water

MDPI

Article

Computationally Efficient Solution of a 2D Diffusive Wave Equation Used for Flood Inundation Problems

Wojciech Artichowicz * and Dariusz Gąsiorowski

Faculty of Civil and Environmental Engineering, Gdansk University of Technology, 80-233 Gdansk, Poland; gadar@pg.edu.pl

* Correspondence: wojartic@pg.edu.pl; Tel.: +48-58-347-28-32

Received: 29 August 2019; Accepted: 18 October 2019; Published: 22 October 2019

Abstract: This paper presents a study dealing with increasing the computational efficiency in modeling floodplain inundation using a two-dimensional diffusive wave equation. To this end, the domain decomposition technique was used. The resulting one-dimensional diffusion equations were approximated in space with the modified finite element scheme, whereas time integration was carried out using the implicit two-level scheme. The proposed algorithm of the solution minimizes the numerical errors and is unconditionally stable. Consequently, it is possible to perform computations with a significantly greater time step than in the case of the explicit scheme. An additional efficiency improvement was achieved using the symmetry of the tridiagonal matrix of the arising system of nonlinear equations, due to the application of the parallelization strategy. The computational experiments showed that the proposed parallel implementation of the implicit scheme is very effective, at about two orders of magnitude with regard to computational time, in comparison with the explicit one.

Keywords: floodplain inundation; diffusive wave equation; computational efficiency; splitting method; finite element method; parallel computations

1. Introduction

In river valleys protected by embankments, floods can occur in adjacent areas due to a dike break or during controlled inflows of flood water into polders. Very often the flood covers a large area and the transition time of the flood wave through the river valley can reach even up to a few weeks. Consequently, the mathematical modeling of the flood wave propagation becomes a challenging problem because the governing equations have to be numerically integrated over a vast area for a long simulation time. For this reason, it is very important to elaborate on a relatively simple model, which is able to simultaneously ensure adequate accuracy of the solution within a reasonable time frame.

An unsteady flow over a floodplain can be simulated using the two-dimensional shallow water equation (SWE) [1–5]. The methodology of the solution of the SWE is well recognized for open channels, shallow reservoirs, and wetlands initially covered with a water layer [6]. However, usually a floodplain is initially dry and the SWE must be solved in a domain changing over time as a wave propagates. Consequently, the solution domain is bounded by a moving wave front that separates dry and wet areas. In order to avoid the mentioned problem, the numerical solution of the SWE requires special algorithms that are often based on the finite volume method (FVM) [3,4,7,8] or the finite element method (FEM) [9,10]. The numerical algorithms of FVM as well as FEM lead to satisfactory results when additional modifications are used for flow over an initially dry area. However, the applied modifications can decrease the accuracy of the obtained solution, and simultaneously increase the complexity of the numerical algorithm. Additionally, they become more time-consuming. For these reasons, a simplified model in the form of a diffusive wave have been developed.

The application of the diffusive wave model ensures a sufficiently accurate solution on condition that the assumptions used for its derivation are fulfilled [11,12], i.e., when the wave propagating over the floodplain is characterized with low dynamics dominated by the forces of gravity, pressure, and friction. Only in cases of a sudden break of a dam or a dike, the provided solutions are less accurate. However, even in these cases, except for in the vicinity of the breach, it is possible to achieve proper accuracy using the diffusive wave model [13–15].

The most commonly used method to solve the two-dimensional nonlinear diffusive wave equation is the finite difference method (FDM). This approach, in the form of the nodal domain integration method, was applied for rectangular mesh by Hromadka and Yen [16] and Han et al. [17]

However, it seems that FEM and FVM are particularly suitable for solving two-dimensional equations since they ensure very flexible spatial discretization. As far as FVM is concerned, it was applied for the solution of the diffusion wave equation by Lal [18] and by Cea et al. [19]. Prestinizi [14] developed FVM in combination with the storage cell scheme, whereas Di Giammarco et al. [20] presented a formulation for a mixed one-dimensional (1D) and two-dimensional (2D) overland flow using conservative volume FEM. FEM with triangular unstructured mesh was successfully applied by Arico et al. [21], whereas Szymkiewicz and Gąsiorowski proposed the modification of the standard Galerkin FEM with triangular elements for a structured grid [22].

Apart from the accuracy and stability of the applied numerical scheme, an important problem is also the computational efficiency, which can be defined as the number of arithmetic operations carried out at a certain time in order to obtain a numerical solution with assumed accuracy. This issue deserves special attention in terms of simulating real scenarios for unsteady flows over vast floodplains. The discretization of such domains results in significant sizes of computational grids. Consequently, a large system of algebraic nonlinear and linear equations must be solved at each time level within the simulation period. In the resulting system of algebraic equations, the matrix of coefficients is banded with non-zero elements located along the main diagonal. The corresponding coefficient matrix usually has a very wide bandwidth whose form depends on the structure of the assumed mesh. Thus, for flow over a vast floodplain, a significant amount of random access memory (RAM) must be assigned.

The computational efficiency can be improved by applying the dimensional splitting method, which is also known as the fractional-step method [23,24]. According to the dimensional decomposition performed for a rectangular grid of nodes, the solution of the 2D equation at a given time level is reduced to the solution of a set of 1D equations for each of the directions of the coordinate system. As a result, the splitting technique leads to systems of algebraic equations with a tridiagonal coefficient matrix. The resulting system can be solved more efficiently than a system with the matrix obtained using the unsplit algorithm with triangular elements. For the 2D diffusive wave equation, such a splitting strategy has been adopted by Neal et al. [25], Yu [26], and Gąsiorowski [27]. Moreover, Hsu et al. presented splitting in the form of an alternative direction explicit (ADE) scheme [28].

The computational efficiency is also related with choice of the time discretization scheme used for solving the governing equation. The most popular methods applied for time discretization in flood inundation modeling are based on explicit schemes [29]. This type of approximation has been used mainly with the storage cell method [14,15,30–32] as well as during the time integration of the system of ordinary differential equations arising after spatial discretization using FVM [8,33,34]. However, it must be noticed that the explicit schemes are conditionally stable. For numerical stability, the Courant condition has to be respected. To fulfil the Courant condition, a very small time step is required in comparison with the run time of real wave propagation (for example, the order of 0.01 s for a spatial interval of about 10 m). Consequently, this leads to considerable lengthening of the calculation time. On the other hand, explicit schemes are usually very easy to parallelize and to implement, because they lead to sequential computations that do not require a solution of systems of algebraic equations.

An acceleration of the computational process can be achieved by modifying the explicit scheme in terms of a variable time step [35–37]. Such an approach allows the length of the time step to be adjusted, according to the stability condition, resulting from the grid resolution and the current flow

conditions. However, in the case of wave propagation problems, the time step can be maintained with a reasonable length only for part of the simulation period, whereas, for the rest of this period, it must be reduced to very small values.

For time integration, the implicit scheme is preferred over explicit ones. The implicit scheme is unconditionally stable and allows a much larger time step to be maintained, which reduces the total number of steps. Consequently, this can make the calculation more efficient. The implicit method, in the form of a two-level difference scheme, was applied for the solution of the diffusive wave equation in the context of a 2D flow [22] as well as a 1D flow [27]. It is interesting that, for the solution of the diffusive wave equation, the implicit methods are seldom applied despite their clear advantage over the explicit schemes. It seems that such a situation can be explained by the fact that the application of the implicit scheme leads to a system of nonlinear algebraic equations that must be solved at each time step. Consequently, the solution of the resulting nonlinear system requires an iterative method, which can complicate the numerical algorithm.

A parallel implementation of solvers for a two-dimensional flow described by the diffusive wave approach has been presented only for explicit schemes [25,26,32,34]. However, using explicit schemes introduces very strict constraints on the computational time step value, and, thus, makes the algorithms very ineffective. As far as implicit schemes are considered, the parallelization for the solution of the 2D diffusive wave equation has not been applied until now. Although many different numerical techniques are used for the solution of this equation, it seems that it is possible to propose a new one, which will be much more efficient. The proposed approach is based on the construction of an algorithm tailored for parallel execution. In such a case, the whole solution approach has to be designed from its basis with the goal of parallelization in mind. To this order, possible decompositions should be applied, and the proper method of the discretization of differential equations has to be chosen. Such a proceeding allows an algorithm to be obtained that can be processed in parallel. Although this approach is much more complicated than the sequential one, it usually results in algorithms that are very efficient. The approach considered in this study deals with the latter category.

This paper focuses on two main aspects of the solution of a 2D diffusive wave equation for floodplain inundation problems. The first concerns improvements for efficiency of the numerical solution, by concentrating on the use of the implicit scheme in parallel computations. For this purpose, in the proposed algorithm, a high computational efficiency was achieved by the application of the dimensional splitting technique and the implicit time integration scheme, which is unconditionally stable, as well as the modified Picard method for the solution of the nonlinear systems of equations, which ensures convergent iterations. Further improvements were achieved using the symmetry of the tridiagonal matrix in the system of linear equations. The second aspect relates to comparing the computational efficiency as well as accuracy of the proposed method with the commonly used algorithm based on the explicit scheme.

2. Methods

In this work, the 2D diffusive wave model is described by a single nonlinear equation, as follows [16,38].

$$\frac{\partial H}{\partial t} - \frac{\partial}{\partial x}\left(K_x \frac{\partial H}{\partial x}\right) - \frac{\partial}{\partial y}\left(K_y \frac{\partial H}{\partial y}\right) = \Phi \tag{1}$$

where t is the time variable (T), x and y are the space co-ordinates (L), H is the water surface elevation above the assumed datum (L), $\Phi = P - I - E$ is the source/sink term, which usually involves rainfall (P), infiltration (I), and evaporation (E) (L·T^{-1}). The parameters K_x and K_y introduced in Equation (1) are called the hydraulic diffusion coefficients (L^2·T^{-1}). They are given by the formulas below.

$$K_x = \frac{h^{5/3}}{n_x}\left|\frac{\partial H}{\partial s}\right|^{-1/2} \tag{2a}$$

$$K_y = \frac{h^{5/3}}{n_y}\left|\frac{\partial H}{\partial s}\right|^{-1/2} \tag{2b}$$

where $h = H - z$ is the flow depth (L), z is the bottom elevation above the assumed datum (L), and n_x and n_y are the Manning roughness coefficients in x and y direction, respectively ($L^{-1/3}$·T). The term $\partial H/\partial s$ is the derivative of the function H with respect to the flow direction s, which is defined as follows.

$$\frac{\partial H}{\partial s} = \sqrt{\left(\frac{\partial H}{\partial x}\right)^2 + \left(\frac{\partial H}{\partial y}\right)^2} \tag{3}$$

Equation (1) is written in a conservative (divergent) form ensuring the consistency with the mass conservation principle represented in the SWE system by the continuity equation. It is worth adding that a nonlinear diffusive wave equation with the water surface elevation H, as a dependent variable, is conservative. However, as has been presented by Gąsiorowski and Szymkiewicz [39] for 1D flood routing problems, the same equation with discharge Q satisfying mass principle cannot be derived.

