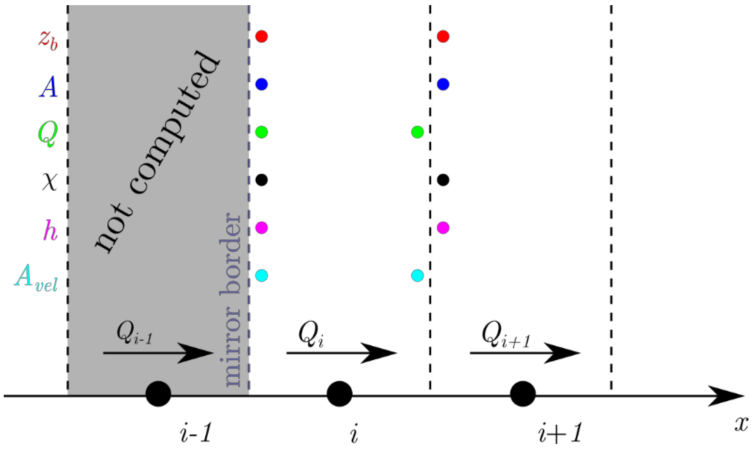


Figure 1. Upwinding directions of the unknown and flow properties at the upstream limit of the computation domain.

The boundary condition on the border between the node i and node $i - 1$ (see Figure 1) is called a "mirror border". On this border, all the unknowns and properties are reconstructed from the computed inner node. This method is equivalent to reproducing the node i in $i - 1$, like a mirror. For node i in Figure 1, the discretization of Equation (2) is:

$$gA_i\left(\frac{z_{s,i+1} - z_{s,i}}{\Delta x} + S_{f,i}\right) - u_{x,i} q_{l,i} = 0 \tag{15}$$

The evolution of the domain relies on three strategies. First, all nodes in the computation domain should have reached a partial residual threshold ξ_p (m^2/s) before considering the extension of the computation domain: $\partial A/\partial\tau \leq \xi_p$. Then, the most downstream node can be removed from the partial domain once it drops under the final residual threshold ξ_f (m^2/s): $\partial A/\partial\tau \leq \xi_f$. Finally, we have to make sure that the nodes that were removed from the domain are not impacted by the computation domain in such a way that their residuals $\partial A/\partial\tau$ become higher than the final precision threshold ξ_f. Dimensionless parameters are discussed further in Section 3.2.

In order to assess whether the removed nodes $\partial A/\partial\tau$ remain lower than ξ_f, an analytical analysis of the discretized model is performed. Let us consider three nodes and their borders (Figure 2). The discretized formulation of Equation (2) at node i is:

$$\begin{aligned}\left[\frac{\partial A}{\partial \tau}\right]_i &= \left[\frac{\partial}{\partial x}\left(\frac{Q^2}{A}\right) + gA\frac{\partial z_s}{\partial x} + gAS_f - u_x q_l\right]_i \\ &\approx \frac{\left(\frac{Q_i^2}{A_i} - \frac{Q_{i-1}^2}{A_{i-1}}\right)}{\Delta x} + g\frac{A_{i+1}^2 - A_i^2}{A_{i+1} - A_i}\left(\frac{z_{s,i+1} - z_{s,i}}{\Delta x}\right) + \left[gAS_f - u_x q_l\right]_i\end{aligned} \tag{16}$$

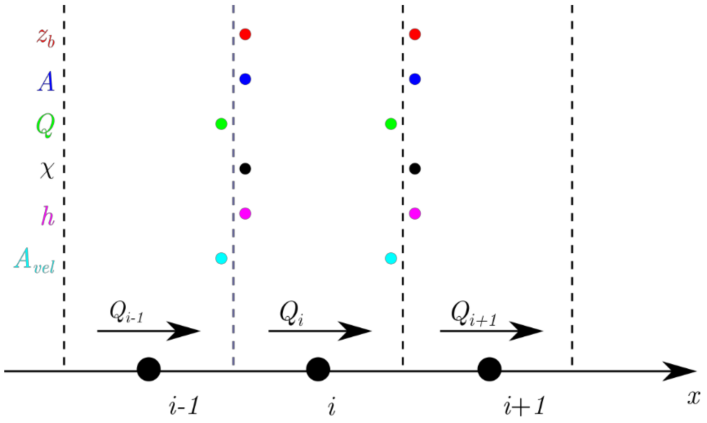


Figure 2. Upwinding directions of the unknown and flow properties in the computation domain.

The influence of a change in the cross-section area in $i-1$ on the value of $\partial A/\partial\tau$ in i is given by the derivative:

$$\frac{\partial}{\partial A_{i-1}}\left(\left[\frac{\partial A}{\partial \tau}\right]_i\right) = \frac{Q_{i-1}^2}{\Delta x}\frac{1}{A_{i-1}^2} \tag{17}$$

From the result in (17), one can predict the evolution of the residual $\partial A/\partial\tau$ of a node that is not in the computation domain anymore. If node i is planned to be removed from the computation domain, one can check that its $\partial A/\partial\tau$ remains below the threshold ξ_f even with an evolution of the cross-section area in $i-1$:

$$\left[\frac{\partial A}{\partial \tau}\right]_{i,next} \approx \left[\frac{\partial A}{\partial \tau}\right]_{i,prev} + \frac{Q_{i-1}^2}{\Delta x}\frac{1}{A_{i-1}^2}\Delta A_{i-1} \tag{18}$$

where ΔA_{i-1} is the evolution of the cross-section A in $i-1$ from the last evaluation of $[\partial A/\partial\tau]_{i,prev}$.

If the evolution of the cross-section area in the computation domain $(i-1)$ implies that $[\partial A/\partial\tau]_i$ exceeds the threshold ξ_f, then node i should be added again in the computation domain in order to make sure it remains below the threshold until the end of the entire computation. It should be noted that node $i+1, i+2, \ldots$ might also be impacted. This technique guarantees that once the sliding domain has moved along the entire domain, all nodes have reached at least the final precision required.

