A non-uniform efficient grid type for GPU-parallel Shallow Water Equations models

Contents lists available at ScienceDirect

Environmental Modelling and Software

journal homepage: www.elsevier.com

A non-uniform efficient grid type for GPU-parallel Shallow Water Equations models

Renato Vacondio [a, *], Alessandro Dal Palù [b], Alessia Ferrari [a], Paolo Mignosa [a], Francesca Aureli [a], Susanna Dazzi [a]

[a] Department of Civil and Environmental Engineering and Architecture, University of Parma, Parco Area delle Scienze 181/A, 43124, Parma, Italy
[b] Department of Mathematics and Computer Science, University of Parma, Parco Area delle Scienze 53/A, 43124, Parma, Italy

ARTICLE INFO

Article history:
Received 10 February 2016
Received in revised form 2 August 2016
Accepted 9 November 2016
Available online xxx

ABSTRACT

A GPU-parallel numerical model for the solution of the 2D Shallow Water Equations, based on a novel type of grid called Block-Uniform Quadtree (BUQ), is presented. BUQ grids are based on a data structure which allows to exploit the computational capability of GPUs with minimum overheads, while discretizing the domain with non-uniform resolution. Different cases have been simulated in order to assess the efficiency of the BUQ grids. Theoretical and laboratory tests demonstrate that speed-ups of up to one order of magnitude can be achieved in comparison with uniform Cartesian grids. In the simulation of a hypothetical flood event induced by a levee breach in a real 83 km long river reach, with maximum resolution of 5 m, a ratio of physical to computational time of about 12 was obtained, opening scenarios of quasi real-time 2D simulations in large domains, still retaining a high resolution where necessary.

Software availability

Name of software	G-Flood
Contact address	DICATeA, University of Parma, Parco Area delle Scienze 181/A, 43124 Parma, Italy
Email	renato.vacondio@unipr.it, alessadro.dalpalu@unipr.it
Language	CUDA, C++
Hardware	PC with CUDA-enabled GPU
Availability	upon request by email (for scientific collaboration)
Year first available	2014

1. Introduction

Flooding events in recent years caused many damages and casualties all over the word. Considering climate change and socio-economic effects, Alfieri et al. (2015) pointed out a future increase in the trend of river flood impact in most European countries. If in 2050 the population affected each year is estimated to be in the range of 500,000–640,000, in 2080 the gap is projected to rise up to 540,000–950,000. The same authors estimated the damages related to flooding in Europe to be 20–40 B€ in 2050 and 30–100 B€ in 2080.

After the European Union Directive (2007/60/EC), many countries established national programs to define flood risk maps from global to local scale (De Moel et al., 2015). In such programs the numerical modelling of flooding events remains one of the main instruments through which calculate for instance water depth, velocity and extent of inundation, essential to define flood hazard and then flood risk maps.

Literature offers various examples of numerical models based on 2D Shallow Water Equations (2D-SWEs) suitable to simulate the flood propagation in river reaches where 1D models are not applicable and/or inundations due to levee overtopping/breaching can occur (Lynch and Gray, 1979; Casulli, 1990; Toro, 1999a; Vacondio et al., 2012; Costabile and Macchione, 2015). Among others, explicit Finite Volume (FV) schemes are both robust and accurate, allowing simulations of flood propagation on real irregular bathymetries, transcritical flows, hydraulic jumps and shocks, etc.. Their main drawback is still the high computational cost that makes arduous to run (many) high – resolution simulations on large domains only by means of serial codes. To overcome this issue parallel numerical schemes have been recently developed. Parallelization can be done using a classical approach based on Message Passing Interface (Sanders et al., 2010), that requires access to supercomputers, or Graphics Processing Units (GPUs) (NVIDIA, 2007) which are much cheaper and then suitable for almost all. The use of GPUs is becoming more common in high parallel computation, mainly due to the possibility of drastically reducing the computational times, without increasing the per processor costs. GPU-accelerated models execute the same program in parallel, through various data elements, on Graphic Cards typical of desktop computers. Applications of GPU computing still involve many different research fields, such as medical imaging, oil and gas, physics, mechanical engineering (Sousa et al., 2012), chemistry and biology, design and visualization and fluid dynamics (Caviedes-Voullieme et al., 2014; Crespo et al., 2015), among the others.

In the last years different authors have already managed to develop GPU accelerated SWEs schemes (Crossley et al., 2010; Kalyanapu et al., 2011; De la Asunción et al., 2013; Brodtkorb et al., 2012). More recently, Vacondio et al., 2014a,b implemented an efficient second order finite volume scheme using the Computed Unified Data Architecture (CUDA®). The same model has been used to simulate real floods with high spatial resolution over domains up to 180 km^2 wide, still

* Corresponding author.
Email addresses: renato.vacondio@unipr.it (R. Vacondio); alessandro.dalpalu@unipr.it (A. Dal Palù); alessia.ferrari3@studenti.unipr.it (A. Ferrari); paolo.mignosa@unipr.it (P. Mignosa); francesca.aureli@unipr.it (F. Aureli); susanna.dazzi@studenti.unipr.it (S. Dazzi)

http://dx.doi.org/10.1016/j.envsoft.2016.11.012
1364-8152/

maintaining a ratio between physical and computational times equal to 15 (Vacondio et al., 2016).

However, most of the GPU-accelerated models are based on uniform Cartesian grids, which have two major limitations: (i) the same resolution has to be adopted in the whole domain and (ii) the shape of the domain has to be rectangular. This still prevents the possibility of using such models to simulate floodings over larger areas (order of 10^3 km^2) if somewhere in the domain a resolution of 1–2 m has to be adopted.

In order to resolve small-scale effects in limited areas while simulating large domains, non-uniform grids are widely adopted for non-parallel numerical models, either in the form of unstructured mesh (Begnudelli and Sanders, 2006; Brufau et al., 2004) or hierarchical Quadtree (Greaves and Borthwick, 1999; Borthwick et al., 2000; Rogers et al., 2001; Liang et al., 2007, 2008). The use of non-uniform grids on GPU codes has recently received some attention in literature. Lacasta et al. (2014) developed several optimizations for a flood simulation model based on unstructured triangular grids; the same type of grids has been also adopted in GPU - accelerated models for landslide (Lacasta et al., 2015a), sediment transport (Juez et al., 2016) and rainfall-runoff (Lacasta et al., 2015b) simulations. A full GPU implementation of a SWE model based on Adaptive Mesh Refinement has also been presented by Sætra et al. (2014). Despite all these contributions, the use of non-uniform meshes in GPU-enhanced numerical schemes still remains an ongoing challenge.

Classical data structures used in models based on unstructured or Quadtree meshes cannot exploit the computational capability of GPUs. The main limitation is the arrangement in memory of data and the cost of accessing graph-like data structures (as in typical Quadtree meshes) as opposed to plain matrix memorization, where regularity in accesses provides the best GPU throughput.

With the goal of providing a data structure that combines the benefits of non-uniform resolution meshes and the performances of plain matrix accesses, in the present work a novel Block-Uniform Quadtree grid (BUQ) in a GPU code which solves the 2D-SWE is introduced. The key idea of BUQ grids is to discretize the domain using cell blocks with uniform resolution, while allowing different resolutions for different blocks of cells.

In the present paper the BUQ grids are adopted to simulate theoretical, experimental and real tests following the procedure proposed by Bennett et al. (2013). The quantitative performance has been investigated by comparing the results with analytical solutions, with data acquired in a literature laboratory experiment and collected in the field during a recent real flooding event.

The paper is organized as follows: in Section 1 the numerical model solving the 2D-SWEs is briefly illustrated. The description of the BUQ grids, together with the adopted data structure, is presented in Section 2. In Section 3 the BUQ grid generation procedure is illustrated in detail. Section 4 is dedicated to the evaluation of the model performances by simulating theoretical, laboratory and real test cases. Some conclusion of the work are finally outlined in Section 5.

2. Numerical model

In this Chapter only the main aspects of the numerical model are briefly summarized, as a detailed presentation is beyond the purpose of this work. Major details can be found in Vacondio et al. (2014a,b).

The numerical model solves through a finite volume scheme the 2D-SWE, in the integral form (e.g. Toro, 1999a):

$$\frac{d}{dt}\int_A \mathbf{U}\ dA + \int_C \mathbf{H}\cdot\mathbf{n}\ dC = \int_A \left(\mathbf{S}_0 + \mathbf{S}_f\right)\ dA \tag{1}$$

where A is the area of the integration element, C the element boundary, $\mathbf{n}$ the outward unit vector normal to C, $\mathbf{U}$ the vector of the conserved variables, $\mathbf{H} = (\mathbf{F},\mathbf{G})$ the tensor of fluxes in the x and y directions, $\mathbf{S}$ and $\mathbf{S}_f$ the bed and friction slope source terms, respectively.

In the present work the modified form of SWEs proposed by Liang and Borthwick (2009) is adopted:

$$\mathbf{U} = [\eta;\quad uh;\quad vh\ ]^T$$

$$\mathbf{F} = \begin{bmatrix} uh \\ u^2h + \frac{1}{2}g\left(\eta^2 - 2\eta z\right) \\ uvh \end{bmatrix},\quad \mathbf{G} = \begin{bmatrix} vh \\ uvh \\ v^2h + \frac{1}{2}g\left(\eta^2 - 2\eta z\right) \end{bmatrix}, \mathbf{S} = \begin{bmatrix} 0; \end{bmatrix}$$

in which h is the flow depth, u and v are the velocity components in the x and y directions, g is the gravitational acceleration, z is the bed elevation and $\eta = h + z$ is the free surface elevation above datum. The slope source term is discretized using a centered approximation (Vacondio et al., 2014a,b) but the main advantage of the modified version of the SWEs reported in Equation (2) is that it guarantees the *C-property* defined as the capability of preserving still water at rest (Vázquez-Cendón, 1999), regardless of the adopted discretization form of the slope source term. The procedure proposed by Liang and Marche (2009) for 1D uniform grids has been extended in the present work to guarantee the *C-property* also in presence of 2D non-uniform grids, as reported in detail in Appendix A.