In the proposed numerical algorithm, the computational efficiency is improved by applying: (1) the dimensional decomposition of the 2D governing equation in order to avoid allocating significant memory resources in computers, (2) unconditionally stable implicit scheme instead of explicit schemes that are conditionally stable, (3) effective methods for solving the system of nonlinear and linear algebraic equations, where a property of the symmetrical banded matrix is used, and (4) parallelization methods utilizing multi-core computer architectures.

2.1. Decomposition of the 2D Equation with the Splitting Method

The dimensional decomposition technique is very usable and effective in the case of overland flow modelling. It is particularly suitable for parallel implementation. According to this technique, the 2D diffusive wave equation (Equation (1)) at a given time level is split into a set of 1D equations for each of the directions [27]. This leads to two equations describing the wave propagation process in directions x and y, respectively.

$$\frac{\partial H^{(1)}}{\partial t} - \frac{\partial}{\partial x}\left(K_x \frac{\partial H}{\partial x}\right) = 0 \tag{4}$$

$$\frac{\partial H^{(2)}}{\partial t} - \frac{\partial}{\partial y}\left(K_y \frac{\partial H}{\partial y}\right) = 0 \tag{5}$$

Indexes in the superscript of the H variable denote the solution stage resulting from the decomposition. In the first stage, Equation (4) is solved with the initial condition $H^{(1)}(x,y,t) = H(x,y_i, t = t_n)$, where $i = 1, \ldots, N$, with N denoting the number of spatial steps in the y direction and n denoting the time step index. As a result of this step, the water stage $H^{(1)}(x,y,t_{n+1})$ is obtained. Then, at the second stage, Equation (5) is solved with the initial condition $H^{(2)}(x,y,t) = H^{(1)}(x_j,y,t_{n+1})$, where $j = 1, \ldots, M$, with M denoting the number of spatial steps in the x direction. The example of a computational grid is displayed in Figure 1.

It is very important that, in the considered case of wave propagation over a floodplain, the directional decomposition allows the use of a rectangular grid of nodes. The only requirement is that the rectangular grid has to cover the entire area where the propagating flood wave is expected.

While solving the diffusive wave equation, the decomposition procedure can be applied with respect to the physical process as well. The physical decomposition leads to the decoupling of the flow over the floodplain from physical processes such as infiltration or evaporation, which is represented by the source/sink term Φ in Equation (1). This additional decomposition at the third stage leads to a

family of differential equations describing the variability of the water stages over time in terms of the intensity of the infiltration, precipitation, and evaporation.

$$\frac{\partial H^{(3)}}{\partial t} = \Phi \tag{6}$$

Equation (6) is solved with the initial condition $H^{(3)}(x,y,t) = H^{(2)}(x,y,t_{n+1})$ for all nodes. After the last stage, the water stage values at the next $(n + 1)^{\text{th}}$ time step are obtained.

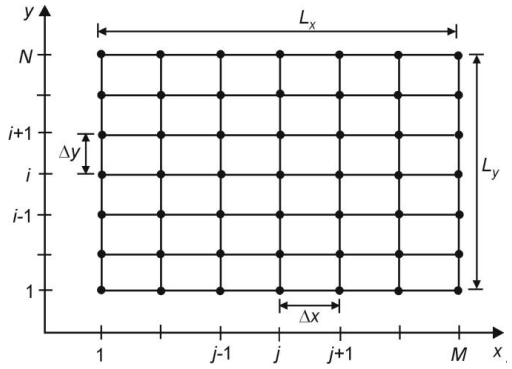

Figure 1. Spatial computational grid.

2.2. Solution of the 1D Equation Using the FDM Explicit Scheme

While solving the diffusion wave equation, an explicit finite difference scheme is usually applied. In such a case, the considered 1D nonlinear equation can be approximated using the forward difference for the time derivative and a combination of backward and forward differences for the diffusion term. As a result, for Equation (4), the following scheme for nodes $j = 2, 3, \ldots, M - 1$ is obtained.

$$\frac{H_j^{n+1} - H_j^n}{\Delta t} - \frac{\left[\overline{K}_j^n\left(H_{j+1}^n - H_j^n\right) - \overline{K}_{j-1}^n\left(H_j^n - H_{j-1}^n\right)\right]}{\Delta x^2} = 0 \tag{7}$$

where j is an index of node in the x direction, n is an index of time level, Δt is the time step, and Δx is the spatial step. For simplicity, in Equation (7) and in other algebraic equations, a second subscript for direction y was omitted. The nodal value of the average diffusion coefficient $\overline{K}_j$ can be estimated as follows [40].

$$\overline{K}_j = \frac{K_j + K_{j+1}}{2} \tag{8}$$

$$K_j = \frac{1}{n_x}\left(H_j - z_j\right)^{5/3}\left(\frac{\partial H}{\partial s}\right)^{-1/2} \tag{9a}$$

$$K_{j+1} = \frac{1}{n_y}\left(H_{j+1} - z_{j+1}\right)^{5/3}\left(\frac{\partial H}{\partial s}\right)^{-1/2} \tag{9b}$$

in which the term $\partial H/\partial s$ is approximated by the following formula.

$$\frac{\partial H}{\partial s} = \sqrt{\left(\frac{H_{j+1} - H_j}{\Delta x}\right)^2 + \left(\frac{H_{i+1} - H_i}{\Delta y}\right)^2} \tag{10}$$

where subscript i refers to the actually considered column of the computational grid (Figure 1). The value of the diffusion coefficient K must be limited near the front of a wave propagating over a dry bottom. For this reason, it is assumed that K is equal to zero [41]:

- when the water depth is negative or zero, $h \leq 0$,

- or when the water stage derivative takes a very small value, $\partial H/\partial s < \varepsilon$, where ε represents the assumed tolerance, usually ranging from 10^{-6} to 10^{-9}.

The rearrangement of Equation (7) yields the following.

$$H_j^{n+1} = \left(\frac{\overline{K}_{j-1}^n \cdot \Delta t}{\Delta x^2}\right) H_{j-1}^n + \left(1 - \frac{\left(\overline{K}_{j-1}^n + \overline{K}_j^n\right) \cdot \Delta t}{\Delta x^2}\right) H_j^n + \left(\frac{\overline{K}_j^n \cdot \Delta t}{\Delta x^2}\right) H_{j+1}^n \tag{11}$$

Introducing the diffusive number, defined as:

$$D = \frac{\overline{K} \cdot \Delta t}{\Delta x^2} \tag{12}$$

Equation (11) can be rewritten in the form below.

$$H_j^{n+1} = D_{j-1}^n H_{j-1}^n + \left(1 - \left(D_{j-1}^n + D_j^n\right)\right) H_j^n + D_j^n H_{j+1}^n \text{for } j = 2, 3, \ldots, M-1 \tag{13}$$

A stability and accuracy analysis performed for the linear variant of the explicit scheme (Equation (13) with D = const.) indicates that this scheme is second-order accurate, but it is only conditionally stable [42]. Using the von Neumann method, it can be proven that, in this case, the stability condition takes the following form.

$$D = \frac{\overline{K} \cdot \Delta t}{\Delta x^2} \leq 0.5 \tag{14}$$

which is also equivalent to the condition for the maximal value of the time step.

$$\Delta t \leq \frac{\Delta x^2}{2\overline{K}} \tag{15}$$

The application of the explicit scheme does not require the gathering of algebraic linear equations written for all nodes into a common system of equations that is solved at a given time level. In this case, after the imposition of the boundary conditions, it is sufficient that the calculations are carried out systematically through nodes from $j = 2$ to $j = M - 1$. The solution process regarding the y direction is analogical to the one described above for the x direction. This method provides a simple implementation in the source code, which does not require the allocation of a significant amount of memory, which is needed for solving large equation systems. However, due to constraints on the length of the time step presented above, such an explicit scheme is not effective. Some improvement of the computational efficiency can be reached by using the adaptive time step method (ATS) [35]. In the ATS method, the value of the time step is based on the stability conditions in the form of Equation (14). Thus, at the current time level, in each node of the computational grid, the diffusion coefficient is examined and the maximal allowable time step values for the following simulation step are computed. However, experiments show that the reduction in computation time using the ATS method is relatively low in comparison to the simulation with the constant time step. Therefore, it seems that remarkable progress can be achieved only when time integration is carried out using implicit schemes.

2.3. Solution of the 1D Equation Using Modified FEM with the Implicit Scheme

A more efficient algorithm can be obtained if the 1D nonlinear diffusive wave equation is solved using the modified Galerkin FEM [43]. A modification of the standard FEM deals with spatial integration. A detailed description of this method in the case of Equation (4) is given by Gąsiorowski [27]. The approximation of the 1D diffusive wave equation (Equation (4) or Equation (5)) using the modified FEM provides the following system of ordinary differential equations (ODE) over time.

- for node $j = 1$

$$\omega\frac{\Delta x}{2}\frac{dH_1}{dt} + (1-\omega)\frac{\Delta x}{2}\frac{dH_2}{dt} - \frac{\overline{K}_1}{\Delta x}(-H_1 + H_2) + K\frac{dH}{dx}\Big|_1 = 0 \tag{16}$$

- for each internal node $j = 2, 3, \ldots, M-1$

$$\begin{aligned}(1-\omega)\tfrac{\Delta x}{2}\tfrac{dH_{j-1}}{dt} + \omega\cdot\Delta x\tfrac{dH_j}{dt} + (1-\omega)\tfrac{\Delta x}{2}\tfrac{dH_{j+1}}{dt} + \\ + \tfrac{\overline{K}_{j-1}}{\Delta x}(-H_{j-1} + H_j) - \tfrac{\overline{K}_j}{\Delta x}(-H_j + H_{j+1}) = 0\end{aligned} \tag{17}$$

- for node $j = M$

$$(1-\omega)\frac{\Delta x}{2}\frac{dH_{M-1}}{dt} + \omega\frac{\Delta x}{2}\frac{dH_M}{dt} + \frac{\overline{K}_{M-1}}{\Delta x}(-H_{M-1} + H_M) - K\frac{dH}{dx}\Big|_M = 0 \tag{18}$$

where ω is the weighting parameter ranging from 0 to 1, and $\overline{K}_j$ is the nodal value of the averaged diffusion coefficient calculated by Equation (8). Equation (17) written for each node gives a system of ODE, which can be expressed using the matrix notation.

$$\mathbf{A}\frac{d\mathbf{H}}{dt} + \mathbf{B}\cdot\mathbf{H} + \mathbf{G} = 0, \tag{19}$$

where $\mathbf{H} = [H_1, \ldots, H_{j-1}, H_j, H_{j+1}, \ldots, H_M]^T$ is the column vector of water stage nodal values, **A** is a tridiagonal constant matrix of size $M \times M$, **B** is a tridiagonal matrix of size $M \times M$ with elements dependent on the solution vector, and **G** is the column vector representing the fluxes through the ends of the elements.