As stated earlier, the method described here was designed within the framework of subcritical flows. In order to be able to deal with a larger range of flow regimes, several adaptations were made. When a supercritical node is detected downstream (let say at position m), it is not computed and

the computation domain is extended until a subcritical node is found upstream (say at position n). Then, the domain starting from m to n is computed and converged. This technique avoids boundary condition problems. Indeed, a supercritical flow requires an upstream BC since the characteristic velocity is directed towards the downstream.

2.2.3. Combination of Solving Strategies

The combined use of the sliding domain and NKA involves several specific considerations. The use of NKA is implemented in the code with a safety coefficient that deactivates this optimizing technique in some cases. Indeed, it was experienced that NKA could lead to some instabilities when there was a sudden change in the cross-section area. This behavior is due to the assumption in NKA that the Jacobian matrix is constant and invertible locally [32,33]. In order to avoid such a situation, the accelerator is deactivated when $\left|A_{i+1} - A_i\right| > \eta A_{i+1}$, where A_i and A_{i+1} refer to the cross-section area of the nodes i and $i + 1$ as depicted in Figure 1, and $0 < \eta \leq 1$ is the safety coefficient (-). After several tests, we found that $\eta = 0.5$ provides stability with a limited impact on the computation time.

The method can be summarized with the following pseudo-code (Algorithm 1):

Algorithm 1

```
Initialize (computation list is empty)
Add most downstream node to computation list
While some nodes still have to be converged (∂A/∂τ > ξ_f):
  Initialize lastly added upstream node
  While nodes of computation list not converged (∂A/∂τ > ξ_p):
    Compute cross section change for each node
    If NKA activated:
      Adapt cross section change with NKA algorithm
  If lastly removed downstream node significantly impacted:
    Add it back to the computation list and do not expand upstream
  Else:
    If downstream node in computation list fully converged (∂A/∂τ ≤ ξ_f):
      Remove this node from computation list
  If upstream nodes remain to be added to computation list:
    Add 1 upstream node to the computation list
Finalize
```

2.3. Alternative Non-Linear Solver

Various techniques can be used to solve nonlinear Equation (2). Up to this point, we have chosen to discretize the equation using a finite volume scheme and solve it with an explicit time scheme, which is consistent with the unified strategy of WOLF [24]. Other techniques can be used, including finite difference schemes and/or implicit time schemes. Another possibility is to use an optimization algorithm for nonlinear systems. One of these is the recently developed CasADi software [34,35].

CasADi first started as an algorithmic differentiation tool. During its evolution, developers chose to shift the focus toward optimization. From non-linear expressions, CasADi is able to generate all the information needed by a nonlinear solver in order to return a solution to the problem. CasADi provides interfaces to MATLAB or Python for easy use.

The purpose of using CasADi is to show how our algorithm performs compared to a state-of-the-art solver.

The implementation in CasADi was done through Opti stack, a collection of helper functions used for nonlinear programming problems. It is possible to define variables to optimize, parameters, an objective function and constraints. The solving of a 1-D steady open channel flow can be done thanks to this framework.

The constraints of the problem are discretized in Equation (2) for each node and a water depth above 0 everywhere. The downstream boundary condition is imposed through Equation (6). If no boundary condition is set, then a flow condition can be imposed through a constraint on the Froude number for the downstream node and the flow head is minimized at the upstream node. If the flow presents a critical section, minimizing the head upstream is equivalent to finding the section with the highest critical head.

Another way to solve a flow with a critical section is to prescribe a Froude number transition fromFr < 1 to Fr > 1 at that critical section. This is done by setting a constraint on the Froude number on the nodes upstream and downstream of the critical section. The identification of the critical section should be done prior to the computation on the basis of a critical head analysis.

3. Results and Discussion

This section presents the validation of the results and focuses on the optimal parameters and performance. The geometries of the tests were different in order to examine as many cases as possible.

3.1. Validation

The validation of the models was performed on a bump placed in a straight horizontal channel, considering three different flow conditions. The bump and channel geometry have been described previously in [36]. The whole domain ranges from 0 to 20 m with the following bed elevation:

$$z_b(x) = \begin{cases} 0.8\left(1 - \frac{(x-10)^2}{4}\right) & 8 \text{ m} \leq x \leq 12 \text{ m} \\ 0 & \end{cases} \tag{19}$$

The channel is considered to be rectangular. The discretization step was chosen as 0.1 m.

The different flow conditions are described in Table 1. All tests were performed with $\xi_p = 10^{-6}$ m^2/s and $\xi_f = 10^{-10}$ m^2/s.

Table 1. Boundary conditions for three test cases.

Test	Upstream BC	Downstream BC
A	$q = 1$ m^2/s	$h = 1.7$ m
B	$q = 0.4$ m^2/s	$h = 0.75$ m
C	$q = 0.4$ m^2/s	Transmissive

The objective was to show that the model is able to deal accurately with various flow regimes and transitions. Test A simulates a fully subcritical flow with no transition. Test B creates a subcritical flow upstream, a subcritical flow downstream and a hydraulic jump in between, downstream of the bump. Finally, the goal of test C is to show the robustness of the method for a downstream supercritical flow and an upstream subcritical flow.

The analytical solutions for tests A, B and C were computed from the Bernoulli principle (head conservation) [22] and the conjugate water depth formula was used for test B. A graphical comparison of the analytical values and numerical results obtained by our algorithm and CasADi is given in Figures 3–5. It appears that the new algorithm and CasADi provide results that fit well with the analytical solution. However, a small difference in energy can be noticed between the analytical solution and the numerical results and also between both numerical methods (see Table 2). This was quantified and explained by Bruwier et al. [22]. Moreover, a more noticeable localized difference appears between CasADi and the new algorithm results at the hydraulic jump. Even if the numerical scheme is the same for the new algorithm and CasADi, each method has its own convergence threshold. Altogether, the analysis validates both models.