The friction source term S_f is based on the Chezy-Manning equation and it is discretized using the implicit formulation proposed by Caleffi et al. (2003) to avoid instabilities at small water depths.

Fluxes are calculated using a HLLC approximate Riemann solver and second order accuracy in space is ensured by adopting a Monotone Upwind Schemes for Scalar Conservation Laws (MUSCL) interpolation with minmod limiter (Toro, 1999b); further details on the MUSCL reconstruction are reported in Appendix A.

In order to prevent spurious non-physical velocities close to the wet-dry front, the following correction proposed by Kurganov and Petrova (2007) is adopted for the specific discharge:

$$uh_c = uh\frac{\sqrt{2}}{\sqrt{1+\max\left[1,\left(\frac{\varepsilon}{h}\right)^4\right]}} \tag{3}$$

where uh_c is the corrected specific discharge in x – direction and ε is a threshold value for the water depth. An identical correction is applied to the specific discharge in y-direction vh.

The finite volume model is second order also in time: this is achieved adopting the second order Runge-Kutta method:

$$\mathbf{U}_{i,j}^{n+1} = \mathbf{U}_{i,j}^{n} + 0.5\Delta t^n\left[\mathbf{D}_i\left(\mathbf{U}_{i,j}^{n}\right) + \mathbf{D}_i\left(\mathbf{U}_{i,j}^{n+\frac{1}{2}}\right)\right] \tag{4}$$

where n represents the time level, i and j the cell positions, Δx and Δy the grid sizes in x and y directions respectively, and Δt^n is the timestep calculated accordingly to the Courant–Friedrichs-Lewy condition (Toro, 1999a,b).

The operator $\mathbf{D}_i(\mathbf{U}_{i,j})$ is defined as:

$$\mathbf{D}_i\left(\mathbf{U}_{i,j}\right) = -\frac{\left(\mathbf{F}_{i+\frac{1}{2},j} - \mathbf{F}_{i-\frac{1}{2},j}\right)}{\Delta x} - \frac{\left(\mathbf{G}_{i,j+\frac{1}{2}} - \mathbf{G}_{i,j-\frac{1}{2}}\right)}{\Delta y} + \mathbf{S}_0 + \mathbf{S}_f \tag{5}$$

and $\mathbf{U}_{i,j}^{n+1/2}$ is obtained as:

$$\mathbf{U}_{i,j}^{n+\frac{1}{2}} = \mathbf{U}_{i,j}^{n} + \Delta t^n \mathbf{D}_i\left(\mathbf{U}_{i,j}\right) \tag{6}$$

3. Block – Uniform Quadtree grid

The model is implemented in a CUDA/C++ code that exploits parallel computation offered by GPU video cards. Here some basic concepts are recalled and some insight into the novel contributions, namely the capability of handling multiresolution grids, are provided. In CUDA the basic work unit is represented by a thread and many threads are grouped into blocks (NVIDIA, 2007). In the present model each thread corresponds to a computational cell used to discretize the physical domain, and $M \times M$ cells form a regular block. Each block of data can be processed in parallel by a multicore of the GPU taking advantage of fast memory communication, which is fundamental for cell neighborhood information exchange. When the domain is discretized with a Cartesian grid, data are stored as a simple two-dimensional array and therefore cell neighbors localization in memory is straightforward (the coordinate system used in the program corresponds to the matrix indexes used to store the array).

The key idea of BUQ grids is to extend the scenario to a multiresolution system, while preserving the computational organization of a data block. Therefore each block, regardless its cell dimensions, contains $M \times M$ cells, but various resolution levels are used: level 1 with cell size Δ_1, level 2 with cell size $\Delta_2 = 2 \times \Delta_1$ up to level n with cell size $\Delta_n = 2^{n-1} \times \Delta_1$. In the present work M has been assumed equal to 8 or 16, but it can be assumed equal to any power of two. This organization requires a different type of storage of the multiresolution grid: blocks with various resolutions are tiled according to their code index and therefore the original neighborhood relationships among blocks is not maintained in the final tiling. Some additional information allows to retrieve the original position in the grid, its resolution and the neighboring blocks. Because of GPU limitations on threads organization, blocks are actually tiled into a square with a power of two side, large enough to contain all the blocks. Fig. 1 shows an example of arrangement, where three different levels of resolution and 10 blocks are rearranged into a 4 × 4 matrix of blocks. In Fig. 1-a the block position in the physical space is represented, whereas the same blocks are allocated in memory as shown in Fig. 1-b. Please note that different colors correspond to different resolution levels. Blocks with different resolution have different size in the physical space (Fig. 1-a), whereas they have the same size in the memory space (Fig. 1-b), since each block contains information about the same number of cells (8 × 8 or 16 × 16).

The other caveat when handling multiresolution blocks is to retrieve information about a cell's neighbor which lies on a different block (this applies to cells on a block border). The procedure differs significantly from the uniform resolution version, since three cases may apply: the neighbor block may have a higher/same/lower resolution with respect to the current block, and therefore some interpolation may be needed.

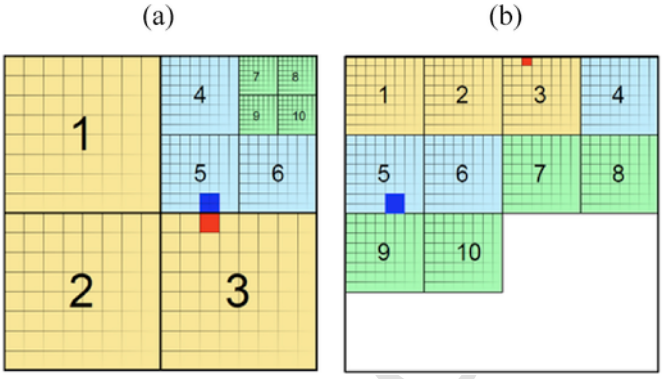


Fig. 1. Example of a BUQ grid represented in the physical space with 8 × 8 cell blocks and three resolution levels (a) and its memory allocation (b).

Moreover, the actual memory location of the required cell(s) has to be retrieved, since close-by blocks in the domain are not necessarily stored in close-by memory locations. As an example, in Fig. 1-a, the cell marked in red belongs to the block 3 and its north neighbor has to be reconstructed from the interpolation of cells marked in blue from the block 5. The spatial adjacency is clearly lost in the actual memorization on the video card (see Fig. 1-b) and it is recovered by on-the-fly index computations.

The preprocessing step determines the set of blocks (patches of $M \times M$ cells) at different resolutions that partition the original high resolution grid. Given n levels of different resolution, the procedure starts with level 1 as the original and highest resolution; each next level reduces the resolution by a factor of two compared to the previous level. During the partition process, the procedure ensures that the number of blocks is kept minimal, while preserving the structural constraint (i.e. adjacent blocks may differ by one resolution level, at most) and data dependent constraints (e.g. specific resolution can be imposed by the user). A minimal number of blocks provides the largest reduction in cells to be computed. Compared to a standard quad-tree like partitioning, the spatial division is equivalent, while the core difference is the association of each node of the quad-tree to a specific block of cells, instead of a single cell. This allows a uniform GPU computation within each block, while keeping an adaptive multi-resolution division of the grid.

The flux computation requires information about the conserved variables of the four neighboring cells, which therefore have to be read from the memory. The procedure that computes one cell's neighbor performs a reconstruction of actual memory addresses, based on precomputed information. This means that a few tests and arithmetic operations are needed to recover the data and thus to optimize the neighbors search algorithm.

Firstly, it is worth noting that the block discretization reduces the overhead by transferring the problem only to cells lying on the block borders, whereas the inner cells have neighbors of the same resolution (grey cells in Fig. 2-a). During the grid partitioning, topological information about neighbors for each block is stored, namely the Cartesian position, eight neighboring blocks references and their resolution levels. In case of a change in resolution between adjacent blocks, two special cases may arise: two higher resolution blocks face a lower resolution block (two neighbors in place of one, Fig. 2-b), and the symmetric case, where a higher resolution block face half lower resolution block (Fig. 2-c). These two cases require extra information to be stored, in order to reconstruct the topological relationships between cells during computation.

When information about a neighboring cell on a different block is requested (see Fig. 1, where the blue cell needs to retrieve the red cell),

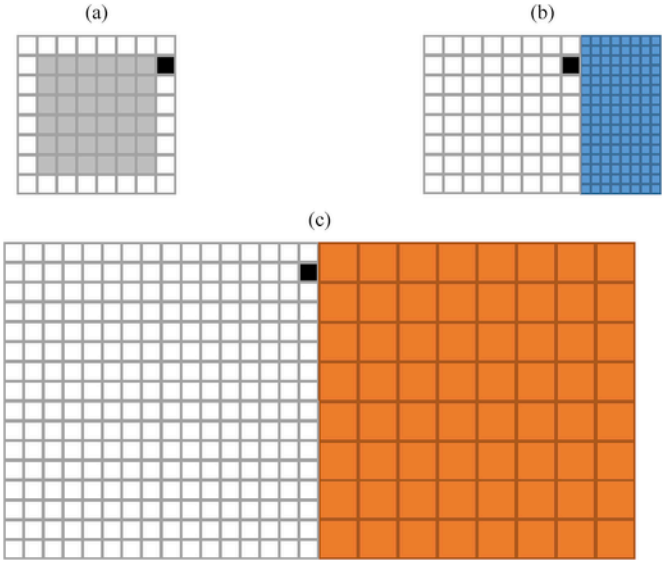


Fig. 2. Neighboring research for a cell (black one) in an 8 × 8 block (a): the neighbor has higher (b) or lower (c) resolution.

the algorithm retrieves the neighbor block address. Moreover, depending on the position of the original cell, it adapts the neighbor cell address in order to select the correct cell position within the block. If a higher resolution is defined in the neighbor block, two different cells need to be addressed.