In vector **G**, the fluxes have a zero value for all internal nodes of elements. These fluxes can exist at the upstream (node $j = 1$) and downstream (node $j = M$) ends if the Neumann condition is imposed. For the Dirichlet condition (when the function *H(t)* is imposed) or for the impervious boundaries (when the flux through the boundary is null $\partial H/\partial x = 0$), all other components of vector **G** have a zero value. In the considered problem the Neumann condition at the impervious boundary or the Dirichlet condition will be imposed, which means, in further analysis, vector **G** is omitted.

Due to the spatial discretization, the partial differential equation (Equation (4) or Equation (5)) was reduced to the system of ODE with respect to time (Equation (19)). For the solution of this system, considered to be the initial value problem, the following two-level method is used [44].

$$\mathbf{H}_{t+\Delta t} = \mathbf{H}_t + \Delta t\left((1-\theta)\frac{d\mathbf{H}}{dt}\Big|_t + \theta\frac{d\mathbf{H}}{dt}\Big|_{t+\Delta t}\right) \tag{20}$$

where $\mathbf{H}_{t+\Delta t}$ and $\mathbf{H}_t$ are the water stage values at the time step $t + \Delta t$ and t, respectively, Δt is a time step, θ is the weighting parameter belonging to the interval [1,0].

The implementation of Equation (20) for the system of ODE (19) yields a system of algebraic non-linear equations.

$$(\mathbf{A} + \Delta t\cdot\theta\cdot\mathbf{B}_{t+\Delta t})\mathbf{H}_{t+\Delta t} = (\mathbf{A} - \Delta t(1-\theta)\mathbf{B}_t)\mathbf{H}_t. \tag{21}$$

Introducing the diffusion number (Equation (12)), approximation equations in the system (21) for the interval node $j = 2, 3, \ldots, M-1$ can be written as follows.

$$\left[\frac{(1-\omega)}{2} - \theta\cdot D_{j-1}^{n+1}\right]H_{j-1}^{n+1} + \left[\omega + \theta\left(D_{j-1}^{n+1} + D_j^{n+1}\right)\right]H_j^{n+1} + \left[\frac{(1-\omega)}{2} - \theta\cdot D_j^{n+1}\right]H_{j+1}^{n+1} = f_j^n \tag{22}$$

where:

$$f_j^n = \left[\frac{(1-\omega)}{2} + (1-\theta)D_{j-1}^n\right]H_{j-1}^n + \left[\omega - (1-\theta)\left(D_{j-1}^n + D_j^{n+1}\right)\right]H_j^n + \left[\frac{(1-\omega)}{2} + (1-\theta)D_j^n\right]H_{j+1}^n.$$

In Equation (22), two weighting parameters ω and θ are involved. These parameters determine the numerical properties of the presented scheme and allow the stability and the accuracy of the solution to be controlled. It is worth noting that the modified FEM with a two-level scheme (Equation (22)) can be considered as a general approximation formula, which, depending on the values of the weighting parameters ω and θ, leads to the appropriate scheme of FEM or FDM. For example, Equation (22) with the weighting parameter $\omega = 1$ corresponds to the family of FDM formulas. Additionally, when the parameter θ is equal to zero, then Equation (22) is consistent with the explicit scheme of the FDM described by Equation (13).

The stability analysis performed by the Neumann method and the accuracy analysis carried out using the modified equation approach [42] showed that $\omega > 1/2$ reduces non-physical oscillations related with numerical dispersity [40,44]. The parameter θ determines the accuracy and the stability of time integration, and particular values of this parameter lead to some well-known methods of solving ODE. For $\theta = 1/2$, Equation (22) becomes the implicit trapezoidal scheme, which ensures an approximation of the 2nd order, while, for the value $\theta > 1/2$, the approximation of the time derivative is of the first order. In such a case, the accuracy of the numerical solution depends on the time step Δt, and the applied scheme generates numerical diffusion, which causes an excessive smoothing of the solution. Moreover, the values of parameter $\theta \geq 1/2$ cause the proposed scheme to be unconditionally stable with respect to time integration. This feature is an improvement over conditionally stable schemes and allows a bigger time step to be used than in the case of the explicit Euler scheme when $\theta = 0$.

2.4. Solution of the System of Algebraic Equations

2.4.1. Solution of the Nonlinear System

Since the elements of matrix **B** depend on the solution, the considered problem is nonlinear. For this reason, the system of algebraic Equations (21) has to be solved using iterative methods. In hydraulic problems, the Picard or the Newton method is usually used. The Picard method is one of the simplest iteration techniques, which does not converge as fast as the Newton method. However, in contrast to the Newton method, it is usually convergent for any starting initial values. The modified version of the Picard method [44], which converges faster than the basic one, can be applied to solve the system of nonlinear Equations (21). Then it gives the following sequence of iterations.

$$\left(\mathbf{A} + \Delta t \cdot \theta \cdot \mathbf{B}_{t+\Delta t}^{(k)}\right)\Delta \mathbf{H}_{t+\Delta t}^{(k+1)} = -\mathbf{F}^{(k)} \tag{23}$$

where k is an iteration index, $\Delta \mathbf{H}_{t+\Delta t}^{(k+1)} = \mathbf{H}_{t+\Delta t}^{(k+1)} - \mathbf{H}_{t+\Delta t}^{(k)}$ is the correction vector, and $\mathbf{F}^{(k)} = \left(\mathbf{A} + \Delta t \cdot \theta \cdot \mathbf{B}_{t+\Delta t}^{(k)}\right)\mathbf{H}_{t+\Delta t}^{(k)} - (\mathbf{A} - \Delta t(1-\theta)\mathbf{B}_t)\mathbf{H}_t$ is the vector in the kth iteration. The iteration process is continued until the following criterion of convergence is satisfied.

$$\left|H_j^{(k+1)} - H_j^{(k)}\right| = \left|\Delta H_j^{(k+1)}\right| \leq \delta \quad \text{for} \quad j = 1, 2, \ldots, M \tag{24}$$

where δ is a specified tolerance, usually $\delta = 0.001$ m is assumed.

As can be seen, Equation (23) differs from the Newton method in terms of the matrix of coefficients including the original matrices **A** and **B** of the system. Thus, such a form of the Picard method does not require the determination of the Jacobian matrix, which, in this case, can be computationally expensive.

2.4.2. Solution of the Linear System

During the numerical solution of the system of nonlinear algebraic equations by the modified Picard method (Equation (23)), in each iteration, the system of linear algebraic equations has to be solved. This system of linear equations can be written in the following matrix form.

$$\mathbf{S}\cdot\Delta\mathbf{H} = \mathbf{f} \tag{25}$$

where $\mathbf{S}$ is the matrix of size $N \times N$ or $M \times M$ (for the modified Picard method $\mathbf{S} = \mathbf{A} + \Delta t\cdot\theta\cdot\mathbf{B}^{(k)}_{t+\Delta t}$), $\Delta\mathbf{H}$ is the vector of unknowns ($\Delta\mathbf{H}^{(k+1)}_{t+\Delta t} = \mathbf{H}^{(k+1)}_{t+\Delta t} - \mathbf{H}^{(k)}_{t+\Delta t}$), and $\mathbf{f}$ is the right-hand side vector ($\mathbf{f} = -\mathbf{F}^{(k)}$) The resulting coefficient matrix $\mathbf{S}$ is tridiagonal with non-zero elements located along the main diagonal. The structure of this matrix is as follows.

$$S = \begin{bmatrix} \beta_1 & \gamma_1 & & & & & & \\ \alpha_2 & \beta_2 & \gamma_2 & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \alpha_{j-1} & \beta_{j-1} & \gamma_{j-1} & & & \\ & & & \alpha_j & \beta_j & \gamma_j & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \alpha_{M-1} & \beta_{M-1} & \gamma_{M-1} \\ & & & & & & \alpha_M & \beta_M \end{bmatrix} \tag{26}$$

Including the properties and structure of matrix $\mathbf{S}$ appearing in the system (25) allows the efficiency of the solution algorithm to be improved. Thus, it reduces the required computer memory and the computational time. Such a system with a tridiagonal matrix can be successfully solved using the Thomas algorithm [42]. In this case, only non-zero elements within the bands are used during computation. Consequently, the total number of arithmetic operations is reduced to $3(M-1)$ additions and multiplications and $2M-1$ divisions, where M is the number of unknowns [45].

When the modified Picard method is applied for the solution of the nonlinear system, the non-zero elements in matrix $\mathbf{S}$ take the following form.

- for row $j = 1$:

$$\beta_1 = \frac{\omega}{2} + \theta\cdot D^{(k)}_1 \tag{27a}$$

$$\gamma_1 = \frac{1-\omega}{2} - \theta\cdot D^{(k)}_1 \tag{27b}$$

- for rows $j = 2, 3, \ldots, M-1$:

$$\alpha_j = \frac{1-\omega}{2} - \theta\cdot D^{(k)}_{j-1} \tag{28a}$$

$$\beta_j = \omega + \theta\cdot\left(D^{(k)}_{j-1} + D^{(k)}_j\right) \tag{28b}$$

$$\gamma_j = \frac{1-\omega}{2} - \theta\cdot D^{(k)}_j \tag{28c}$$

- for row $j = M$:

$$\alpha_M = \frac{1-\omega}{2} - \theta\cdot D^{(k)}_M \tag{29a}$$

$$\beta_M = \frac{\omega}{2} + \theta\cdot D^{(k)}_M \tag{29b}$$

It can be noticed that, except for the first and last rows in matrix $\mathbf{S}$ (Equation (27a,b) and Equation (29a,b)), the element α_j in the current row j (Equation (28a)) is the same as the element γ_{j-1} in the

preceding row (Equation (28c)) for $j-1$. This allows the numerical algorithm to be improved with respect to memory capacity and performance. Taking this into consideration, the resulting matrix **S** will have the following structure.

$$S = \begin{bmatrix} \beta_1 & \gamma_1 & & & & & & & \\ \alpha_2 & \beta_2 & \gamma_2 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & \gamma_{j-2} & \beta_{j-1} & \gamma_{j-1} & & & & \\ & & & \gamma_{j-1} & \beta_j & \gamma_j & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \alpha_{M-1} & \beta_{M-1} & \gamma_{M-1} & \\ & & & & & & \alpha_M & \beta_M \end{bmatrix} \tag{30}$$

The utilization of the resulting matrix structure allows memory to be saved since it is required to store $2(M-3)+7$ elements instead of $3M$. In addition, the Thomas algorithm used for solving tridiagonal systems can be improved in a very simple manner. This will be mainly due to the decreased number of RAM hits. For this reason, the whole matrix **S** is stored in a single one-dimensional array. The repeating values γ are included only once in the vector. This storage format has the following form.

$$\mathbf{S} = \begin{bmatrix} \beta_1 & \gamma_1 & \alpha_2 & \beta_2 & \gamma_2 & \cdots & \gamma_{j-2} & \beta_{j-1} & \gamma_{j-1} & \beta_j & \gamma_j & \cdots & \alpha_{M-1} & \beta_{M-1} & \gamma_{M-1} & \alpha_M & \beta_M \end{bmatrix} \tag{31}$$

2.5. Parallelization and Solver Implementation

Due to the applied domain splitting, computations for different rows or columns of the computational grid can be executed independently. Thus, such a numerical algorithm is easily scalable and suitable for parallel computations. For code parallelization, the Open Multi-Processing (OpenMP) framework was used. The OpenMP is a multi-platform extension that provides a set of pre-processor directives and functions, which allow parallel computations to be easily implemented in a fork-join model on a multi-core personal computer with a shared memory. The utilization of the OpenMP requires the suitable specification for the parts of code for parallel execution. This is achieved with the appropriate OpenMP directives. The specified part of the code is executed in parallel. A master thread is forked to a number of slave threads, which are run concurrently, and when the execution of all threads is finished, then the results are merged (Figure 2b).