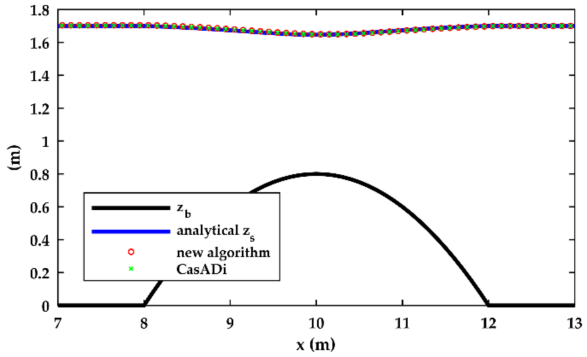


Figure 3. Comparison between the analytical solution and the numerical results produced by the new algorithm and CasADi for validation test A.

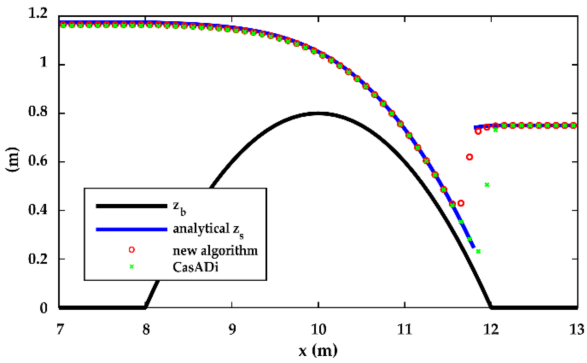


Figure 4. Comparison between the analytical solution and the numerical results produced by the new algorithm and CasADi for validation test B.

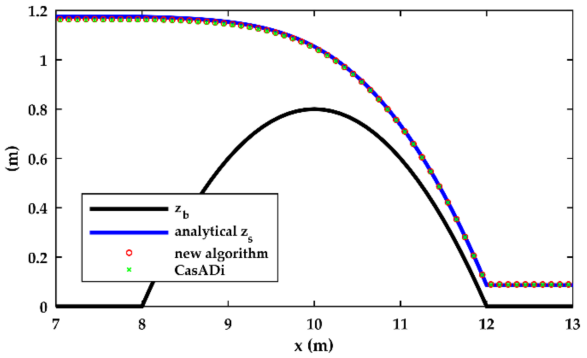


Figure 5. Comparison between the analytical solution and the numerical results produced by the new algorithm and CasADi for validation test C.

Table 2. Upstream head values for tests A, B and C and differential to the analytical value.

Test	Analytical	New Algorithm		CasADi	
	Head (m)	Head (m)	Diff.	Head (m)	Diff.
A	1.71764	1.72147	0.22%	1.71958	0.11%
B	1.18040	1.17062	−0.83%	1.16664	−1.17%
C	1.18040	1.17062	−0.83%	1.16664	−1.17%

3.2. Optimal Setting of the New Algorithm

Five test cases were defined in order to specify the optimal values for the parameters for the new algorithm. These five tests were designed in order to induce changes in the flow characteristics

due to topographic or cross-section variations. The first three tests (1 to 3) concern a channel with a rectangular cross-section and a bed slope that follows a sine function:

$$z(x) = \alpha \sin(\beta \pi x) + \gamma \tag{20}$$

with $x \in [0;20]$, $\alpha = 0.05$, $\gamma = 0.05$, $\beta = 1/2$ (for tests 1 and 2) and $\beta = 2$ for test 3. Two hundred nodes were used to discretize the 20 m long channel, resulting in a 10 cm spatial step. For test 1, the downstream boundary condition is a free surface elevation imposed at 1.2 m. For tests 2 and 3, the same type of boundary condition was imposed with a smaller value of 0.55 m, which results in a higher Froude number downstream. The specific discharge was imposed upstream at 1 m^2/s for tests 1 to 3. The Manning equation was used for friction in tests 1 to 3, with the Manning coefficient n = 0.04 $s/m^{1/3}$.

The topography and the hydraulic solutions were found thanks to the principle of head conservation (Bernoulli) and are shown in Figures 6–8 for tests 1 to 3. Table 3 summarizes the characteristics of each test. The objective of tests 1 to 3 was to analyze the influence of a variation of the bed topography on the behavior of the sliding domain. For test 1, the irregularity of the bed has only a slight influence on the water level. For the other tests, the higher Froude number and less spaced bed elevation variations were meant to investigate the possible influence of oscillations in the water level on the sliding domain performance.

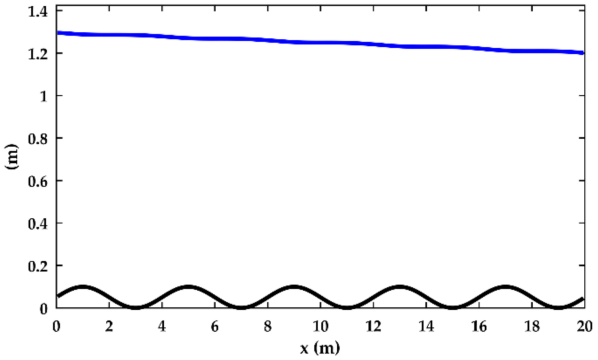


Figure 6. Hydraulic solution for test 1.

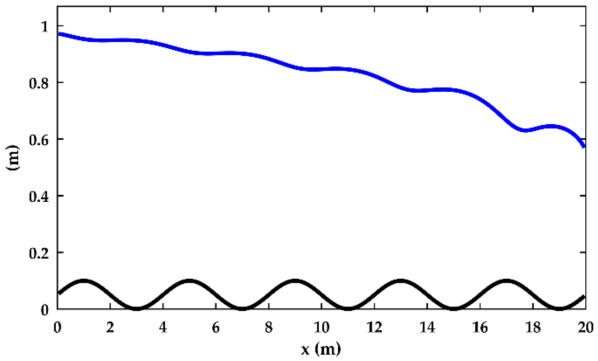


Figure 7. Hydraulic solution for test 2.