If the neighbor block has a different resolution, the conserved variables at neighbor grid point n are reconstructed according to the natural neighboring interpolation procedure, with configurations shown in Fig. 3 and the following weights (see for example Liang, 2011).

When quantities have to be updated on cell i and the East block has a higher resolution then the following formula applies (see Fig. 3-a):

$$\mathbf{U}_n = \frac{1}{4}\left(\mathbf{U}_{n_1} + \mathbf{U}_{n_2} + \mathbf{U}_{n_3} + \mathbf{U}_{n_4}\right) \tag{7}$$

Conversely, when the neighbor block has a lower resolution two different sub-cases are possible. If the South-East block has the same resolution of the one where cell i is located, then the interpolation scheme is as shown in Fig. 3-b and the weights are the following ones:

$$\mathbf{U}_n = \frac{\mathbf{U}_i}{4} + \frac{\mathbf{U}_{n_1}}{2} + \frac{\mathbf{U}_{n_2}}{4} \tag{8}$$

Finally, if the South-East block has the same resolution of the East one, then the interpolation scheme is the one shown in Fig. 3-c and the weights are:

$$\mathbf{U}_n = \frac{\mathbf{U}_i}{3} + \frac{\mathbf{U}_{n_1}}{2} + \frac{\mathbf{U}_{n_2}}{6} \tag{9}$$

Interpolations in West, North and South directions can be obtained in the same way by rotating Fig. 3.

4. Grid generation procedure

As described in the Introduction, the aim of the model is to simulate flood propagation over large high-resolution domains, still preserving an accurate geometrical description where necessary.

Fine resolution is typically required in specific zones to accurately reproduce local phenomena: consider for instance the area close to a dam in a dam-break problem, the region of a levee near a breach or a motorway cloverleaf interchange.

The user can impose different resolution zones before the simulation starts, by defining seeding points, each of which has an associated multi-resolution level. Then, an automatic grid generation procedure has been implemented to respect the imposed requirements by means of the seeding points. From an operational point of view seeding points are listed in an input file containing their planimetric coordinates and the associated resolution level.

The algorithm starts by dealing with the highest resolution block (with grid size Δ_1) and then it iterates among the lower resolution ones. Firstly (Algorithm 1 - line 2), it identifies those cells that are required to have the highest resolution and recognizes the blocks with resolution 1 that contain such cells (line 3–4).

There are two rules that are enforced by the algorithm. The first one comes from the quad-tree block division. Recall that a square region (i.e. a block with lower resolution) can be divided into 4 quadrants (i.e.

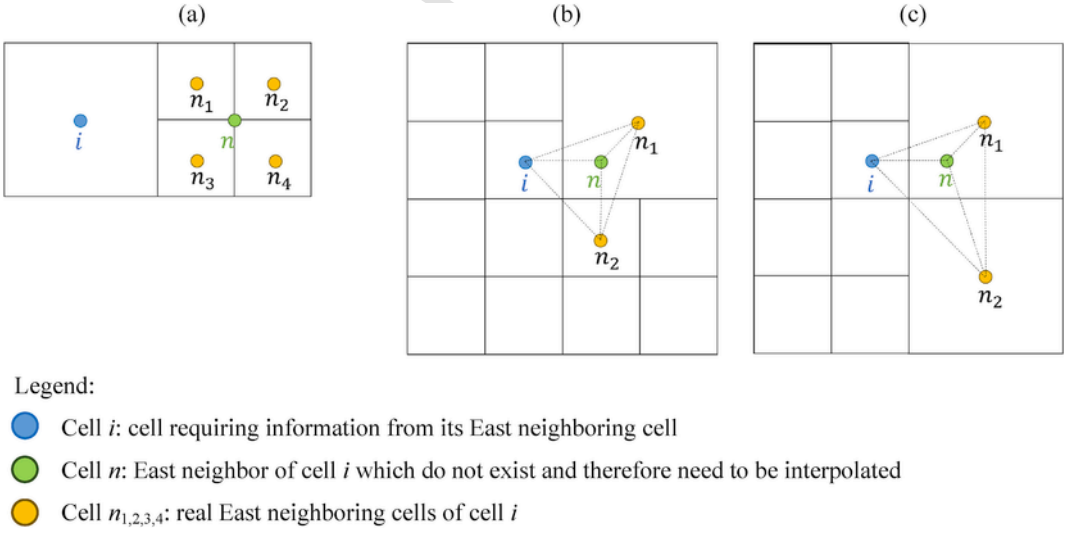


Fig. 3. Interpolation schemes at the interface between different resolution (see also Liang, 2011).

4 blocks with higher resolution). If a subset of these quadrants are identified, the remaining quadrants are forced to be at the same resolution, in order to properly tile the area, since there would be no other mean to assign the other quadrants to a different resolution level (line 5). An example of this procedure is shown in Fig. 4, assuming that four different resolution levels are used and only one seeding point that enforces resolution 1 is assigned. Due to the rule explained before, that seeding point enforces resolution 1 not just to the block that contains the point (the traced one) but also to the other three neighboring blocks that, together with the original one, form a possible block at resolution 2 (see Fig. 4-a).

The second rule forces a controlled decrease in resolution between adjacent blocks in the partition: two neighboring blocks (including diagonal adjacency) cannot differ more than one level in resolution. Therefore, for each selected block at level 1 (line 8), the corresponding block that includes it at resolution level 2 is retrieved (line 9), and the neighbors of such block are forced to resolution level 2, if not already used at level 1 (line 11). Due to this rule, 8 blocks at resolution 2 are identified in Fig. 4-a.

The same procedure is then repeated for each resolution level $i = 2 \ldots n$ (loop at line 1 of Algorithm 1), considering the constraint assigned by the seeding points and those resulted from the resolution level i-1. This means that, in the considered example, when the iteration with resolution levels $i = 2$ and 3 are operated, the second rule forces 12 blocks to be at resolution level 3 (Fig. 4-b) and 4 (Fig. 4-c), respectively.

Algorithm 1: pseudocode to create blocks at different resolution levels

```
1: for each level i=1..n
2:   for each c in cell_forced(level)
3:     b=find_block(c,level)
4:     set_block(b,level)
5:     force_quadrants(b,level)
6:   end
7:   if (level<n)
8:   for each b in set_block(level)
9:     b1=compute_next_level_block(b,level+1)
10:    for each b2=neighbor(b1)
11:      if (b2 not used at level) set_block(b2,level1)
12:    end
13:  end
14:end
```

Besides the downsampling process previously described, the grid generation excludes all the blocks outside the domain, which are not even loaded in the GPU memory. Operationally, the domain is identified by means of a closed polyline (Fig. 5-a), and the information related to the blocks that are located outside are simply not taken into account. In comparison with Cartesian grids, where rectangular domains have to be considered, this leads to a considerable reduction of the allocated memory on the GPU, as it will be shown in Section 4.

Fig. 5 exemplifies the four steps of the grid generation process: in (a) the selected domain and the two seeding points, in which the resolution levels are enforced equal to 1 and 3, respectively, are shown. In (b) all the grey blocks outside the domain are not loaded in the memory. Please note that blocks that are partially inside the polyline are taken into account in the computation. In (c), the resolution levels around the seeding points are defined according to the quad-tree division and, finally, in (d) the resolution of the adjacent blocks is corrected as to maintain the difference of just one level among each other.

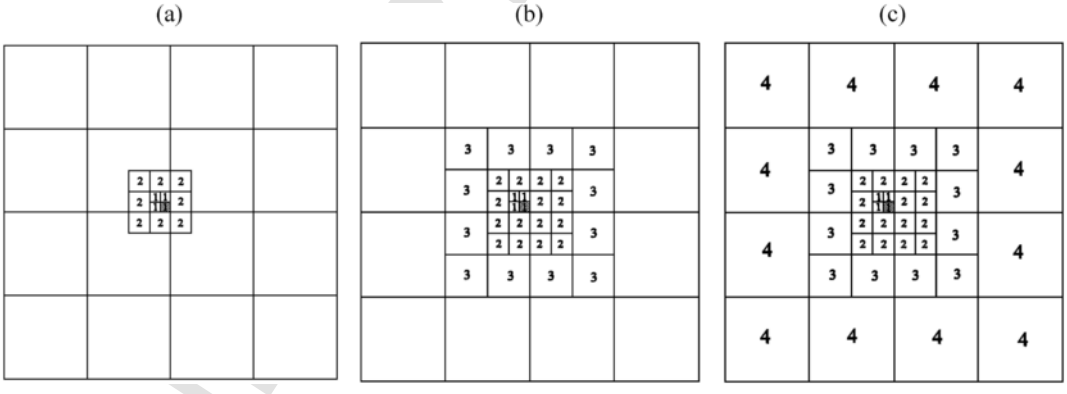


Fig. 4. Example of a grid generation using 1seeding point at resolution 1 and 4 resolution levels.