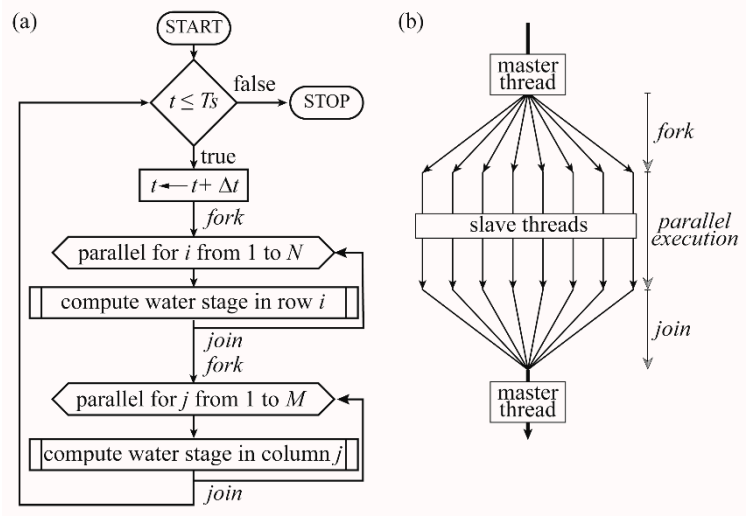

Figure 2. Graph of the main iteration loop in the parallelization process (**a**) and the fork-join model in the OpenMP framework (**b**).

The implemented solver consists of the main time loop, which includes two inner loops with one iterating through columns (integration in the y direction) and one iterating through rows (integration in the x direction) of the grid (Figure 2a). The work in these loops is forked to the slave threads and executed in parallel. The slave threads are assigned to the different CPU cores by the OpenMP runtime environment. This situation is schematically depicted in Figure 3.

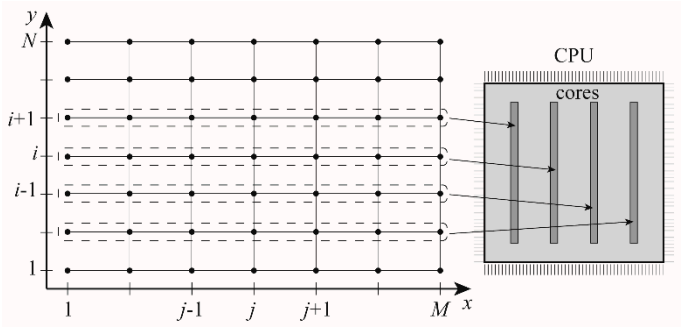

Figure 3. Schematic of the parallelization strategy resulting from the domain decomposition.

Due to the directional decomposition, the computations in slave threads are performed independently and there is no need to communicate between them. In order to eliminate the unnecessary usage of shared variables, private variables are introduced, which are assigned to the slave threads. Each OpenMP thread has its own memory pool in which its private variables are allocated (Figure 4).

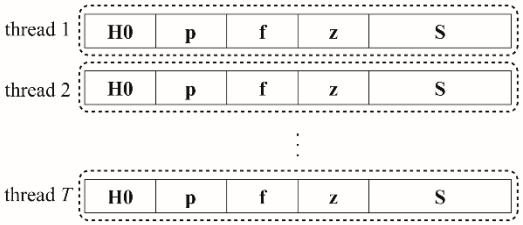

Figure 4. Memory management where each thread has its own memory pool with private variables.

Communication between the master and the slave threads is done with the use of shared variables. The shared variables are the bottom elevation **Z** and water stage values **H** for the whole solution domain. However, extensive use of the shared variables by the slave threads would introduce a significant overhead due to the handling of concurrent access to the memory resources. To improve the efficiency of the algorithm, the communication between master and slave threads during parallel execution should be minimal. Shared variables are accessed by the slave threads only when the necessity of reading or writing the data occurs. Such a situation takes place when the master thread is forked and when the slave threads are joined. There is no communication between slave threads during the computations of the water stage values. In the first operation after the fork, the water stage values and bottom elevations corresponding to a considered row or column are copied into the private variables of the slave threads. The water stage values are known from the initial condition or the previous step of computation. When the slave threads finish their tasks, then the result of the computations is written in the shared variable **H**.

The data from the shared variables **H** and **Z** is copied to the private variables denoted as **H0** and **z**, respectively. The private variable **z** stores the bottom elevation of the considered row or column, whereas **H0** stores the water stage values. Apart from the water stage values and bottom elevation in a row or column, the variables required to form and solve the system of arising nonlinear algebraic equations are used. These private variables are used for storing the system matrix **S** (Equation (25))

and the right-hand side vector **f**. To improve the solver efficiency, an auxiliary variable **p** for storing the vector on the right-hand side of Equation (21) is used. This expression is also a part of vector **f** (Equation (23)) and is constant at a given time level t. Therefore, it has to be computed only once in a considered time step. The computed water stage at time level $t + \Delta t$ is saved to the variable **H0**.

Apart from the utilized solution algorithm, in order to achieve the best possible performance of the solver, the proper format of variables has to be used. As linear data structures allow better efficiency to be obtained than two-dimensional data structures, all variables are stored in the form of linear arrays. Bottom elevation data **Z** and water stage values **H** for the whole solution domain are stored in the column-major format in linear arrays of size $N \times M$. The private variables storing the rows/columns of water stage values or bottom elevations are arrays of size max (M,N), respectively. A similar situation occurs in the case of the right-hand side vector **f**. The tridiagonal system matrix **S** is stored in the row-major format specified in Equation (31).

In modern computer architectures, one of the significant bottlenecks are RAM calls. Since the operating system can store variables in different locations of the RAM, then the access to multiple resources can be time-consuming. Therefore, in the implementation of the solver, a separate memory pool is assigned to each of the threads (Figure 4). To increase the memory search performance, private variables are allocated as adjacent memory blocks in the memory pools assigned to the threads.

Computations in the horizontal direction are performed first and then computations in the vertical direction are performed. Therefore, there is no need to allocate memory for each direction separately, only for the one that has the greater number of nodes. The same memory block is used for the computations in the other direction.

The computation process performed in each thread and the utilization of the variables is listed below.

(1) Bottom height values for the considered row or column are copied from the shared bottom variable **Z** and stored in the private variable **z**,
(2) Water stage values from the previous computation step for the considered row or column are copied from the shared water stage variable **H** and written in the private variable **H0**,
(3) Private variable **S** is used to create and then temporarily store the matrix $\mathbf{A} - \Delta t \cdot (1 - \theta)\mathbf{B}_t$,
(4) Product $(\mathbf{A} - \Delta t \cdot (1 - \theta)\mathbf{B}_t) \cdot \mathbf{H}_t$ is stored in the private variable **p**,
(5) Private variable **S** is used to create and store matrix $\mathbf{A} + \Delta t \cdot \theta \cdot \mathbf{B}_{t+\Delta t}^{(k)}$,
(6) Vector $\mathbf{f}^{(k)} = \left(\mathbf{A} + \Delta t \cdot \theta \cdot \mathbf{B}_{t+\Delta t}^{(k)}\right) \cdot \mathbf{H}_{t+\Delta t}^{(k)} - \mathbf{p}$ is created and stored in the private variable **f**,
(7) System (23) is solved and the solution result $\mathbf{H}_{t+\Delta t}^{(k+1)}$ is written in the variable **f**,
(8) Water stage vector $\mathbf{H}_{t+\Delta t}^{(k+1)} = \mathbf{H}_t + \Delta \mathbf{H}_{t+\Delta t}^{(k+1)}$ is updated and stored in **H0**,
(9) If the required solution accuracy is obtained, go to step 10, if not, return to step 5,

Obtained water stage values for the considered row or column are copied into the global variable **H** storing the water stage values for the whole domain.

In the above list, steps 5 to 9 refer to the solution of the arising system of nonlinear equations using the modified Picard method.

2.6. Measure of Efficiency

The computational efficiency is considered as the number of operations carried out at the execution time in order to obtain a numerical solution with assumed accuracy. Thus, the efficiency can be increased by reducing the computational time, i.e., using the longer time step Δt for assumed accuracy or by choosing a spatial grid of a suitable size for a given computational time. At the same time, it should be remembered that increasing the time step or spatial step decreases the accuracy of the obtained solution. For this reason, the computational efficiency, including the accuracy of the solution, is a more preferable and useful measure for the comparison of different numerical algorithms than the

solution accuracy or execution time considered separately. In the study, the computational efficiency was estimated by the following formula [42].

$$E = \frac{c}{RMSE \cdot T_S} \tag{32}$$

where E is the efficiency, c is a constant (it is assumed $c = 1$), T_S is the computational time, and $RMSE$ is an accuracy of the solution estimated using the root mean square error.

$$RMSE = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\left(H_{ref}(x_j) - H_{test}(x_j)\right)^2} \tag{33}$$

where H_{ref} is the reference solution, H_{test} denotes the tested solution, and x_j is the coordinate of the computational grid node in the tested solution, where $j = 1, \ldots, M$.

The results of the computations were also compared with regard to the relative efficiency.

$$\Delta E = \frac{(E_{EA} - E_E)}{E_E} \cdot 100\% \tag{34a}$$

or

$$\Delta E = \frac{(E_I - E_{EA})}{E_{EA}} \cdot 100\% \tag{34b}$$

In which ΔE is the relative efficiency, and E_E, E_{EA}, and E_I denote the efficiency (calculated using Equation (32)) of the considered scheme: pure explicit, explicit with a variable adaptive time step (ATS), and implicit, respectively.

In order to gain information about the behavior of the solver when it is executed in parallel, an additional measure, in the form of the speed up and the efficiency of parallel computations, was also used. The speed up parameter is defined as the ratio of the sequential computation time to the parallel computation time in order to execute the same algorithm [46].

$$S = \frac{T_S}{T_p} \tag{35}$$

whereas parallel efficiency is defined as the ratio of the speed up to the number of processors.

$$E_P = \frac{S}{N_p} \tag{36}$$

where S is the speed up, T_S is the computation time with one processor, T_P is the computation time with N_P processors, and N_P is the number of processors.