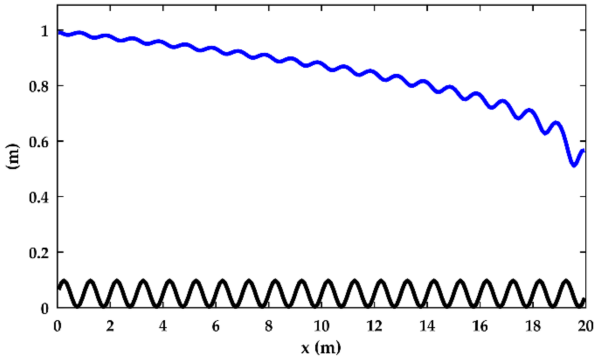


Figure 8. Hydraulic solution for test 3.

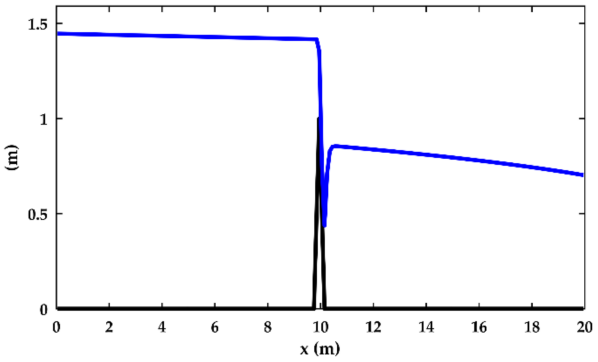
Table 3. Summary of the 5 tests used to investigate the best parameters for the new algorithm.

Test	Channel Bed	Friction	BC
1	$z(x) = 0.05\sin(\pi x/2) + 0.05$, rectangular cross-section	Manning, n = 0.04 s/m$^{1/3}$	q = 1 m/s^2, z_{down} = 1.2 m
2	$z(x) = 0.05\sin(\pi x/2) + 0.05$, rectangular cross-section	Manning, n = 0.04 s/m$^{1/3}$	q = 1 m/s^2, z_{down} = 0.55 m
3	$z(x) = 0.05\sin(2\pi x) + 0.05$, rectangular cross-section	Manning, n = 0.04 s/m$^{1/3}$	q = 1 m/s^2, z_{down} = 0.55 m
4	Flat bottom with a weir, rectangular cross-section	Manning, n = 0.04 s/m$^{1/3}$	q = 1 m/s^2, z_{down} = 0.7 m
5	0.2% slope, sudden change in cross-section	Manning, n = 0.04 s/m$^{1/3}$	q = 1 m/s^2, z_{down} = 0.7 m

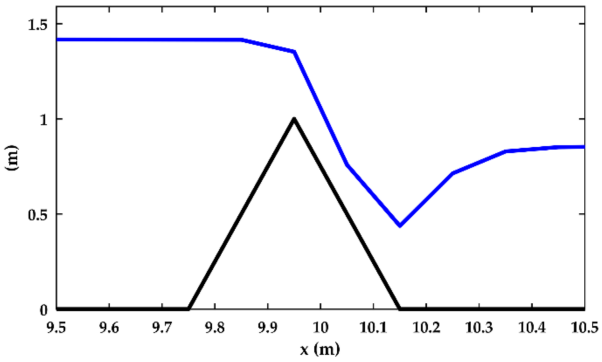
Tests 4 and 5 were performed on regular bottoms. The discontinuities that we wanted to explore here are linked to a change in the flow regime due to the presence of a weir (test 4) or a severe change in the cross-section (test 5). For test 4, the topography was set at z = 0 m for all nodes except for three of them: z = 0.5 m at x = 9.85 m and x = 10.05 m, and z = 1 m at x = 9.95 m. Friction was computed with the Manning formula and a coefficient n = 0.04 s/m$^{1/3}$. The cross-section is uniform along the channel and is rectangular with a width of 1 m. A discharge of 1 m^2/s was injected upstream and a water depth equal to 0.7 m was imposed downstream.

Test 5 deals with a severe change in the cross-section on an inclined bottom. The channel extends 100 m, discretized with 200 nodes. The slope is 0.2%. The cross-section is trapezoidal upstream ($x \leq$ 47.5 m), then suddenly becomes rectangular in the middle of the channel (47.5 m $< x <$ 52.5 m), and finally returns to a trapezoidal shape in the downstream part ($x \geq$ 52.5 m). The trapezoidal sections have a width at the bottom of 2 m and the banks are inclined with an angle of 45°. The rectangular cross-sections are 1 m wide. The friction and boundary conditions are the same as in test 4.

The topography and final water levels for tests 4 and 5 are depicted in Figures 9 and 10. A summary of these tests is given in Table 3.

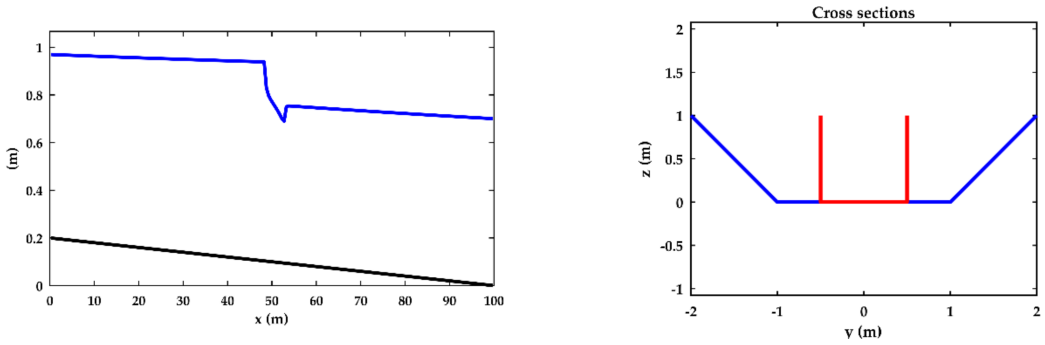


(**a**)



(**b**)

Figure 9. Hydraulic solution for test 4 on the entire domain (**a**) and zoomed on the weir (**b**).