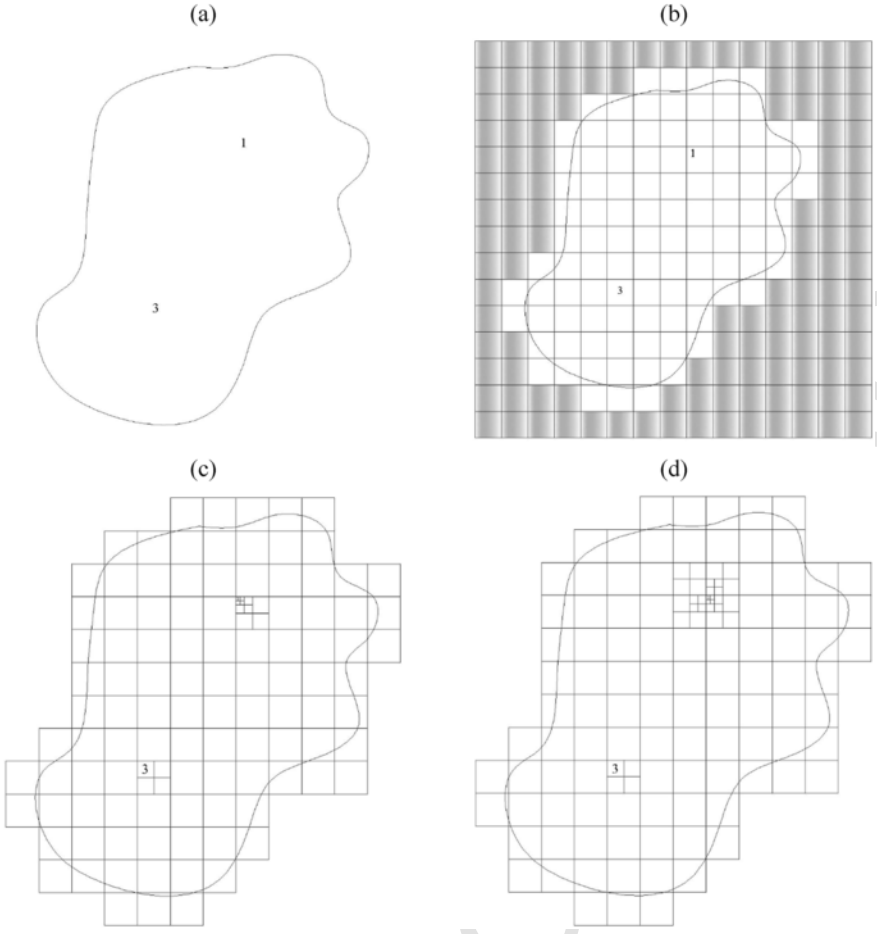


Fig. 5. BUQ grid generation procedure: the domain and two seeding points (a), the blocks which are not loaded in the memory (b), the resolution level definition (c) and the correction of the adjacent blocks (d).

5. Test cases and assessment of the model performances

In this section the performances of the model are discussed through four test cases. Firstly, a theoretical case with analytical solution, focusing on a steady-state vortex flow, is described; then the CADAM (Concerted Action on Dam Break Modelling) (Soares Frazão et al., 1998) dam-break laboratory experiment is reproduced. In the third and fourth test cases a real and an hypothetical flood events are respectively simulated.

All the simulations were run using a K40 T® GPU. For the first three test cases accuracy and efficiency of the simulations are compared against the results obtained with uniform Cartesian grids. The computational efficiency of a particular simulation with BUQ grid is verified by means of its compression rate C_R and speed-up S_U defined as

$$C_R = \frac{N_C}{N} \quad S_U = \frac{T_C}{T} \tag{10}$$

where N and T are the number of allocated cells and the computational time, respectively, for the considered simulation with the BUQ grid; N_c and T_c are the same quantities referred to an identical simulation run using a Cartesian Grid with grid size equal to the maximum resolution adopted in the BUQ grid Δ_1.

Due to the memory requirements, the fourth test case cannot be simulated with uniform Cartesian grid and thus compression rate C_R and speed-up S_U cannot be computed.

5.1. Vortex test case

In this test case a steady vortex circulating clockwise on a frictionless and horizontal bed has been considered. The analytical solution (Sanders and Bradford, 2006) is described in terms of water depth and velocities by the relations:

$$h = h_0 + \frac{U_0^2}{4g}\left[1 - \frac{2r}{r_0}\exp\left(\frac{-2r}{r_0}\right) - \exp\left(\frac{-2r}{r_0}\right)\right] \tag{11}$$

$$u = \frac{U_0\left(y - y_0\right)}{r_0}\exp\left(\frac{-r}{r_0}\right) \tag{12}$$

$$v = \frac{-U_0\left(x - x_0\right)}{r_0}\exp\left(\frac{-r}{r_0}\right) \tag{13}$$

where x_0 and y_0 are the coordinates of the vortex center, $r = \sqrt{x^2 + y^2}$,

h_0 is the water depth at the center of the vortex, U_0 and r_0 are a characteristic velocity and length scales of the vortex, respectively. The parameters were set at the following values: $h_0 = 10$ m, $U_0 = 1.5$ m/s, $r_0 = 100$ m, $x_0 = 0$ m, $y_0 = 0$ m, $r = 2000$ m. The velocity magnitude is equal to zero in the center and reaches the maximum value equal to $U_0 e^{-1}$ at $r = r_0$.

Eighteen different simulations have been performed (Table 1) by varying the resolution, the type of adopted mesh and the size of the cell blocks. In particular three different minimum cell sizes $\Delta x_{min} = 8$, 4, 2 m have been used. For each Δx_{min}, the simulations have been run with three different configurations:

- Cartesian grid with size Δx_{min} (16-C-x and 8-C-x in Table 1);
- BUQ grids with uniform resolution in the whole domain (16-U-x and 8-U-x in Table 1);
- BUQ grids, where Δx_{min} corresponds to the resolution level 1 and varying the resolution inside the domain (16-Q-x and 8-Q-x in Table 1).

If the same Δx_{min} is adopted, the BUQ grid with uniform resolution is identical to the Cartesian one, and thus produces identical results. Nevertheless, as explained in Section 3, the underlying data structure for the BUQ is created in such a way that it can potentially handle non-uniform resolution, and thus the runtimes are different. Moreover, with the aim of analyzing the influence of block size on efficiency, simulations with BUQ grids have been performed with blocks of 8 × 8 and 16 × 16 cells.

In the non-uniform BUQ grids (16-Q-x and 8-Q-x) the highest resolution has been imposed in the region close to the center of the vortex (dashed white circle in Fig. 6), where highest velocity and water depths gradients occur, while decreasing values have been allowed reaching the edge of the domain, up to resolution level 4. Fig. 7 shows the non-uniform BUQ grids 8-Q-1 (a) and 16-Q-1 (b), whereas Fig. 6 shows the detail of the different resolutions for the grid 8-Q-1 in the area close to the center of the vortex: the 1st level corresponds to cells of $\Delta x_1 = 2$ m, the 2nd level to $\Delta x_2 = 4$ m, the 3rd level to $\Delta x_3 = 8$ m and the 4th level to $\Delta x_4 = 16$ m.

It should be noticed that, if blocks of 8 × 8 cells are used, the region with the lower resolution is bigger than in the grid with blocks of 16 × 16 cells and this is confirmed also by the compression rate, equal to 21 and 18, respectively.

As suggested by An and Yu (2014) the simulation was run for 1000 s of physical time, and then the results were compared with the analytical solution. Figs. 8 and 9 show the water depth and velocity magnitude, respectively, along $y = 0$ and $y = x$ lines obtained for the 16-Q-1 and 8-Q-1 simulations. Both numerical results agree very well with the analytical solution.

The L_2 non-dimensional error norm of water depth and velocity components has been calculated as follows:

$$L_2(f) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\frac{f^i_{num} - f^i_{exact}}{f_0}\right]^2} \tag{14}$$

where N is the total number of cells in the domain, f is the variable of interest (water depth, velocity in x and y direction), f_{num} and f_{exact} are the numerical and analytical solutions and, finally, f_0 is a reference value, assumed equal to the parameters h_0 and U_0.

As shown in Table 1 the L_2 error norms for the water depth variable are in the range of 10^{-5}-10^{-6} (close to the machine precision), while 10^{-3}-10^{-4} is the range for the v_x (or v_y) velocity error norm. The L_2 error norms confirm that uniform BUQ grids produce the same results of the Cartesian ones, while the adoption of non-uniform BUQ grids leads to a slight increase in the computed norm values. This is essentially due to the unavoidable approximation introduced by the interpolation of the conserved variables at the interface between different resolutions.

The efficiency of the BUQ grids, in comparison with the Cartesian grid, was then investigated both in term of runtimes and number of cells used to discretize the domain. Fig. 10 shows number of cells and runtimes obtained for the simulations with minimum cell size $\Delta x_{min} = 2$ m. To facilitate the comparison, data have been scaled on the basis of the results obtained for simulation 16-C-1. The simulation run with the non-uniform BUQ grids (16-Q-1 and 8-Q-1) are able to remarkably reduce the memory allocated and the runtimes, in comparison with the analogous simulations with Cartesian grids: the simulation with 8 × 8 cell blocks (8-Q-1) produces both the maximum reduction in terms of number of cells (0.05) and runtime (0.10). Slightly higher − but substantially similar − values are obtained in the (16-Q-1) simulation.

Table 1
ID, grid type, minimum and maximum cell sizes (Δx_{min}, Δx_{max}), number of cells, compression rate C_R, run times, speed-up S_U and L_2 norms for water depth and velocity components for the Vortex simulations.