3. Results

In order to check the accuracy and computational efficiency of the considered implicit as well as the explicit scheme, 1D and 2D simulations of flood wave propagation were performed. The solver was implemented in C++ language (GCC 6.4.0 compiler suite) and compiled to a 64-bit executable.

3.1. One-Dimensional Flow, Horizontal Plane Wetting Test

In this test, a flat channel of the length $L = 1000$ m with a constant value of the Manning roughness coefficient $n = 0.04$ m$^{-1/3}$s is considered. This coefficient value corresponds to a natural channel with gravel in the bottom [47]. The initial water elevation at the time $t = 0$ s corresponds to the bottom

elevation $H_0(x, t = 0) = Z(x) = 0$ m. At the upstream end ($x = 0$), the inflow boundary condition is imposed in the form of water elevations varying in time.

$$H(x = 0, t) = \begin{cases} 0\,\text{m} & \text{for } t = 0.0\,s \\ 2t/60\,\text{m} & \text{for } 0\,s < t \leq 60\,s \\ 2\,\text{m} & \text{for } t > 60\,s \end{cases}$$

At the downstream end ($x = L$), as the boundary condition, a zero flux is imposed for the water elevation. During the calculations, the following values of numerical parameters were assumed.

(1) Tolerance for the iterative process of the solution of the system of nonlinear equations: $\delta = 0.001$ m,
(2) Threshold for the water stage derivative in Equation (9a) $\partial H/\partial s = \partial H/\partial x$: $\varepsilon = 10^{-9}$,
(3) Maximum number of iterations in a simulation step is 50.

In Figure 5, the examples of numerical solutions of the 1D diffusive wave Equation (4) with the space interval $\Delta x = 10$ m at the selected time $t = 450$ s are displayed. The computation by the explicit scheme was obtained using the adaptive time step method (ATS) with the minimal time step $\Delta t = 0.017$ s. Numerical solutions with the implicit scheme were performed for various time step values Δt = [2, 10, 20] s as well as for various weighting parameter values θ = [0.5, 0.6, 0.8, 1.0]. All computations regarding the implicit scheme were made with the weighting parameter equal to $\omega = 0.7$. Additionally, the results of the calculations were compared with the reference solution. Such a solution was obtained with the implicit scheme for a fine numerical mesh with the spatial element size $\Delta x = 0.01$ m and time step $\Delta t = 0.001$ s with $\theta = 0.5$, $\omega = 0.7$.

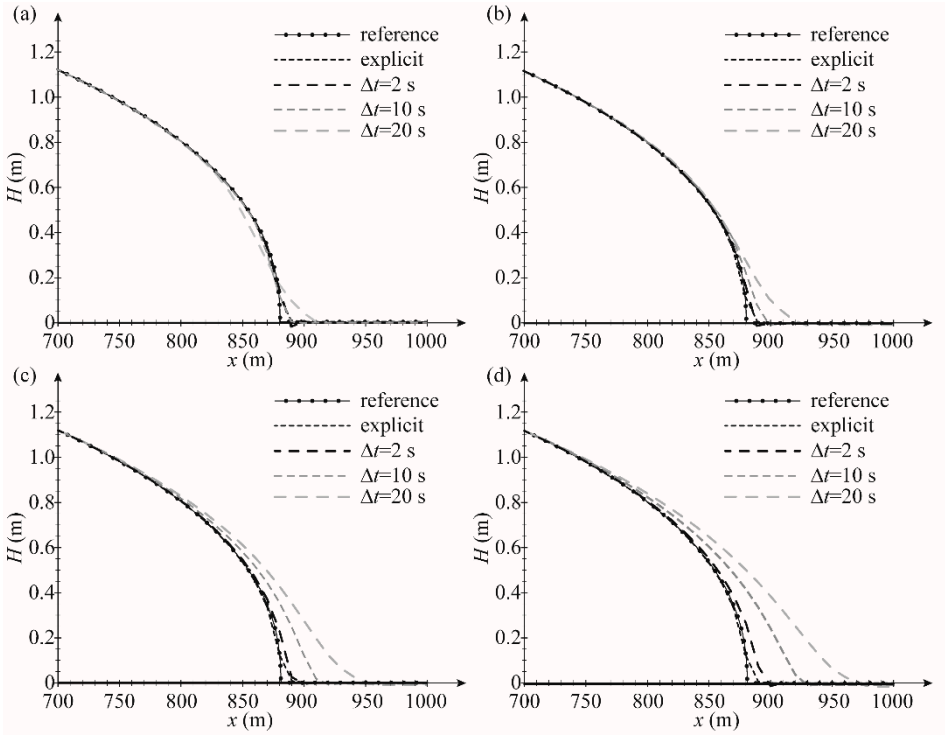

Figure 5. Numerical solution of Equation (4) at $t = 450$ s with $\Delta x = 10$ m using the explicit scheme ($\Delta t = 0.017$ s) and the implicit scheme for various time step values Δt = [2, 10, 20] s and for various values of the weighting parameter: $\theta = 0.5$ (**a**), $\theta = 0.6$ (**b**), $\theta = 0.8$ (**c**), $\theta = 1.0$ (**d**).

It can be seen that the solution obtained with the implicit scheme for $\theta = 0.5$ (Figure 5a) and with different time step values is only slightly different from the reference solution as well as from the solution obtained with the explicit scheme. However, as the values of the weighting parameter

and the time step increase ($\theta = 0.6 \div 1.0$ and $\Delta t = 2 \div 20$ s), the obtained solution is systematically overestimated with regard to the reference solution (Figure 5b–d). These results confirm that the applied implicit algorithm with $\theta = 0.5$ becomes an approximation of the second order with respect to the time, whereas, for other values, the scheme generates the numerical diffusion with a maximal intensity for $\theta = 1$ (Figure 5d).

Table 1, this table presents the influence of the time step and spatial step variability on the accuracy of solutions. In this test, the spatial step Δx was equal to 2, 5, and 10 m, which correspond to 500, 200, and 100 finite elements, respectively. A comparison of solutions (at the time $t = 450$ s) was provided for the explicit scheme with the ATS adjustment and for the implicit scheme with different values of the θ parameter. The accuracy of the solution for each scheme was estimated using RMSE (Equation (33)) as a criterion.

Table 1. Impact of the space interval Δx, time interval Δt, and spatial approximation weighting parameter θ on the accuracy of the numerical solution obtained using the implicit and explicit schemes with ATS.

Spatial Step Δx	Explicit		Implicit				
	Time Step min/max Δt	RMSE	Time Step Δt	RMSE			
				$\theta = 0.5$	$\theta = 0.6$	$\theta = 0.8$	$\theta = 1.0$
(m)	(s)	(10^{-3} m)	(s)	(10^{-3} m)	(10^{-3} m)	(10^{-3} m)	(10^{-3} m)
2	0.00067/2	2.09	2	1.43	5.45	12.66	18.64
5	0.00420/5	2.64	5	1.58	10.61	24.89	36.26
10	0.01690/10	4.73	10	3.55	18.70	41.55	59.77

This experiment showed that the accuracy of the solution obtained with the explicit scheme is very similar to the corresponding one generated by the implicit scheme, but with an incomparably greater time step. It takes place when the numerical diffusion is minimized, i.e., for a low value of the weighting parameter θ. For example, the RMSE obtained for the spatial step $\Delta x = 5$ m with the explicit scheme has the same order of magnitude ($RMSE = 2.64 \times 10^{-3}$ m) as the solution obtained with the implicit method for identical input data with the parameter θ ranging from 0.5 to 0.6 ($RMSE = 1.58 \times 10^{-3}$ m for $\theta = 0.5$ and $RMSE = 10.61 \times 10^{-3}$ m for $\theta = 0.6$). In the case of other values of this parameter, $\theta = 0.8$ and $\theta = 1.0$, higher errors were received of $RMSE = 24.89 \times 10^{-3}$ m and $RMSE = 36.26 \times 10^{-3}$ m, respectively. For this reason, in all further examples, the weighting parameter θ equal to a value ranging from 0.5 to 0.65 will be used. Such values ensure a stable solution without oscillations, and, simultaneously, the generated numerical diffusion is minimized.

In the next tests, the computational efficiency is considered. At first, a comparison of the computational efficiency between an explicit scheme with a constant time step and a scheme with the adaptive time step (ATS) was performed. Computations were driven for different values of the spatial step Δx varying between 2 and 10 m. In order to obtain the allowable time step for computations with a constant time step, the minimal value was taken from the adaptive method computations. The results for a simulation period of 450 s are presented in Table 2. Additionally, in Figure 6, the total computation time (Figure 6a) and the efficiency (Figure 6b) with respect to the spatial step Δx are presented.

Table 2. Accuracy and efficiency statistics (the total computation time, the computational efficiency (Equation (32)) and the relative efficiency (Equation (34a)) for the explicit scheme with the constant time step and the adaptive time step (ATS).

Spatial Step Δx	Constant Time Step				Adaptive Time Step (ATS)				Relative Effic. ΔE
	Time Step Δt	RMSE	Comp. Time T_S	Comp. Effic. E_E	Time Step Min/Max Δt	RMSE	Comp. Time T_S	Comp. Effic. E_{EA}	
(m)	(s)	(10^{-3} m)	(s)	($m^{-1}s^{-1}$)	(s)	(10^{-3} m)	(s)	($m^{-1}s^{-1}$)	(%)
2	0.0006	2.05	67.89	7.2	0.0006/2	2.09	53.46	8.9	23.6
5	0.0042	1.92	4.35	119.9	0.0042/5	2.64	3.53	106.9	−10.8
10	0.0169	1.31	0.55	1386.7	0.0169/10	4.73	0.45	469.0	−66.2

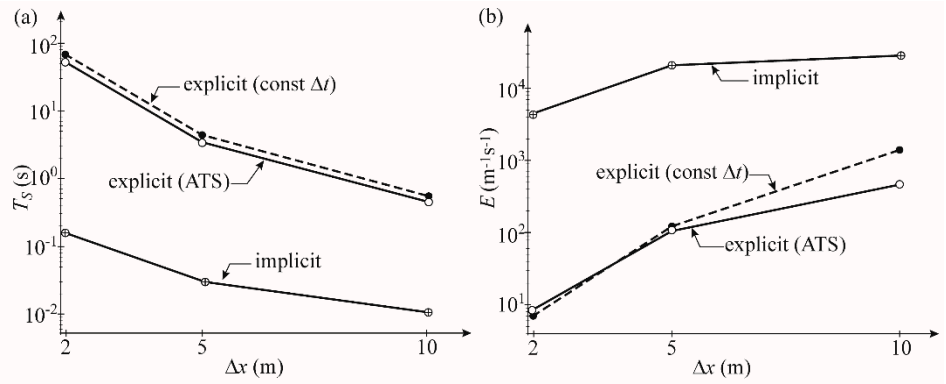

Figure 6. Comparison of the explicit, explicit with ATS, and implicit schemes ($\theta = 0.5$ and $\omega = 0.7$) with regard to computation time T_S (**a**) and computational efficiency E (**b**).