(**a**) (**b**)

Figure 10. Hydraulic solution (**a**) and cross-section change (**b**) for test 5.

Our algorithm includes many parameters that need to be specified. These parameters are the partial residual threshold ξ_p, the final residual threshold ξ_f, the temporal scheme to solve Equation (4) and the coefficient for the deactivation of the nonlinear Krylov accelerator. The final and partial residuals are two closely linked parameters. They also have a direct impact on the computation time. For a given final residual, which has to be parametrized by the user, the partial residual influences the number of iterates required to converge the partial domain and the size of these domains. After investigation, the other two parameters were shown to have almost no influence on the computation time. The

following results focus on the best value to use for the partial residual threshold ξ_p for a fixed value of $\xi_f = 10^{-8}$ m^2/s.

CPU times were measured on a desktop computer (Intel i7 3.5 GHz CPU) for the five tests and various partial residual thresholds. The results are reported graphically in Figure 11. It appears that the overall computation time decreases with an increase in the partial residual threshold. Some stagnation appears around 10^{-2} m^2/s for tests 3 and 5. This can be explained by the fact that the residual naturally decreases at each iteration. Keeping some nodes in the computation domain results in a decrease in the residual for each node included in the computation domain.

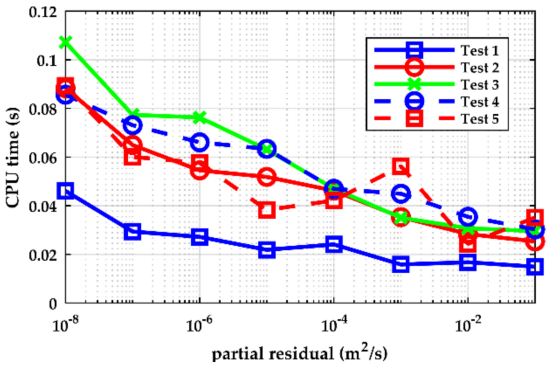


Figure 11. CPU time according to the partial residual threshold for tests 1 to 5.

In order to assess the efficiency of the new algorithm, two scalability tests were performed. The first one consisted of extending the domain upstream, with a constant spatial discretization. The second test consisted of keeping the same channel but refining the discretization and then increasing the number of computation nodes.

The first test took place on a frictionless sine bed elevation described by:

$$z(x) = 0.05 \ \sin(2\pi x) + 0.05 \tag{21}$$

The downstream boundary condition is a water level imposed at 0.65 m. The discharge is constant along the channel stretch and is equal to 1 m^2/s. Cross-sections are rectangular and 1 m wide. Four domain lengths were tested (20 m, 200 m, 2 km and 20 km) with a spatial step of 0.1 m, meaning that these domains include 200, 2000, 20,000 and 200,000 nodes.

The computation results (Figure 12a) showed that the computation time per node decreases when the number of nodes increase. This can be explained by the fact that when the domain gets longer, the flow conditions upstream are smoother than downstream. Longer domains undergo fewer changes than shorter domains, leading to shorter computation time per node. This example shows that higher partial residual values provide the best computation times.

In order to complete this scalability study, we looked at the behavior of the algorithm when the spatial step decreases for a given domain length. In classical explicit schemes, this case leads to a quadratic increase in the computation time. Indeed, when the spatial step decreases, the number of nodes increases and the time step decreases.

A 100 km-long channel with a constant 0.025% slope was chosen to illustrate the behavior of the new algorithm. The cross-sections are trapezoidal and are described using tabular values (1 m wide at the bottom of the section and 5 m wide at 1 m above the bottom). Friction was generated using a Manning law with n = 0.03 s/m$^{1/3}$. The downstream boundary condition is a water level set at 1 m, and 1 m^3/s is injected at the upper node and the injection of 4 m^3/s is shared amongst the other nodes (through the q_l term in Equation (1), $u_x = 0$).

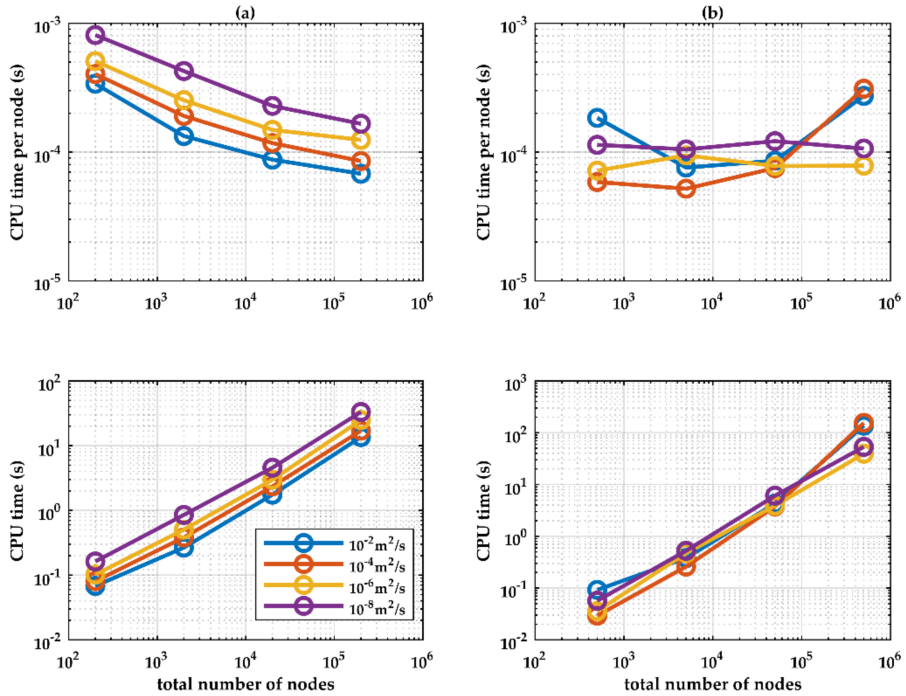


Figure 12. CPU time evolution with the number of nodes when (**a**) the spatial step is kept constant and when (**b**) the domain span is fixed.