ID	Grid type	Δx_{min} (m)	Δx_{max} (m)	# cells (10^6)	C_R (−)	Run time (s)	S_U (−)	$L_2(h)$	$L_2(v_x) = L_2(v_y)$
16-C-1	Cartesian	2	2	4.194	1.00	501.38	1.00	4.59E-06	2.45E-04
8-C-1	Cartesian	2	2	4.194	1.00	583.91	1.00	4.59E-06	2.45E-04
16-U-1	BUQ	2	2	3.201	1.31	670.81	0.75	4.59E-06	2.45E-04
8-U-1	BUQ	2	2	3.164	1.33	691.86	0.84	4.59E-06	2.45E-04
16-Q-1	BUQ	2	16	0.240	17.50	54.63	9.18	9.55E-06	9.12E-04
8-Q-1	BUQ	2	16	0.201	20.84	48.30	12.09	1.03E-05	1.03E-03
16-C-2	Cartesian	4	4	1.049	1.00	74.16	1.00	9.94E-06	7.75E-04
8-C-2	Cartesian	4	4	1.049	1.00	84.08	1.00	9.94E-06	7.75E-04
16-U-2	BUQ	4	4	0.816	1.29	89.65	0.83	9.94E-06	7.75E-04
8-U-2	BUQ	4	4	0.797	1.31	97.29	0.86	9.94E-06	7.75E-04
16-Q-2	BUQ	4	32	0.083	12.64	11.70	6.34	2.89E-05	2.44E-03
8-Q-2	BUQ	4	32	0.060	17.39	8.74	9.62	3.36E-05	2.87E-03
16-C-3	Cartesian	8	8	0.262	1.00	12.16	1.00	2.98E-05	2.28E-03
8-C-3	Cartesian	8	8	0.262	1.00	13.12	1.00	2.98E-05	2.28E-03
16-U-3	BUQ	8	8	0.209	1.25	14.23	0.85	2.98E-05	2.28E-03
8-U-3	BUQ	8	8	0.202	1.30	15.67	0.84	2.98E-05	2.28E-03
16-Q-3	BUQ	8	32	0.040	6.56	3.60	3.37	6.82E-05	5.24E-03
8-Q-3	BUQ	8	64	0.022	12.19	2.57	5.10	9.09E-05	7.02E-03

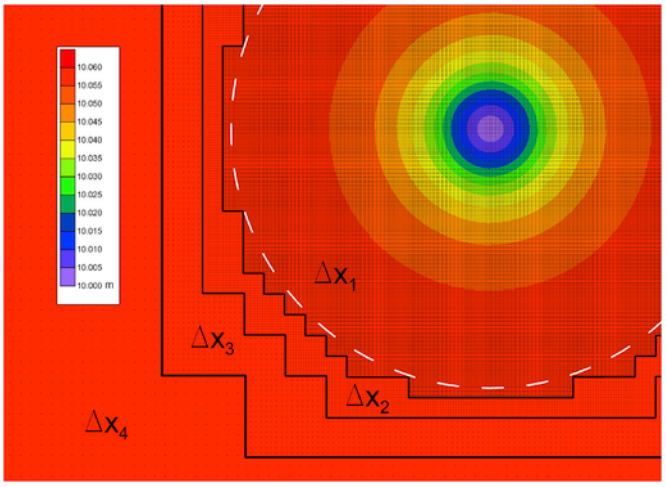


Fig. 6. Detail of the resulted 8-Q-1 non-uniform BUQ grid: evidence of the cell sizes Δx_1, Δx_2, Δx_3, Δx_4. In background the map of the water levels at the end of the simulation.

As explained before and demonstrated by the L_2 error results, all grids with uniform resolution 16-C-1, 8-C-1, 16-U-1 and 8-U-1 are identical. Thus, by simply comparing their runtimes, the overhead produced by the data structure of the BUQ grid can be assessed. The runtimes obtained with the 8 × 8 cell blocks highlight that simulation 8-U-1 is 19% slower (1.38/1.16) than 8-C-1. This is expected because, as explained in Section 2, in BUQ grids the neighbors for cells located at the edge of the blocks are not close in memory as in the Cartesian grids. This leads to a more expensive procedure to identify those neighbors, and to a non – coalesced memory access, which in turn causes the unavoidable overheads. This behavior is further confirmed by the 16-U-1 mesh which, having a smaller ratio between the number of cells on the edges and the total cells of the block, is more efficient than the 8-U-1 one. Similar results can be found by analyzing the data of a simulation performed with different resolution (Table 1).

In order to evaluate the scalability of the non-uniform BUQ grid implementation, in Fig. 11 the normalized runtimes T_N are plotted against the total number of cells. T_N is defined as follows:

$$T_N = \frac{T}{n \cdot N} \tag{15}$$

where T is the total computational time (s), n the number of iterations in time and N the total number of cells. The ratio has been calculated separately for 8 × 8 and 16 × 16 blocks. In both cases the scalability typical of GPUs process is achieved: the more cells are computed, the less time is required for each cell. For more than $0.2 \cdot 10^6$ cells the normalized computational time remains almost constant, showing that the numerical scheme is able to absorb the overheads for communication and thread setup/switch.

5.2. CADAM test case

In order to test the numerical model on a laboratory case concerning a dam-break flow, one of the well-known experiments carried out by the CADAM group was considered (Soares Frazão et al., 1998; Soares Frazão and Zech, 2002). As represented in Fig. 12 the laboratory facility consists in a reservoir connected with a rectangular channel characterized by a sharp 90° bend.

The reservoir is connected with the flat channel bottom through a 0.33 m positive step. At the beginning of the experiment the channel is dry while in the reservoir there is a still water level of 0.53 m (0.2 m upon the downstream channel bed level). The flow is triggered by the instantaneous removal of the gate which separates the reservoir from the downstream channel (Fig. 12).

The Manning coefficient was set equal to 0.012 $s/m^{1/3}$. This value is an average between bottom and walls roughness and allows to reproduce in a correct way the advancing of the wetting front (Soares Frazão et al., 1998; Aureli et al., 2004). At the outflow, a short, fictitious steep lengthening of the channel bottom was added, in order to distance the necessarily approximated boundary condition (far-field). This allows a better reproduction of the cross waves up to the end of the physical channel and, particularly, the time series recorded at the G6 gauge.

The simulations were run for 40 s of physical time with the three different grids:

- Cartesian grid with uniform resolution of Δx_{min} = 0.5 cm (16-C and 8-C);

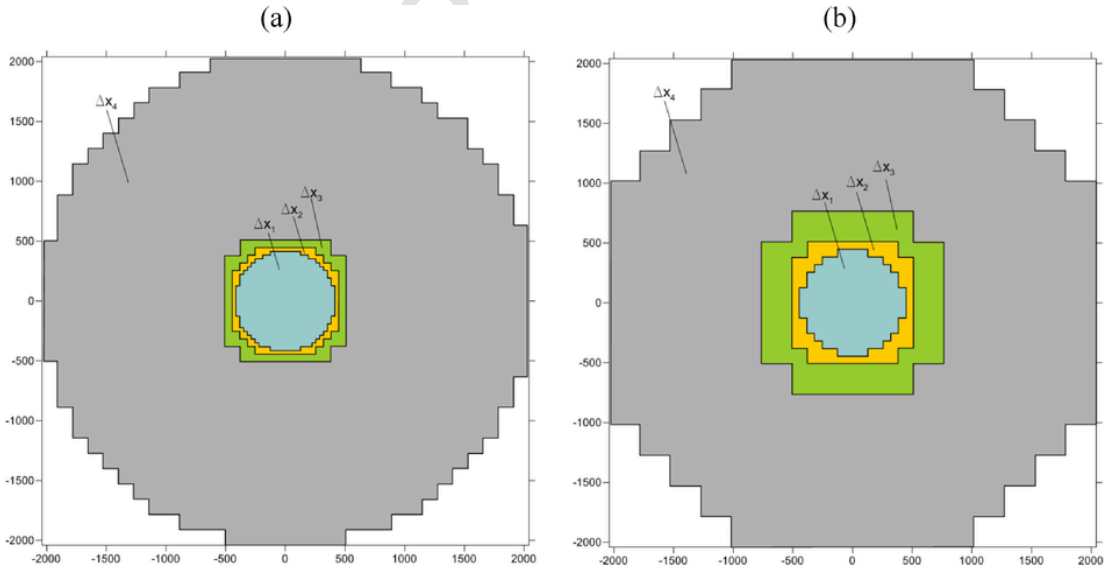


Fig. 7. Resulted non-uniform BUQ grids for the 8-Q-1 (a) and 16-Q-1 (b) simulations.

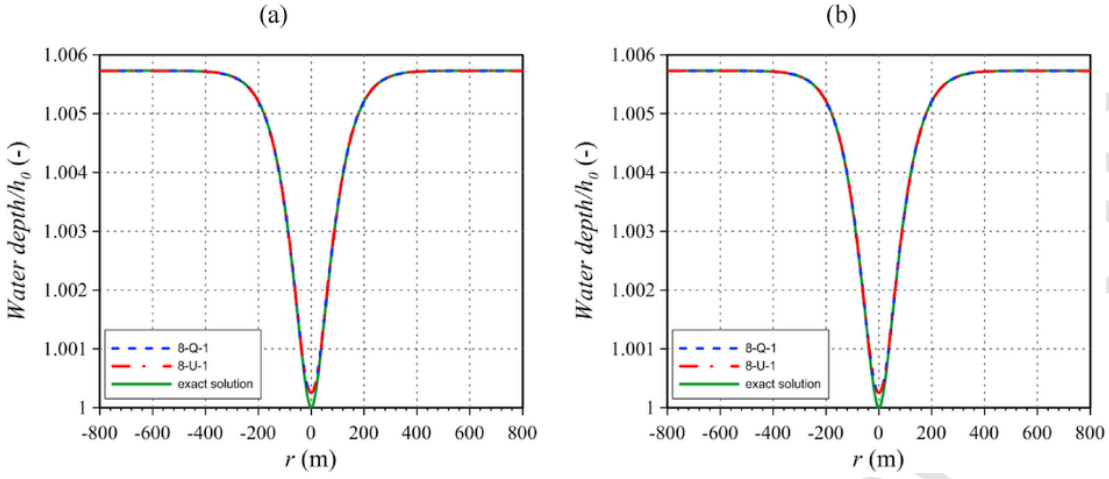


Fig. 8. Uniform (8-U-1) and non-uniform (8-Q-1) BUQ grid: comparison of the water depth trends along the y = 0 (a) and y = x (b) lines.