It is apparent (Table 2 and Figure 6a) that the profit from the time step adaptation with the ATS method is moderate with respect to the execution time. For example, in the case of the computational grid with $\Delta x = 5$ m, the required computational time T_S was equal to 4.35 s for the constant time step algorithm, whereas the ATS method required 3.82 s. This corresponds to a reduction in the computation time by about 20% in comparison to the simulation with the constant time step. However, if in the considered example we take into account the accuracy of the solution, then it can be seen that the relative efficiency of the ATS method ($E_{EA} = 106.9$ m^{-1} s^{-1}) is less than the efficiency of the pure explicit method ($E_{EA} = 119.9$ m^{-1}1 s^{-1}). Consequently, the relative computational efficiency ΔE increases to a value 10.8% in favor of the explicit method with the constant time step. This tendency is even more visible in Figure 6b, where it is clearly displayed that the ATS method becomes less effective with regard to the spatial step $\Delta x \geq 5$ m.

In Figure 7, the variability of the time step length with the simulation time for the solution of the ATS method is displayed. It can be seen that the time step length rapidly becomes very small after the initial steps of the simulation and converges to a value of the time step corresponding to a pure explicit scheme. Consequently, the time step can be maintained at a reasonable length only during part of the wave propagation period, whereas, for the rest of the simulation, it must be reduced to very small values.

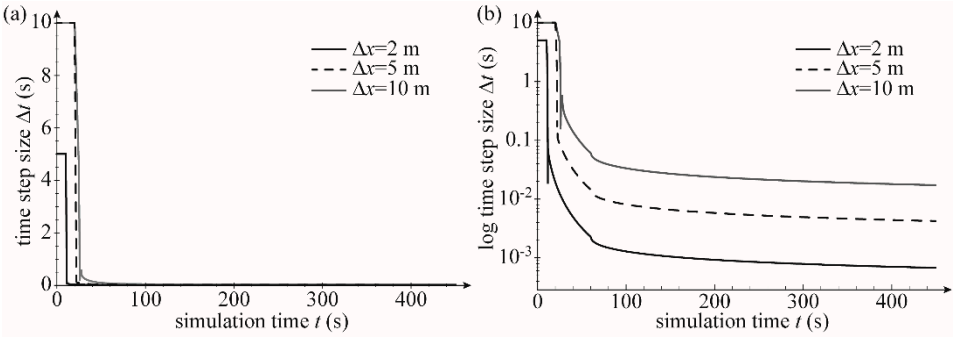

Figure 7. Dependence of the time step length on the simulation time for the ATS method: linear scale (**a**), logarithmic scale (**b**).

Table 3, this table presents a detailed comparison between the explicit ATS method and the implicit scheme with regard to the computational efficiency. As shown, the computational time required to finish the simulation using the implicit method is incomparably smaller, by about two orders of magnitude, than the time required by the explicit method. For example, in the case of computations with $\Delta x = 5$ m and $\Delta t = 5$ s imposed as the initial time step for the explicit method, the computation time was equal to 3.53 s. The computations for an analogous case using the implicit method require a significantly shorter time 0.03 s to finish the simulation.

Table 3. Accuracy and efficiency statistics (the total computation time, the computational efficiency, and (Equation (32)) the relative efficiency (Equation (34b)) for the explicit scheme with the constant time step, the adaptive time step (ATS), and the implicit method with $\theta = 0.5$ and $\omega = 0.7$.

Spatial Step x	Explicit Adaptive Time Step (ATS)				Implicit				Relative Effic. ΔE
	Time Step min/max t	RMSE	Comp. Time T_S	Effic. E_{EA}	Time Step Δt	RMSE	Comp. Time T_S	Effic. E_I	
(m)	(s)	(10^{-3} m)	(s)	(m^{-1} s^{-1})	(s)	(10^{-3} m)	(s)	(m^{-1} s^{-1})	(%)
2	0.0006/2	2.09	53.46	8.9	2	1.42	0.16	4386	49,105
5	0.0042/5	2.64	3.53	106.9	5	1.58	0.03	21,059	19,590
10	0.0169/10	4.73	0.45	469.0	10	3.55	0.01	28,158	5303

The obtained results also indicate that, for a similar order of solution accuracy ($RMSE = 1.58 \times 10^3$ m for the implicit and $RMSE = 2.64 \times 10^{-3}$ m for the explicit scheme), the implicit scheme is definitely more effective than the explicit scheme. In the considered example (for $\Delta x = 5$ m), the implicit scheme (with efficiency $E_I = 21{,}059$ m^{-1} s^{-1}) outperforms the explicit scheme (with efficiency $E_I = 106.9$ m^{-1} s^{-1}), and the relative efficiency ΔE is equal to 19,590%. Considerable differences between the schemes become more obvious in Figure 6, which displays the variability of total computation time and the computational efficiency with regard to the spatial step. It can be shown that the amount of time T_S required to perform the simulation differs significantly for implicit and explicit methods (Figure 6a), even considering the explicit scheme with the variable time step (ATS). If we consider the computational efficiency (Figure 6b), it is clearly visible that, in this case, the effectiveness of the implicit scheme also increases significantly with the spatial step increasing to the value $\Delta x = 5$ m. After reaching this value, the efficiency is at a high level in comparison to the explicit scheme, but the increase is not so intense.

3.2. Two-Dimensional Flow, Parallel Computation for the Wetting-Drying Test

In the final test, simulations of a 2D flow over a floodplain using parallel computations were examined. For this purpose, an area of the dimensions $L_x \times L_y = (1000 \times 1000)$ m^2 was considered. The bottom elevation in the considered area is given with the following formula.

$$Z(x,y) = \sin\left(10\frac{x-L_x}{L_x}\right)\cdot\cos\left(10\frac{y-L_y}{L_y}\right)$$

The bottom takes a shape with local hillocks. The bottom elevations, in the form of a contour-density plot, are given in Figure 8a. The Manning roughness coefficient is assumed to be constant and equal to $n = 0.05$ m$^{-1/3}$ s over the whole floodplain. At the beginning of the simulation, the area is dry, so the initial water elevation at the time $t = 0$ s corresponds to the bottom elevation $H_0(x,y,t = 0) = Z(x,y)$. The zero-flux boundary condition with regard to the water stage is imposed at the boundary of the considered area. The exception is the section placed at $y = 0$ m and at x between 450 m and 550 m, where the Dirichlet boundary condition, in the form of the variable water level $H_b(t)$, is imposed (Figure 8b).

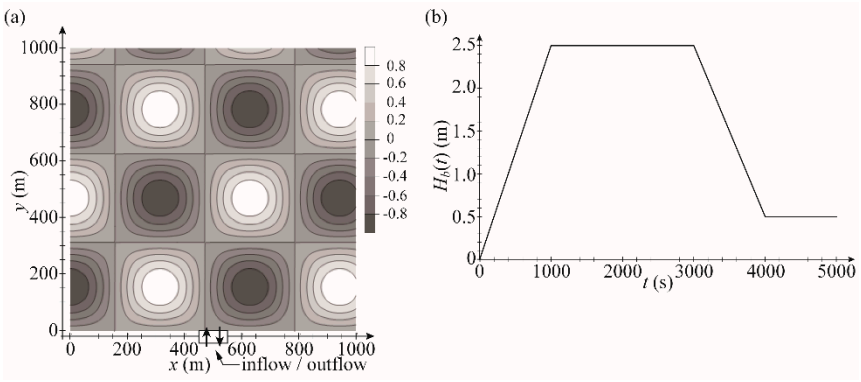

Figure 8. Contour density plot of the bottom elevation (**a**) and the Dirichlet type boundary condition (**b**) assumed in the two-dimensional wave propagation test.

The simulations were performed for four different grid resolutions with a number of nodes [100 × 100, 200 × 200, 500 × 500, 1000 × 1000], which correspond to the spatial length $\Delta x = \Delta y$ = [10, 5, 2, 1] m and the time step length Δt = [2.5, 1.25, 0.5, 0.25] s, respectively. For the implicit scheme, the assumed values of the weighting parameters were $\theta = 0.65$ and $\omega = 0.7$. Other values of parameters in the numerical computation, such as tolerance for the iterative process δ, threshold for the water stage gradient ε, and maximal number of iterations per time step are assumed as in the previous test.

In Figure 9 the computation results (for 200 × 200 elements), in the form of a 3D view of the water surface at selected times, are presented. It is apparent that, due to the assumed inflow boundary condition at the beginning of the inundation, the front of the water wave moves over a dry bottom (Figure 9a,b). After a time of about $t = 3900$ s, the area is filled up to the level of about $H = 1.85$ m (Figure 9c) and local obstacles in the form of hillocks are under the water. A decrease in the water level specified by the boundary condition causes the outflow of water from the floodplain (Figure 9d).

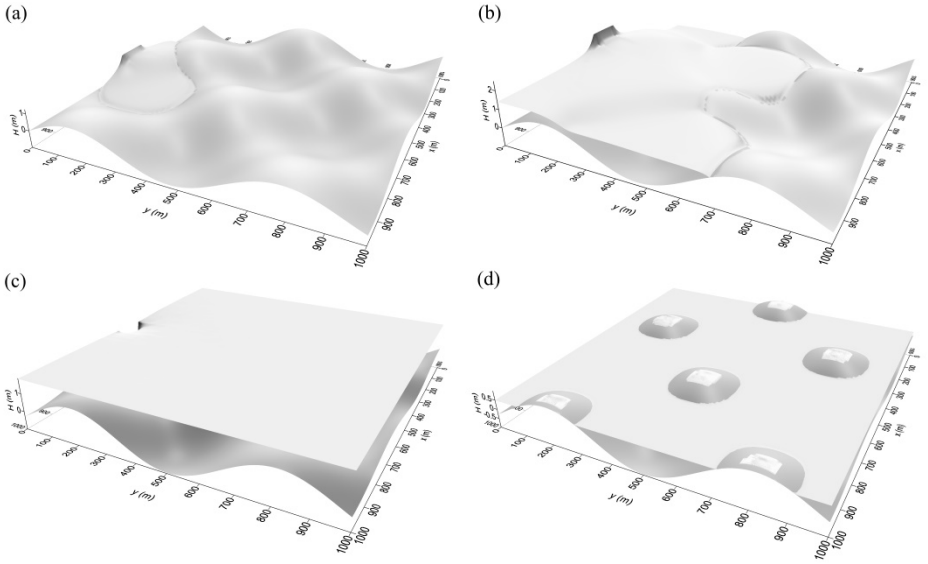

Figure 9. Three dimensional view of water surface propagation over the bottom (for 200 × 200 elements, Δx = 5 m, Δt = 5 s) at selected times: t = 500 s (**a**), t = 1200 s (**b**), t = 3900 s (**c**), t = 30,000 s (**d**).