The computation times are showed in Figure 12b. The behavior is slightly different from that observed in the previous test. Indeed, the evolution of the computation time is linear in regard to the number of nodes (the CPU time per node is globally constant) when the partial residual is set at 10^{-6} and 10^{-8} m^2/s. For partial residual values of 10^{-2} and 10^{-4} m^2/s, the evolution is linear up to 50,000 nodes; however, for the finest discretization, the computation time increases in a nonlinear way.

This point was investigated further. It appears that at some moment in the computation, the algorithm needs to increase its computation domain size without being able to reduce it quickly (i.e., upstream nodes are added to the computation list while downstream nodes cannot be removed for residual values reasons). This increase in the number of nodes in the computation list was nonlinear compared to the situation with coarser discretization.

We looked at dimensionless values for parameters ξ_p and ξ_f by dividing them by $\sqrt{g}A_0^{3/4}$, A_0, which is a characteristic cross-section area. The results obtained showed rather constant values, suggesting that a coherent choice for $\xi_f / \left(\sqrt{g}A_0^{3/4} \right)$ should be around 10^{-10}. We also found that an efficient ratio ξ_f / ξ_p is around 10^{-6}.

3.3. Performance of the Models

The test case chosen for the comparison between CasADi and our algorithm is a rectangular channel (1 m wide) with a 100 m long sine wave bottom as shown in the following equation:

$$z(x) = 0.05 \sin\left(\frac{\pi}{4}x\right) + 0.05 \tag{22}$$

A Manning friction formula with $n = 0.04$ s/m$^{1/3}$ was used to estimate friction losses. The downstream boundary condition is a water depth equal to 0.6 m. The unit discharge is uniform and is equal to 1 m^2/s. The precision parameters are as follows: $\xi_p = 10^{-6}$ m^2/s and $\xi_f = 10^{-8}$ m^2/s.

We compared the CPU time spent in the solving stages of our new algorithm and CasADi. The goal was to compare the evolution of the computation time with the number of nodes, rather than comparing absolute values. The results are given in Figure 13. They confirm that the computation time evolves

almost linearly with the number of nodes when the new algorithm is used. This is not the case with CasADi: the computation time increases following a power law N^{α}, $\alpha > 1$. This is due to the matrix operations that CasADi has to perform. Increasing the number of nodes leads to a non-proportional increase in computation time. The speed up factor, which is the ratio between the CPU time spent when using our new algorithm and the CPU time spent with CasADi, ranges from in order of 10^0 to in order of 10^2 according to the number of nodes considered.

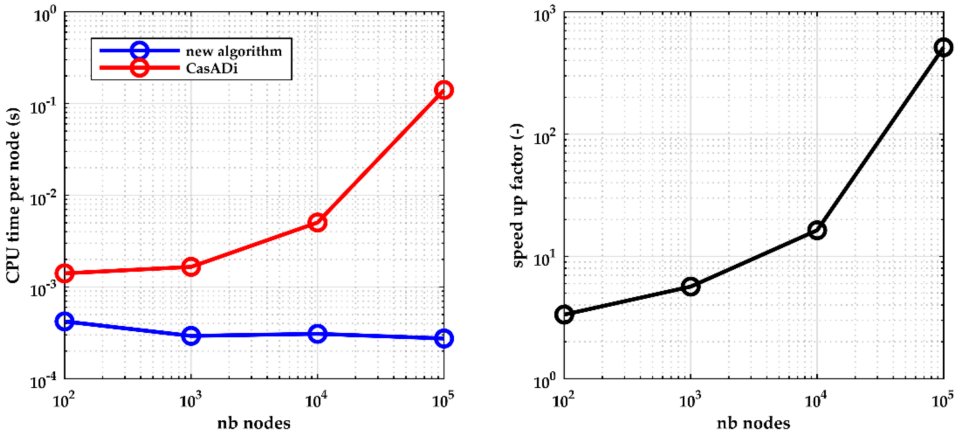


Figure 13. Comparison of the CPU time evolution with the number of nodes between our algorithm and CasADi and the associated speed up factor.

3.4. Real-World Application

The Romanche River in the French Alps is currently facing a number of significant changes. A new hydropower facility is being built in order to replace older power plants. In this context, the dam operator needs a fast computation routine in order to operate its facilities in an optimized and safe way. To do so, an unsteady 1-D model was implemented in Fortran and integrated in a Simulink S-Function in order to be compatible with the operator model [7]. Our new algorithm was used for the fast computation of an initial condition.

The studied part of the Romanche River stretches over 9.5 km (Figure 14), from the new Livet Dam to Gavet. The geometry and the calibration of the friction coefficients are detailed in [7]. The river was discretized with 191 nodes (50 m-long meshes). The downstream boundary condition was set at an elevation of 436.8 m. Two discharges were tested: 10 and 40 m^3/s. Since the details of the hydraulic results are of limited interest for this paper, we focused on the execution time and showed that the Froude number remains under 1 for both discharges (Figure 14). For the same machine as the one used earlier, the execution times (CPU time) are 0.018 s and 0.022 s, respectively ($\xi_f = 10^{-8}$ m^2/s and $\xi_p = 10^{-2}$ m^2/s). In comparison, the computation time of a five-minute full simulation was about 0.2 s. Given the length of the stretch and the celerity of a wave, the time for a wave to travel from downstream to upstream is of the order of 80 min. This means that approximately 3.2 s of computation time are required is order to simulate the propagation of a wave downstream to upstream. It is clear that the gain of time offered by our original technique is significant.