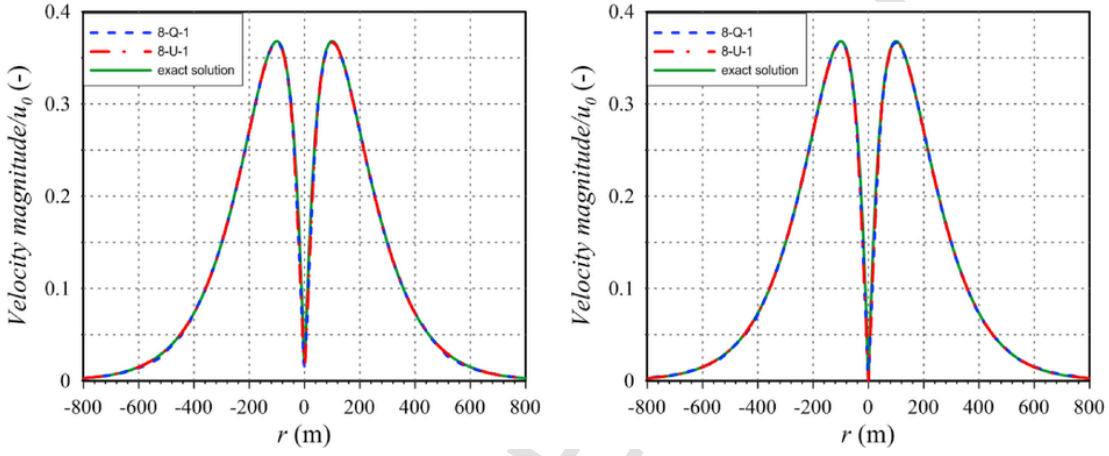


Fig. 9. Uniform (8-U-1) and non-uniform (8-Q-1) BUQ grid: comparison of the velocity magnitude trends along the y = 0 (a) and y = x (b) lines.

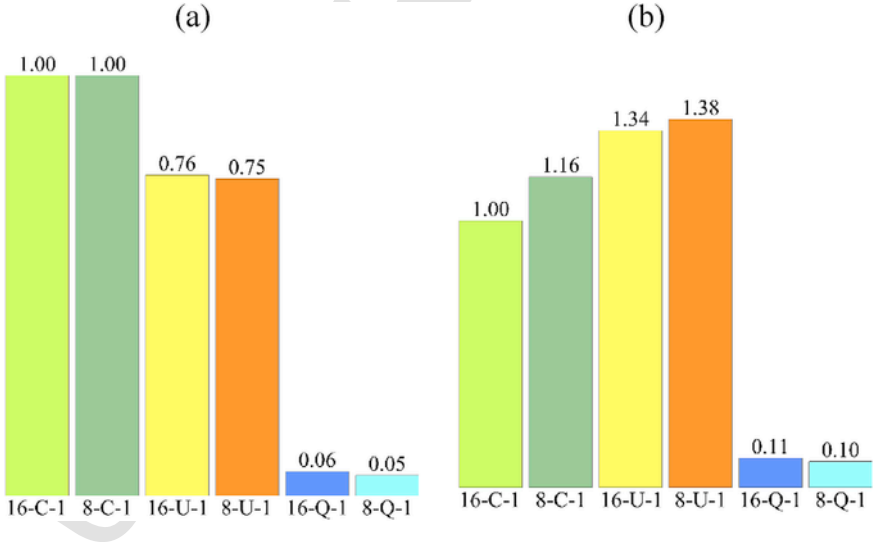


Fig. 10. Vortex test case with minimum cell sizes Δx_{min} = 2 m: number of cells (a) and non-dimensional computational times (b) rescaled on 16-C-1 simulation.

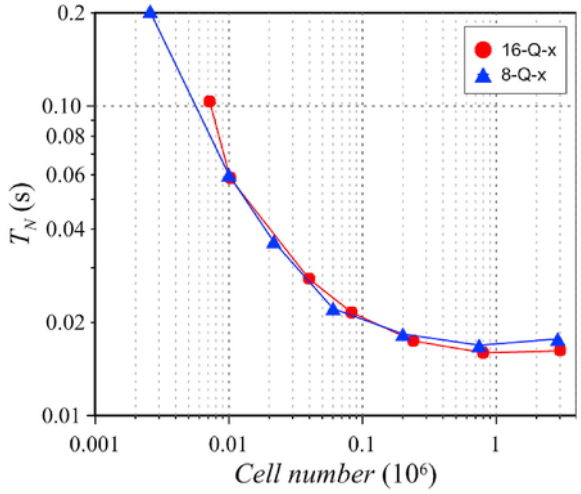


Fig. 11. Analysis of the code scalability using non-uniform BUQ grids.

- BUQ grid with uniform resolution of $\Delta x_{min} = 0.5$ cm (16-U and 8-U);
- BUQ grid with non-uniform resolution with $\Delta x_{min} = 0.5$ cm and $\Delta x_{max} = 4$ cm corresponding to resolution level 4 (16-Q and 8-Q);

and considering the 8 × 8 and 16 × 16 cell blocks for each grid (see Table 2 for details of each simulation). In the BUQ grids with non-uniform resolution (16-Q and 8-Q), the highest resolution level ($\Delta x_1 = 5$ mm) was set only in the channel, while in the upstream reservoir the computed grid sizes gradually increases to 4 cm (Figs. 13 and 14).

Fig. 15 shows the water elevation: (a) 1 s after the gate opening and (b) 10 s after the opening, when the bore reflected by the sharp bend is travelling upstream toward the reservoir and cross-waves characterize the flow downstream the bend.

Fig. 16 shows the comparison between computed and experimental time series at the six gauging points G1÷G6 along the channel. Firstly, it clearly emerges that the phenomenon is correctly described both with non-uniform BUQ (8-Q) grids and with Cartesian ones (8-C). The water elevation at G1 gauge shows the decrease of the initial water elevation in the reservoir (20 cm above the channel flat bottom), initially halved in 14 s and then gradually decreasing till reaching the lowest value of 6 cm at 40 s. At the G2÷G4 gauging points both the initial increase of the water level, due to the arrival of the wetting front and the reflected front, caused by the arrival of the bore at the elbow (which determines an instantaneous raise of the water level) are correctly reproduced. Only the decrease of water levels at G2, before the backward arrival of the bore reflected by the sharp bend, is overestimated by the model. However, this behavior is similar to that obtained by many other authors (Soares Frazão et al., 1998; Liang et al., 2004). Also the numerical results obtained downstream the bend well reproduce the experimental data with the exception of a slight underestimation of the peak of the water depth at gauge G5.

The efficiency of the numerical scheme was investigated comparing the computational time and the number of cells between Cartesian and BUQ grids (Fig. 17). The compression rate C_R for the simulation with uniform BUQ grid is equal to 2.67 and 2.79, adopting 16 × 16 (16-U) and 8 × 8 (8-U) cell blocks, respectively. The remarkable reduction of allocated memory is due to the capability of BUQ grid to exclude grid blocks located outside the domain. Nevertheless, the computational time of simulations with uniform BUQ grids (8-U and 16-U) are similar to the ones obtained with Cartesian grids (8-C and 16-C). This is due the Block Deactivation optimization procedure, which guarantees that cell blocks completely dry or located outside the domain, although allocated in memory, are not processed (Vacondio et al., 2014a,b).

On the other side, the use of non-uniform BUQ grids allows a further reduction of the number of cells used to discretize the domain, reaching the compression rate C_R of 5.87 and 6.59 for the 16 × 16 (16-Q) and 8 × 8 (8-Q) cell blocks, respectively. The simulation with non-uniform grids are also the most efficient in terms of computational times, reaching the speed-up S_U of 1.9 and 2.3 for the 16 × 16 (16-Q) and 8 × 8 (8-Q) cell blocks, respectively.

5.3. October 2014 parma flooding event

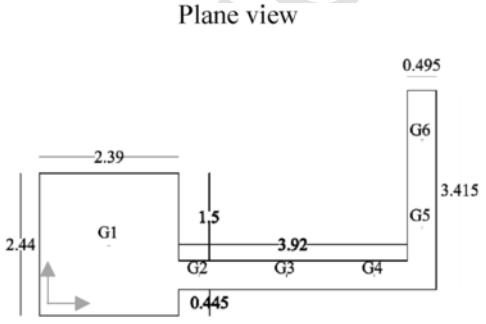
In this section the simulation of the 13th October 2014 flood occurred on the Parma River (Northern Italy), between the homonymous town and the confluence in the Po River, is presented (Fig. 18). This 38 km long reach of the Parma River is characterized by a deep main channel which meanders within two artificial earthen levees of remarkable height above the surrounding lands. In some places flood plains are directly connected with the main channel; elsewhere one or more dikes must be overtopped before the inundation of the flood plain commences.

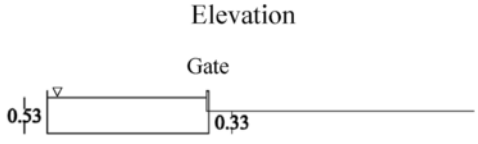
The whole bathymetry (main channel and floodplains) was set up starting from a 1 m resolution DTM obtained through a LiDAR survey carried out during the dry season, in drought condition.

As upstream boundary condition the discharge hydrograph, obtained through a stage-discharge relationship from the recorded water levels at Parma Ponte Verdi gauging station, was imposed. The downstream boundary condition was the reconstructed time-series of water levels at the confluence with the Po River.

The simulations were run with the three different grids:

- Cartesian grid with uniform resolution of $\Delta x_{min} = 2$ m (16-C and 8-C);
- BUQ grid with uniform resolution of $\Delta x_{min} = 2$ m (16-U and 8-U);





	G1	G2	G3	G4	G5	G6
x (m)	1.19	2.74	4.24	5.74	6.56	6.56
y (m)	1.20	0.69	0.69	0.69	1.51	3.01

Fig. 12. CADAM laboratory facility schematization (dimension in m).