As in the previous test, the computation results were compared to the reference solution obtained by the implicit scheme with the mesh element size Δx = 0.1 m (number of elements 10,000 × 10,000) and the time step size Δt = 0.01 s. Computations conducted by the explicit scheme with the spatial step $\Delta x = \Delta y$ = 1 m (number of nodes 1000 × 1000) and the time step Δt = 0.0001 s were also compared to the reference solution. Since the resolution grid for the reference solution is very fine and the total computation time is large for the considered 2D flow problem, the comparison between solutions with regard to the accuracy is performed at the selected time of 1200 s, whereas the computation times T_S are compared for the entire simulation period t = 30,000 s. Example solutions of water profiles at given cross-sections corresponding to x = 500, computed for different grid resolutions, are given in Figure 10.

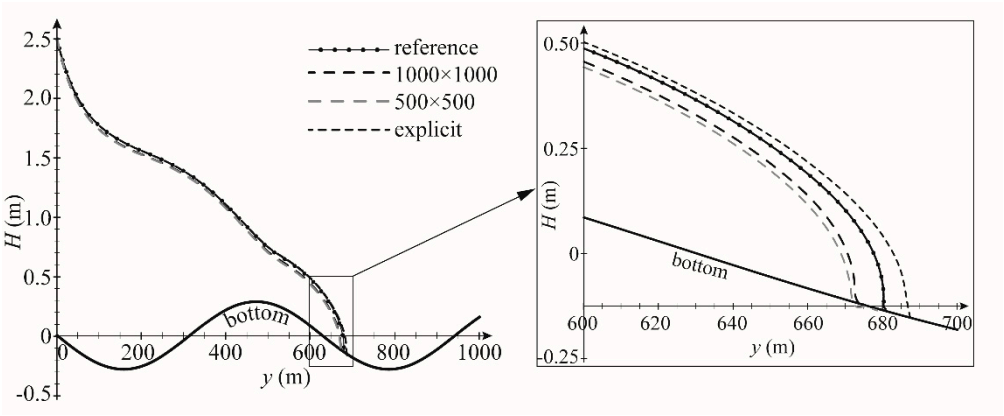

Figure 10. Water profiles along the row corresponding to x = 500 m computed at the time 1200 s using the explicit and implicit schemes for grid resolutions 1000 × 1000 and 500 × 500.

It can be seen that the results obtained with the explicit and implicit schemes are close to the reference solution. In the simulation driven using the explicit scheme, the wave propagates with the highest speed and, after the time 1200 s, it reaches a distance of about 694 m. After the same time in the simulation conducted by the implicit scheme, the wave front propagates at a lower speed while achieving a distance of about 674 m. Thus, the solution obtained with the explicit scheme is

overestimated, whereas the solution with the implicit scheme is underestimated in comparison with the reference solution where the wave front is located at 680 m.

The quantitative comparison of the results for the implicit scheme is given in Table 4. The errors, expressed in terms of *RMSE*, vary from 0.026 m (for 100 × 100 elements) to 0.036 m (for 200 × 200 elements), which can be considered as insignificantly small from a practical viewpoint. Similar results were also obtained for the explicit scheme, where the total *RMSE* achieved the value of 0.02 m.

Table 4. Accuracy, efficiency, and statistics (computational time, speed up (Equation (35)), and parallel efficiency (Equation (36)) for parallel computations using the implicit scheme (θ = 0.65 and ω = 0.7) for the simulation time equal to t = 30,000 s.

No. Elements	Spatial Step $\Delta x = \Delta y$	Time Step Δt	RMSE	No. CPU Proc.	Comp. Time T_S	Speed Up S	Parallel Effic. E_P
(-)	(m)	(s)	(m)	(-)	(s)	(-)	(-)
100 × 100	10	2.5	0.026	1	87.59	1.00	1.00
				2	50.17	1.75	0.87
				4	30.35	2.89	0.72
				8	23.04	3.80	0.48
200 × 200	5	1.25	0.036	1	627.99	1.00	1.00
				2	346.80	1.81	0.91
				4	202.73	3.09	0.77
				8	140.17	4.48	0.56
500 × 500	2	0.5	0.035	1	9047.14	1.00	1.00
				2	5245.99	1.72	0.86
				4	2959.87	3.06	0.76
				8	1996.98	4.53	0.57
1000 × 1000	1	0.25	0.027	1	73,267.80	1.00	1.00
				2	40,659.70	1.80	0.90
				4	24,075.80	3.04	0.76
				8	15,064.30	4.86	0.61

The calculations of the computation time, the speed up (Equation (35)) and the parallel efficiency (Equation (36)) with regard to the grid resolution and the number of used CPU processors are summarized in Table 4. The dependencies of the speed up as well as parallel efficiency on the number of processors are plotted in Figure 11.

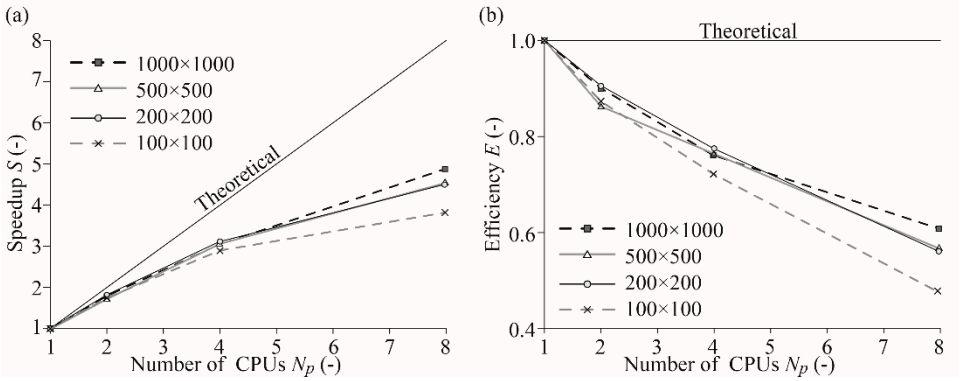

Figure 11. Dependency of (**a**) speed up and (**b**) efficiency on the number of processors applied in the CPU for parallel computations.

As shown, the parallelization strategy is very effective in comparison with the sequential computation. For all assumed grid resolutions, an enormous reduction in computational time is

observed. For example, the calculation performed using eight cores on a grid of 1000 × 1000 nodes took T_S = 15,064.3 s (4.1 h), whereas the calculation on a single processor was carried out within T_S = 73,267.8 s (20.3 h). The computational grid with 1000 × 1000 elements corresponds to over one million nodes. Such a refined grid, applied to the simulation of a flood wave at a real time equal to t = 30,000 (8.33 h), constitutes a demanding task with regard to computing power. The same simulation performed by the explicit scheme takes a very long time. In this case, it took 1,638,140 s (455 h = 18 days) with the minimum time step Δt = 0.001 s using eight threads. Thus, the application of the explicit scheme is very ineffective on a high-resolution grid with a large number of nodes.

As previously stated, the analysis of parallelization by means of the speed up factor and parallel efficiency allows the determination of the optimal number of processors to use for the considered simulation problem. With regard to the speed up (Figure 11a), we can see that this parameter increases as the number of processors increases. Thus, the proposed algorithm is scalable, i.e., it ensures the adequate acceleration growth of calculations as the number of processors increases. In the study, the obtained tendency in terms of the speed up (Figure 11a) has a nonlinear character, where the growth rate of the speed up gradually decreases with the number of processors. For example, a simulation with two processors using a grid of 1000 × 1000 gives a speed-up factor of 1.8, whereas the use of eight processors leads to a speed-up of 4.86 (-) for the same grid resolution. This decreasing trend is even more apparent with regard to the efficiency (Figure 11b), which yields the values 0.9 and 0.61 (-), respectively.

4. Discussion

The explicit scheme, due to the stability condition, requires the use of a small-time step, which allows obtaining relatively high accuracy with regard to time integration. In the case of an implicit scheme, the accuracy of the calculations depends on the value of the weighting parameter θ and the time step. As shown, for θ = 0.5, the accuracy of the solution is practically the same as for the explicit scheme and does not depend on the length of the time step. However, for such a value of parameter (θ = 0.5), a numerical solution can generate the non-physical oscillations, which usually lead to the breakdown of calculations. In practical hydrological problems, in order to eliminate these oscillations, the numerical diffusion is introduced by using the weighting parameter θ larger than 0.5. In such a case larger time step leads to results that will be slightly less accurate than when using an explicit scheme. It is worth noting that the proposed scheme also provides a high flexibility, because, by selecting the appropriate values of numerical parameters θ and ω various schemes can be obtained. Thus, due to this feature, the accuracy and stability of the numerical solution can be controlled. The explicit scheme does not provide such possibilities.

For a similar order of accuracy, the implicit algorithm is significantly more effective, about two orders of magnitude with regard to the computational time, than the explicit scheme. Moreover, the implicit scheme outperforms the explicit scheme with the relative efficiency ranging from E = 5300% to E = 49,100% for a 1D problem. Even the use of a variable adaptive time step (ATS) in the explicit scheme leads to a reduction of computation time of only about 20% in comparison with the simulation with the constant time step. It is interesting that the time step can be maintained with a reasonable length only during part of the wave propagation period, whereas, for the rest of the simulation, it must be reduced to very small values to fulfil the stability condition. Moreover, if the accuracy of the obtained solution is considered in the efficiency estimation, then the calculated efficiency of the ATS method can be lower than for the pure explicit scheme. Such a surprising situation is due to the lower accuracy of the solution obtained by means of the ATS method when the larger time step is used. The increasing time step involves a higher value of RMSE, which, in turn, leads to the decreased relative efficiency ΔE (according to Equation (34a)).

An approximation of the diffusive wave equation using the explicit schemes allows calculations to be performed that do not require iterative methods to solve a system of nonlinear algebraic equations, which is the case with an implicit scheme. For this reason, the explicit schemes are easy to implement

for parallel computations. However, the application of the explicit scheme is very ineffective for a simulation of a two-dimensional flow on a grid with a large number of nodes. Even the use of parallelization techniques with a maximal number of processors (eight processors in the test) does not bring acceptable results, because, when a long real-time simulation is required, then the execution time can reach unacceptable values from the practical point of view. For 1000 × 1000 (1 million) nodes, the computational time was 455 h. On the other hand, the parallelization of computation using the implicit scheme is very effective and leads to a significant reduction in computational time. In the considered test, this time was only 4.1 h.

The proposed parallel strategy is scalable and ensures the appropriate speed up growth of calculations as the number of processors increases. In theory, the parallel execution of the solver should lead to a linear growth in speed-up for the maintenance of a constant efficiency equal to 1 (see Figure 11b). However, in reality, such a relationship cannot be achieved. The management of the parallel execution of the program and the copying of data results in a limit on the speed-up. In the study, the speed-up has a nonlinear character and is similar to those obtained by Leandro et al. [34]. However, in their study, an explicit scheme was implemented, which significantly extended the computational time.