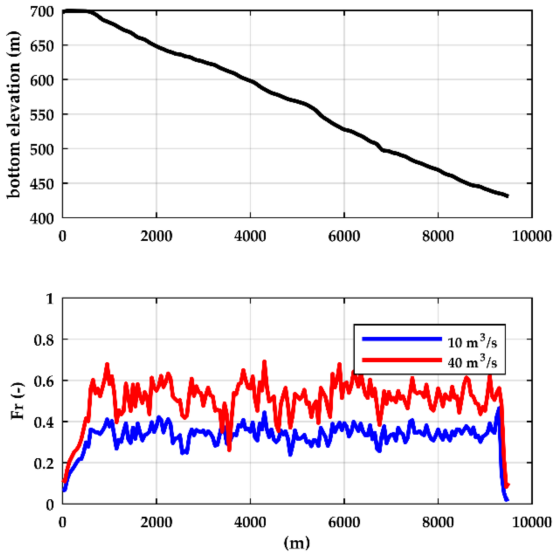


Figure 14. Bottom elevation and Froude number distribution for both simulated discharges.

4. Conclusions

Several innovations are introduced in this paper, including the use of the non-linear Krylov accelerator in open-channel flows, an evolutionary domain algorithm and the use of CasADi to solve steady 1-D flows. These improvements lead to an algorithm that is able to quickly solve steady open-channel flows. Therefore, optimization problems and uncertainty analyses that require many evaluations, become more tractable.

An original algorithm was implemented in order to significantly improve the computation time of a steady 1-D open-channel flow problem. It includes two main optimizing strategies: a non-linear Krylov accelerator and an evolutionary domain algorithm. This new algorithm was validated against the academic benchmarks of flows over a bump. The results showed a good agreement between the numerical and analytical values.

The performance of the suggested algorithm was evaluated against the non-linear optimization software CasADi. It showed good scalability properties. Indeed, the execution time of the proposed algorithm evolves linearly with the number of nodes. This is not the case with other techniques when the mesh is refined and/or when the number of nodes increase.

Finally, we demonstrated the capabilities of our algorithm in a real-world case. We used the optimized algorithm in order to compute quickly the initial condition required by the operational model for the Romanche River in France. Our technique was able to provide a steady state solution to the unsteady model in a very short period of time.

Author Contributions: Conceptualization, M.P. and P.A.; methodology, L.G. and P.A.; software, L.G. and P.A.; validation, L.G.; writing—original draft preparation, L.G.; writing—review and editing, P.A., M.P., B.D. and S.E.; supervision, M.P. and P.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Computer Code and Software: The following software and codes were used for this paper: (a) WOLF was developed by the HECE research group (http://www.hece.ulg.ac.be/cms/) at the University of Liège since 2000 and is not freely available, (b) CasADi is freely available at https://web.casadi.org/, and (c) the routines used to test CasADi are freely available at https://gitlab.uliege.be/HECE/HydroCasADi.