Table 2
ID, grid type, minimum and maximum cell sizes (Δx_{min}, Δx_{max}), number of cells, compression rate C_R, run times and speed-up S_U for the CADAM simulations.

ID	Grid type	Δx_{min} (cm)	Δx_{max} (cm)	# cells (10^3)	Compression rate C_R (−)	Run time (s)	Speed-up S_U (−)
16-C	Cartesian	0.5	0.5	1115.48	1.00	318.09	1.00
8-C	Cartesian	0.5	0.5	1115.48	1.00	367.56	1.00
16-U	BUQ	0.5	0.5	418.30	2.67	350.50	0.91
8-U	BUQ	0.5	0.5	400.32	2.79	366.63	1.00
16-Q	BUQ	0.5	4	189.95	5.87	167.70	1.90
8-Q	BUQ	0.5	4	169.34	6.59	159.48	2.30

- BUQ grid with non-uniform resolution with $\Delta x_{min} = 2$ m and $\Delta x_{max} = 16$ m, corresponding to resolution level 4 (8-Q);

and considering the 8 × 8 and 16 × 16 cell blocks for each grid. Table 3 summarizes the details of the six simulations.

Adopting a non-uniform BUQ grid, the resolution level in the main channel has been forced to 2 m close to Colorno (Fig. 18) and 4 m elsewhere. Fig. 19 (a) shows the seeding points used to set the resolution level to 4 m in the main channel. The influence of the block dimension in the non-uniform BUQ grid generation emerges by comparing Fig. 19 (b), obtained with 8 × 8 cell blocks, and (c), obtained with 16 × 16 cell blocks. In this particular meandering reach of the Parma River, the adoption of 16 × 16 cell blocks constrains the same resolution level (4 m) almost everywhere. The 8 × 8 cell blocks, instead, allow varying the resolution outside the main channel, with further reduction of the total number of cells.

Fig. 20 (b) and (c) show a portion of the non-uniform BUQ grids near Colorno, where the resolution level in the main channel was forced to 2 m (Fig. 20-a), adopting 8 × 8 and 16 × 16 cell blocks respectively. Once again, the adoption of 8 × 8 cell blocks allows further reduction of the number of cells, in comparison with the 16 × 16 cell blocks.

The flooding event has been simulated from 12 a.m. of the 13th October 2014, till 2.30 p.m. of the following day, for a total duration of 26.5 h. Because of the particular river morphology previously highlighted, the flow field strongly varies at low (Figs. 21a and 22a), and high discharge values (Figs. 21b and 22b). At the highest levels most of the meanders are cut by the flow, flood plains are inundated and contribute significantly to convey the total discharge, see for example the C and D areas in Figs. 21b and 22b. Under these conditions, it is quite clear that at least a 2D model is necessary to correctly simulate the flood propagation.

During and soon after the flood, the Interregional Agency for the Po River (AIPo) collected the maximum water levels reached in $N = 110$ locations along the river main levees. To evaluate the performance of the simulations the root mean square error, *RMSE*:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(WSE_i^{obs} - WSE_i^{\mathrm{mod}}\right)^2}{N}} \tag{16}$$

between observed and modeled maximum Water Surface Elevations has been calculated and reported in Table 3. The differences of *RMSE* values obtained in the six simulations are almost negligible, confirming that the use of non-uniform BUQ grids does not determine an appreciable loss of accuracy. Instead, the adoption of BUQ grids with uniform resolution guarantees a remarkable reduction of the allocated memory, in comparison with the identical simulation run with Cartesian grid. As the river course resemble the shape of an overturned "L" (Fig. 18), almost $40{\cdot}10^6$ cells must be allocated in a Cartesian grid, even if most of them are located outside the main river levees and thus remain always dry (no levee overtopping nor breaches occur during this simulation). Because the BUQ grids do not require to allocate a rectangular domain, a compression rate C_R of 23 or 25, with 16 × 16 (16-U) or 8 × 8 (8-U) cell blocks, respectively, can be achieved even with uniform BUQ grids. However, as in the previous test case, the compression rate obtained with uniform BUQ grids does not produce a significant reduction of the computational time, thanks to the Block Deactivation Optimization procedure, which excludes the completely dry blocks from the computation even if a Cartesian grid is adopted.

Fig. 23-a highlights this significant reduction in the number of computed cells obtained adopting a non-uniform BUQ grid. Focusing for instance on 8 × 8 cell blocks, assumed the number of cells in the Cartesian grid 8-C as reference (1), the uniform BUQ grid 8-U allocate only the 4.4% (0.044) of cells and, finally, the non-uniform BUQ grid 8-Q compute less than the 1% (0.009) of cells. With non-uniform BUQ grids the values of the compression rate amount to 62 and 109 for the 16 × 16 (16-Q) and 8 × 8 (8-Q) cell blocks, respectively.

The analysis of the computational time (Fig. 23-b) shows that 8.73 h or 11.52 h are required for the simulation with Cartesian grid and 16 × 16 cell blocks or 8 × 8 cell blocks, respectively. The increase of the total computational time in simulation 8-C is due essentially to the code routine that identifies wet blocks. This time, as expected, roughly increases by a factor of 4 in the 8 × 8 Cartesian case, introducing an overhead that slows down the computation by a 1.3 factor.

For the same reason the 8-U simulation performs slightly better than the 8-C one.

As shown in Table 3, the choice of non-uniform BUQ grids allows simulating the flood event in about 3.64 (16 × 16 cell blocks) or 2.60 (8 × 8 cell blocks) hours, with speed-ups equal to 2.4 or 4.4, respectively, and a ratio of physical to computational time of about 7.3 or 10.2.

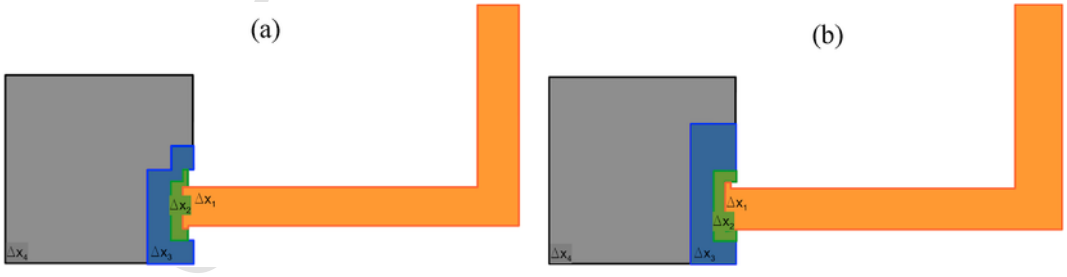


Fig. 13. Resulted non-uniform BUQ grid using blocks with 8 (a) and 16 (b) cells per side.

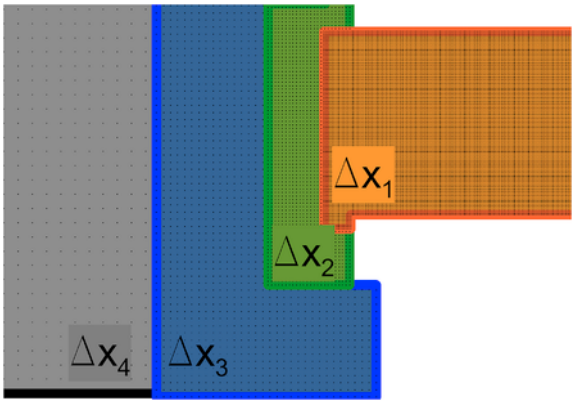


Fig. 14. Detail of the resulted 8-Q non-uniform BUQ grid: evidence of the cell sizes Δx_1, Δx_2, Δx_3, Δx_4.

5.4. Flood inundation generated by a hypothetical levee breach

In order to assess the model performance in the simulation of flooding events over large areas, the numerical results concerning the flooding scenario induced by a hypothetical levee breaching in the Secchia River (Northern Italy) are presented in this section. The domain, whose extension is approximately 840 km^2, includes the 83 km-long reach of the Secchia River between the town of Modena and the confluence in the Po River, and the potentially inundated region located on its right bank, which is enclosed by the Panaro River (to the east) and by a railway embankment (south), as shown in Fig. 24-a. The bathymetry, derived from a 1 m-resolution DTM, is also reported in Fig. 24-a.

The domain is discretized by means of a BUQ grid with non-uniform resolution, with Δx_{min} = 5 m and Δx_{max} = 40 m and 16 × 16 cell blocks. The resolution level is forced to Δx_{min} along the Secchia River (levees included) and along the main road embankments and channel dikes in the flooding plain, while it is allowed to increase gradually to Δx_{max} (according to the algorithm described in Section 3) elsewhere. Fig. 24-b depicts the grid size distribution, the entire domain is discretized by means of $7.3 \cdot 10^6$ cells. The same area would require $99 \cdot 10^6$ and $33 \cdot 10^6$ cells if a Cartesian grid or a BUQ grid with uniform resolution (equal to Δx_{min}) were used, respectively. Please note that this test case cannot be simulated with uniform resolution (both Cartesian and BUQ grids) because the memory requirements exceed the capability of the K40 T® GPU here adopted for the numerical computations.

An inflow discharge hydrograph (50 years-return period) is imposed upstream, while a rating curve is set as downstream boundary condition (very far from the breach location). The levee breach is generated on the right levee at the location indicated in Fig. 24-a after 22 h of simulation, when the peak value of discharge is observed at that cross-section. The simulation is prolonged for 58 h after the breach opening, so that the entire event lasts 80 h. The Manning roughness coefficient is assumed equal to 0.05 s/m$^{1/3}$ everywhere. The first-order accurate version of the scheme is used in this case.