Taking into account the presented results, the proposed parallel algorithm based on the implicit scheme seems to be a very attractive strategy for the two-dimensional modeling of flood inundation, where calculations with a high resolution or a long-term simulation for a vast domain are required.

Author Contributions: Conceptualization, W.A. and D.G. Methodology, W.A. and D.G. Software, W.A. Validation, W.A. Formal analysis, D.G. Investigation, D.G. and W.A. Resources, D.G. Data curation, W.A. Writing—original draft preparation, D.G. and W.A. Writing—review and editing, D.G. and W.A. Visualization, W.A. and D.G. Supervision, D.G. and W.A. Project administration, D.G. and W.A.

Funding: This research received no external funding.

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

References

1. Heniche, M.; Secretan, Y.; Boudreau, P.; Leclerc, M. A two-dimensional finite drying-wetting shallow water model for rivers and estuaries. *Adv. Water Resour.* **2000**, *23*, 359–372. [CrossRef]
2. Horritt, M.S. Evaluating wetting and drying algorithms for finite element models of shallow water flow. *Int. J. Numer. Methods Eng.* **2002**, *55*, 835–851. [CrossRef]
3. Begnudelli, L.; Sanders, B.F. Unstructured grid finite-volume algorithm for shallow-water flow and scalar transport with wetting and drying. *J. Hydraul. Eng. ASCE* **2006**, *132*, 371–384. [CrossRef]
4. Liang, Q.; Borthwick, A.G.L. Adaptive quadtree simulation of shallow flow with wet-dry fronts over complex topography. *Comput. Fluids* **2009**, *38*, 221–234. [CrossRef]
5. Szydłowski, M.; Kolerski, T.; Zima, P. Impact of the Artificial Strait in the Vistula Spit on the Hydrodynamics of the Vistula Lagoon (Baltic Sea). *Water* **2019**, *11*, 990. [CrossRef]
6. Tan, W.Y. *Shallow Water Hydrodynamics*; Elsevier: Amsterdam, The Netherlands, 1992.
7. Song, L.; Zhou, J.; Guo, J.; Zou, Q.; Liu, Y. A robust well-balanced finite volume model for shallow water flows with wetting and drying over irregular terrain. *Adv. Water Resour.* **2011**, *34*, 1915–1932. [CrossRef]
8. Vacondio, R.; Palù, A.D.; Mignosa, P. GPU-enhanced Finite Volume Shallow Water solver for fast flood simulations. *Environ. Modell. Softw.* **2014**, *57*, 60–75. [CrossRef]
9. Oritz, P. Shallow water flows over flooding areas by a flux-corrected finite element method. *J. Hydraul. Res.* **2013**, *52*, 241–252. [CrossRef]
10. Gourgue, O.; Comblen, R.; Lambrechts, J.; Kärnä, T.; Legat, V.; Deleersnijder, E. A flux-limiting wetting–drying method for finite-element shallow-water models, with application to the Scheldt Estuary. *Adv. Water Resour.* **2009**, *32*, 1726–1739. [CrossRef]
11. Cimorelli, L.; Cozzolino, L.; Della Morte, R.; Pianese, D.; Singh, V.P. A new frequency domain analytical solution of a cascade of diffusive channels for flood routing. *Water Resour. Res.* **2015**, *51*, 2393–2411. [CrossRef]

12. Cimorelli, L.; Cozzolino, L.; D'Aniello, A.; Pianese, D. Exact solution of the Linear Parabolic Approximation for flow-depth based diffusive flow routing. *J. Hydrol.* **2018**, *563*, 620–632. [CrossRef]
13. Zhang, X.H.; Ishidaira, H.; Takeuchi, K.; Xu, Z.X.; Zhuang, X.W. An approach to inundation in large river basins using the triangle finite difference method. *J. Environ. Inform.* **2004**, *3*, 51–63. [CrossRef]
14. Prestininzi, P. Suitability of the diffusive model for dam break simulation application to a CADAM experiment. *J. Hydrol.* **2008**, *361*, 172–185. [CrossRef]
15. Moussa, R.; Bocquillon, C. On the use of the diffusive wave for modeling extreme flood events with overbank flow in the floodplain. *J. Hydrol.* **2009**, *374*, 116–135. [CrossRef]
16. Hromadka, T.V.; Yen, C.C. A diffusion hydrodynamic model (DHM). *Adv. Water Resour.* **1986**, *9*, 118–170. [CrossRef]
17. Han, K.Y.; Lee, J.T.; Park, J.H. Flood inundation analysis resulting from levee-break. *J. Hydraul. Res.* **1998**, *36*, 747–759. [CrossRef]
18. Lal, A.M. Weighted implicit finite-volume model for overland flow. *J. Hydraul. Eng. ASCE* **1998**, *124*, 941–950. [CrossRef]
19. Cea, L.; Garrido, M.; Puertas, J. Experimental validation of two-dimensional depth-averaged models for forecasting rainfall-runoff from precipitation data in urban areas. *J. Hydrol.* **2010**, *382*, 88–102. [CrossRef]
20. Di Giammarco, P.; Todini, E.; Lamberti, P. A conservative finite elements approach to overland flow: The control volume finite element formulation. *J. Hydrol.* **1996**, *175*, 267–291. [CrossRef]
21. Arico, C.; Sinagra, M.; Begnudelli, L.; Tucciarelli, T. MAST-2D diffusive model for flood prediction on domains with triangular Delaunay unstructured meshes. *Adv. Water Resour.* **2011**, *34*, 1427–1449. [CrossRef]
22. Szymkiewicz, R.; Gąsiorowski, D. Simulation of unsteady flow over floodplain using the diffusive wave equation and the modified finite element. *J. Hydrol.* **2012**, *464–465*, 165–167. [CrossRef]
23. Toro, E.F. *Riemann Solvers and Numerical Methods for Fluid Dynamics*; Springer: Berlin, Germany, 1997.
24. LeVeque, R.J. *Finite Volume Methods for Hyperbolic Problems*; Cambridge University Press: Cambridge, MA, USA, 2002. [CrossRef]
25. Neal, J.C.; Fewtrell, T.J.; Bates, P.D.; Wright, N.G. A comparison of three parallelisation methods for 2D flood inundation models. *Environ. Modell. Softw.* **2010**, *25*, 398–411. [CrossRef]
26. Yu, D. Parallelization of a two-dimensional flood inundation model based on domain decomposition. *Environ. Modell. Softw.* **2010**, *25*, 935–945. [CrossRef]
27. Gąsiorowski, D. Analysis of floodplain inundation using 2D nonlinear diffusive wave equation solved with splitting technique. *Acta Geophys.* **2013**, *61*, 668–689. [CrossRef]
28. Hsu, M.H.; Chen, S.H.; Chang, T.J. Inundation simulation for urban drainage basin with storm sewer system. *J. Hydrol.* **2000**, *234*, 21–37. [CrossRef]
29. Teng, J.; Jakeman, A.J.; Vaze, J.; Croke, B.F.W.; Dutta, D.; Kim, S. Flood inundation modelling: A review of methods. Recent advances and uncertainty analysis. *Environ. Modell. Softw.* **2017**, *90*, 210–216. [CrossRef]
30. Bates, P.D.; De Roo, A.P.J. A simple raster-based model for flood inundation simulation. *J. Hydrol.* **2000**, *236*, 54–77. [CrossRef]
31. Yu, D.; Lane, S.N. Urban fluvial flood modelling using a two-dimensional diffusion-wave treatment. part 1: Mesh resolution effects. *Hydrol. Process.* **2006**, *20*, 1541–1565. [CrossRef]
32. Neal, J.; Fewtrell, T.; Trigg, M. Parallelisation of storage cell flood models using OpenMP. *Environ. Modell. Softw.* **2009**, *24*, 872–877. [CrossRef]
33. Prestininzi, P.; Di Baldassarre, G.; Schumann, G.; Bates, P.D. Selecting the appropriate hydraulic model structure using low-resolution satellite imagery. *Adv. Water Resour.* **2011**, *34*, 38–46. [CrossRef]
34. Leandro, J.; Chen, A.S.; Schumann, A. A 2D parallel diffusive wave model for floodplain inundation with variable time step (P-DWave). *J. Hydrol.* **2014**, *517*, 250–259. [CrossRef]
35. Hunter, N.M.; Horritt, M.S.; Bates, P.D.; Wilson, M.D.; Werner, G.F. An adaptive time step solution for raster-based storage cell modeling of floodplain inundation. *Adv. Water Resour.* **2005**, *28*, 975–991. [CrossRef]
36. Bates, P.D.; Horrit, M.S.; Fewtrell, T.J. A simple inertial formulation of the shallow water equations for efficient two-dimensional flood inundation modeling. *J. Hydrol.* **2010**, *387*, 33–45. [CrossRef]
37. Dazzi, S.; Vacondio, R.; Dal Palù, A.; Mignosa, P. A local time stepping algorithm for GPU-accelerated 2D shallow water models. *Adv. Water Resour.* **2018**, *111*, 274–288. [CrossRef]
38. Feng, K.; Molz, F.J. A 2-D, diffusion based, wetland flow model. *J. Hydrol.* **1997**, *196*, 230–250. [CrossRef]

39. Gąsiorowski, D.; Szymkiewicz, R. Mass and momentum conservation in the simplified flood routing models. *J. Hydrol.* **2007**, *346*, 51–58. [CrossRef]
40. Gąsiorowski, D. Impact of diffusion coefficient averaging on solution accuracy of the 2D nonlinear diffusive wave equation for floodplain inundation. *J. Hydrol.* **2014**, *517*, 923–935. [CrossRef]
41. Lal, A.M. Performance comparison of overland flow algorithms. *J. Hydraul. Eng. ASCE* **1998**, *124*, 342–349. [CrossRef]
42. Fletcher, C.A.J. *Computational Techniques for Fluid Dynamics*; Springer: Berlin/Heidelberg, Germany, 1991; Volume I.
43. Szymkiewicz, R. Method to solve 1D unsteady transport and flow equations. *J. Hydraul. Eng. ASCE* **1995**, *121*, 396–403. [CrossRef]
44. Szymkiewicz, R. *Numerical Modeling in Open Channel Hydraulics*; Water Science and Technology Library; Springer: New York, NY, USA, 2010. [CrossRef]
45. Dahlquist, G.; Bjorck, A. *Numerical Methods*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1974.
46. Lin, C.A.; Ecer, A.; Periaux, J.; Satofuka, N.; Fox, P. Parallel Computational Fluid Dynamics. In *Development and Applications of Parallel Technology*; Elsevier: Amsterdam, The Netherlands, 1999.
47. Chow, V.T. *Open Channel Hydraulics*; McGraw-Hill Book Company: New York, NY, USA, 1959.