References

1. Proust, S.; Berni, C.; Boudou, M.; Chiaverini, A.; Dupuis, V.; Faure, J.-B.; Paquier, A.; Lang, M.; Guillen-Ludena, S.; Lopez, D.; et al. Predicting the flow in the floodplains with evolving land occupations during extreme flood events (FlowRes ANR project). In Proceedings of the E3S Web of Conferences, Lyon, France, 17–21 October 2016; Volume 7.
2. Drab, A.; Riha, J. An approach to the implementation of European Directive 2007/60/EC on flood risk management in the Czech Republic. *Nat. Hazards Earth Syst. Sci.* **2010**, *10*, 1977–1987. [CrossRef]
3. Schwanenberg, D.; Becker, B.P.J.; Xu, M. The open real-time control (RTC)-Tools software framework for modeling RTC in water resources sytems. *J. Hydroinform.* **2014**, *17*, 130–148. [CrossRef]
4. Tayefi, V.; Lane, S.N.; Hardy, R.J.; Yu, D. A comparison of one- and two-dimensional approaches to modelling flood inundation over complex upland floodplains. *Hydrol. Process.* **2007**, *21*, 3190–3202. [CrossRef]
5. Horritt, M.S.; Bates, P.D. Evaluation of 1D and 2D numerical models for predicting river flood inundation. *J. Hydrol.* **2002**, *268*, 87–99. [CrossRef]
6. Cook, A.; Merwade, V. Effect of topographic data, geometric configuration and modeling approach on flood inundation mapping. *J. Hydrol.* **2009**, *377*, 131–142. [CrossRef]
7. Goffin, L.; Dewals, B.J.; Erpicum, S.; Pirotton, M.; Archambeau, P. Non-linear optimization of a 1-D shallow water model and integration into Simulink for operational use. In *Proceedings of the Sustainable Hydraulics in the Era of Global Change—4th European Congress of the International Association of Hydroenvironment Engineering and Research (IAHR 2016)*; Liege, Belgium, 27–29 July 2016, CRC Press/Balkema: Boca Raton, FL, USA, 2016; pp. 445–451.
8. Bourdarias, C.; Gerbi, S.; Gisclon, M. A kinetic formulation for a model coupling free surface and pressurised flows in closed pipes. *J. Comput. Appl. Math.* **2008**, *218*, 522–531. [CrossRef]
9. Paiva, R.C.D.; Collischonn, W.; Tucci, C.E.M. Large scale hydrologic and hydrodynamic modeling using limited data and a GIS based approach. *J. Hydrol.* **2011**, *406*, 170–181. [CrossRef]
10. Paz, A.R.; Bravo, J.M.; Allasia, D.; Collischonn, W.; Tucci, C.E.M. Large-scale hydrodynamic modeling of a complex river network and floodplains. *J. Hydrol. Eng.* **2010**, *15*, 152–165. [CrossRef]
11. Lai, X.; Jiang, J.; Liang, Q.; Huang, Q. Large-scale hydrodynamic modeling of the middle Yangtze River Basin with complex river-lake interactions. *J. Hydrol.* **2013**, *492*, 228–243. [CrossRef]
12. Remo, J.W.F.; Pinter, N. Retro-modeling the Middle Mississippi River. *J. Hydrol.* **2007**, *337*, 421–435. [CrossRef]
13. Biancamaria, S.; Bates, P.D.; Boone, A.; Mognard, N.M. Large-scale coupled hydrologic and hydraulic modelling of the Ob river in Siberia. *J. Hydrol.* **2009**, *379*, 136–150. [CrossRef]
14. Kerger, F.; Archambeau, P.; Erpicum, S.; Dewals, B.J.; Pirotton, M. A fast universal solver for 1D continuous and discontinuous steady flows in rivers and pipes. *Int. J. Numer. Methods Fluids* **2011**, *66*, 38–48. [CrossRef]
15. Sandric, I.; Ionita, C.; Chitu, Z.; Dardala, M.; Irimia, R.; Furtuna, F.T. Using CUDA to accelerate uncertainty propagation modelling for landslide susceptibility assessment. *Environ. Model. Softw.* **2019**, *115*, 176–186. [CrossRef]
16. Neal, J.C.; Fewtrell, T.J.; Bates, P.D.; Wright, N.G. A comparison of three parallelisation methods for 2D flood inundation models. *Environ. Model. Softw.* **2010**, *25*, 398–411. [CrossRef]
17. Lacasta, A.; Garcia-Navarro, P.; Burguete, J.; Murillo, J. Preprocess static subdomain decomposition in practical cases of 2D unsteady hydraulic simulation. *Comput. Fluids* **2013**, *80*, 225–232. [CrossRef]
18. Brodtkorb, A.R.; Sætra, M.L.; Altinakar, M. Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. *Comput. Fluids* **2012**, *55*, 1–12. [CrossRef]
19. Petaccia, G.; Leporati, F.; Torti, E. OpenMP and CUDA simulations of Sella Zerbino Dam break on unstructured grids. *Comput. Geosci.* **2016**, *20*, 1123–1132. [CrossRef]
20. Smith, L.S.; Liang, Q. Towards a generalised GPU/CPU shallow-flow modelling tool. *Comput. Fluids* **2013**, *88*, 334–343. [CrossRef]
21. Chaudhry, M.H. *Open-Channel Flow*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2007.
22. Bruwier, M.; Archambeau, P.; Erpicum, S.; Pirotton, M.; Dewals, B. Discretization of the divergence formulation of the bed slope term in the shallow-water equations and consequences in terms of energy balance. *Appl. Math. Model.* **2016**, *40*, 7532–7544. [CrossRef]

23. Franzini, F.; Soares-Frazão, S. Efficiency and accuracy of Lateralized HLL, HLLS and Augmented Roe's scheme with energy balance for river flows in irregular channels. *Appl. Math. Model.* **2016**, *40*, 7427–7446. [CrossRef]
24. Erpicum, S.; Dewals, B.; Archambeau, P.; Pirotton, M. Dam break flow computation based on an efficient flux vector splitting. *J. Comput. Appl. Math.* **2010**, *234*, 2143–2151. [CrossRef]
25. Broyden, C.G. A class of methods for solving nonlinear simultaneous equations. *Math. Comput.* **1965**, *19*, 577–593. [CrossRef]
26. Knoll, D.A.; Keyes, D.E. Jacobian-free Newton–Krylov methods: A survey of approaches and applications. *J. Comput. Phys.* **2004**, *193*, 357–397. [CrossRef]
27. Anderson, D.G. Iterative Procedures for Nonlinear Integral Equations. *J. ACM* **1965**, *12*, 547–560. [CrossRef]
28. Walker, H.F.; Ni, P. Anderson acceleration for fixed-point iterations. *SIAM J. Numer. Anal.* **2011**, *49*, 1715–1735. [CrossRef]
29. Saad, Y.; Schultz, M.H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* **1986**, *7*, 856–869. [CrossRef]
30. Carlson, N.N.; Miller, K. Design and application of a gradient-weighted moving finite element code I: In one dimension. *SIAM J. Sci. Comput.* **1998**, *19*, 728–765. [CrossRef]
31. Carlson, N.N.; Miller, K. Design and Application of a Gradient-Weighted Moving Finite Element Code II: In Two Dimensions. *SIAM J. Sci. Comput.* **1998**, *19*, 766–798. [CrossRef]
32. Calef, M.T.; Fichtl, E.D.; Warsa, J.S.; Berndt, M.; Carlson, N.N. Nonlinear Krylov acceleration applied to a discrete ordinates formulation of the k-eigenvalue problem. *J. Comput. Phys.* **2013**, *238*, 188–209. [CrossRef]
33. Wang, C.; Cheng, J.; Berndt, M.; Carlson, N.N.; Luo, H. Application of nonlinear Krylov acceleration to a reconstructed discontinuous Galerkin method for compressible flows. *Comput. Fluids* **2018**, *163*, 32–49. [CrossRef]
34. Andersson, J.A.E.; Gillis, J.; Horn, G.; Rawlings, J.B.; Diehl, M. CasADi—A software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* **2018**, *11*, 1–36. [CrossRef]
35. Baayen, J.; Piovesan, T.; VanderWees, J. Optimization problems subject to the nonlinear semi-implicitly discretized Saint-Venant equations have a unique solution. *arXiv* **2018**, arXiv:1801.06507.
36. Aureli, F.; Maranzoni, A.; Mignosa, P.; Ziveri, C. A weighted surface-depth gradient method for the numerical integration of the 2D shallow water equations with topography. *Adv. Water Resour.* **2008**, *31*, 962–974. [CrossRef]