Fig. 25 shows a detail of the flow field around the breach location 2 h after the breach opening. In particular, the water depth contour map is represented in Fig. 25-a, while Fig. 25-b reports both the velocity vector and the velocity magnitude contour map. Different resolution zones are also sketched in Fig. 25-a. It can be noticed that part of the water coming out from the breach moves eastward (and then northward) "canalized" between the levee and a road embankment, while the rest propagates southward. The subsequent flooding evolution in time is outlined in Fig. 26, which reports a selection of water surface elevation maps. An area of approximately 62 km^2 is flooded at the end of the simulation. The total run time is equal to 6.32 h, hence the ratio of physical to computational time is approximately 12.6.

6. Conclusions and outlook

A fast and accurate 2D SWE model is fundamental to simulate flooding events on rivers reaches, like the one here presented, where the velocity field significantly varies, both in magnitude and direction, from low to high discharges during the same event. Not to mention the cases in which levee overtopping or breaching occurs, causing the inundation of large prone areas (Vacondio et al., 2016).

The novel Block-Uniform Quadtree (BUQ) grid, here implemented in a CUDA code which solves the 2D-SWE, exploits the computational capability of GPUs with minimum overheads, allowing at the same time to reproduce small scale effects, thus overcoming one of the main limitations of the GPU codes based on Cartesian Grids. This reduces the computational burden, with a substantial decrease of the number of stored cells and of the runtimes.

The simulations of a theoretical and of a laboratory test case demonstrate that the BUQ grids, compared to the Cartesian grid, allow reaching significant speed-ups (2–10). Moreover, the simulation of a real flood event on a 38 km long river reach, reproduced using a maximum resolution of 2 m, allows to achieve a ratio of physical to computational time of about 10, without appreciable losses of accuracy with respect to a uniformly fine Cartesian Grid. In the simulation of a hypothetical flooding over a domain with extension equal to 840 km^2 the ratio of physical to computational time equal to 12.6 was obtained, confirming the capability of the model to provide quite fast flood simulations.

Since the time step is still constrained, all other things being equal, by the smaller cell size, the use of a local time step, instead of a global one (Dazzi et al., 2016), can further increase the model speed-up using BUQ grids. This, together with the continuously increasing capabilities of the new CUDA-GPU cards, could open scenarios of real-time 2D simulations in large (of the order of 10^3 km^2) domains, still retaining a high resolution where necessary.

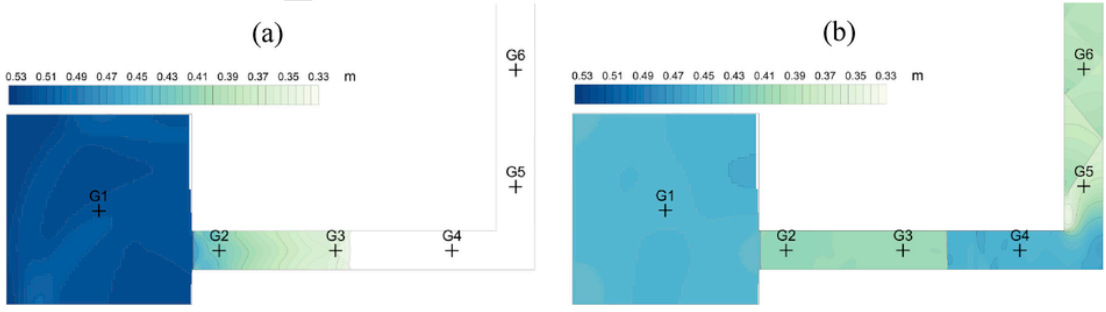


Fig. 15. Water surface elevation for the CADAM test case: 1 s (a) and 10 s (b) after the gate opening.

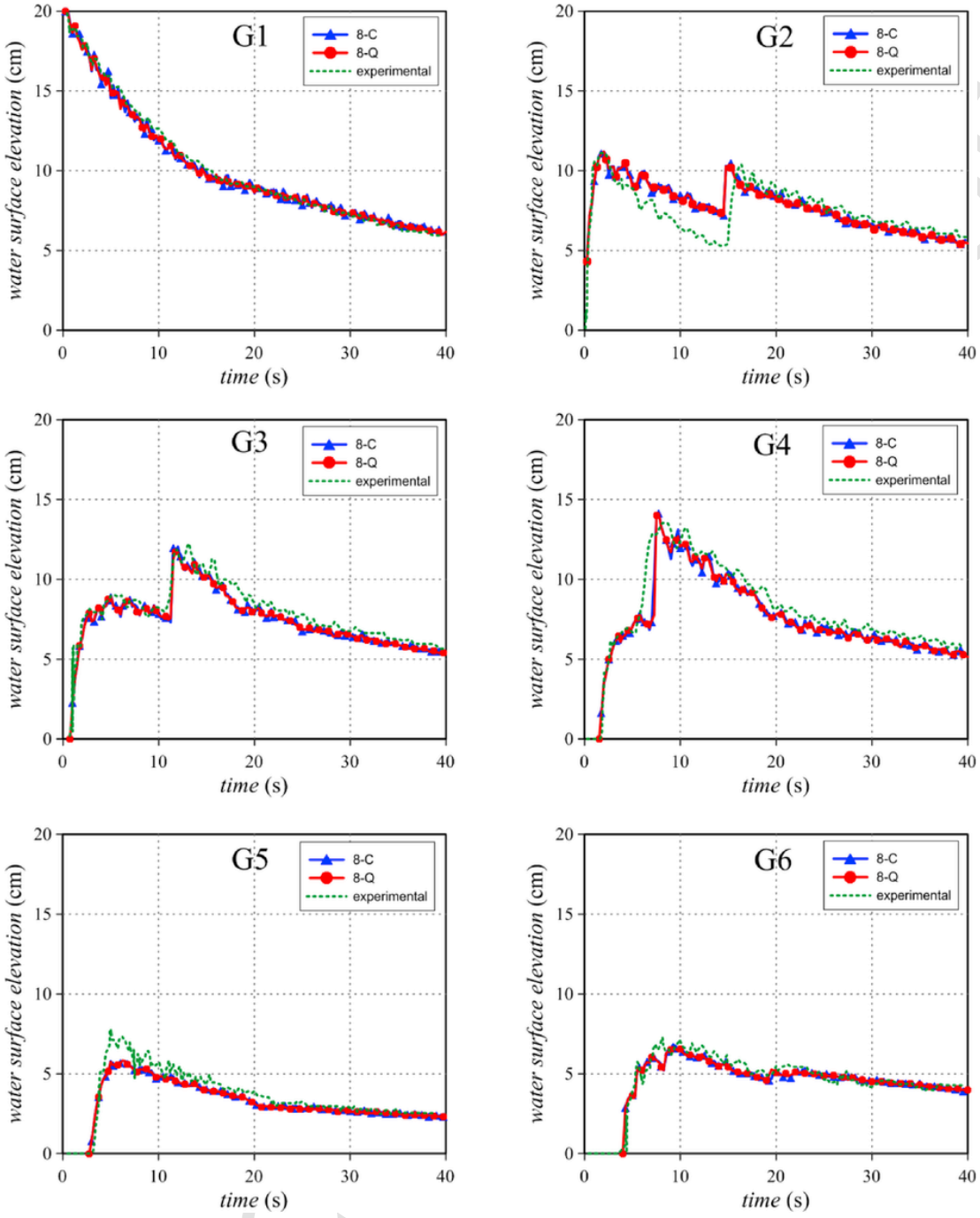


Fig. 16. Comparison between registered and simulated water levels at the six gauging points.

Acknowledgments

This work was partially supported by Ministry of Education, Universities and Research under the Scientific Independence of young Researchers project, grant number RBSI14R1GP, CUP code D92I15000190001. The authors gratefully acknowledge the support of CINECA under project P-FLOOD2-HP10CHAL0S, NVIDIA for providing support under the CUDA Research Center Program and the support of INDAM/GNCS2015 and INdAM - GNCS Project 2016. Interregional Agency for the Po River (AIPo) is also gratefully acknowledged for providing a copious amount of field data collected during the October 2014 Parma flooding event.

Appendix A. Flux and source term calculation

To ensure the *C-property* in a uniform 1D grid Liang and Marche (2009) proposed a formulation which has the power of preserving the water depth positivity. In the present work the same procedure has been extended and generalized to ensure the *C-property* in 2D non-uniform grids, also in presence of wet-dry fronts.

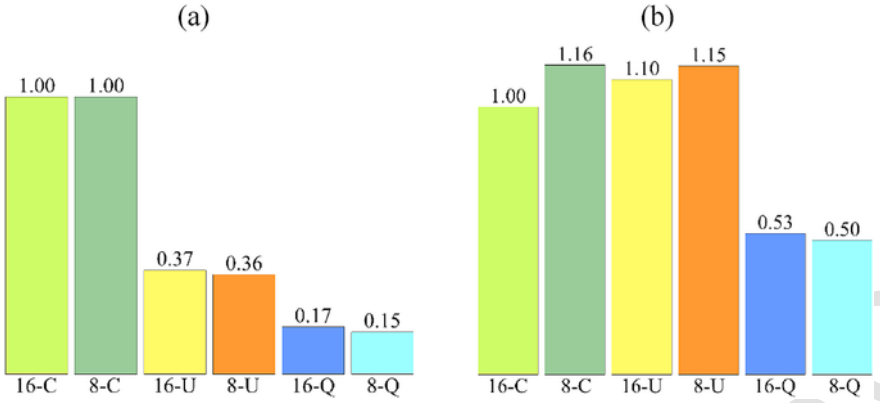


Fig. 17. CADAM test case: number of cells (a) and non-dimensional computational times (b) rescaled on 16-C simulation.

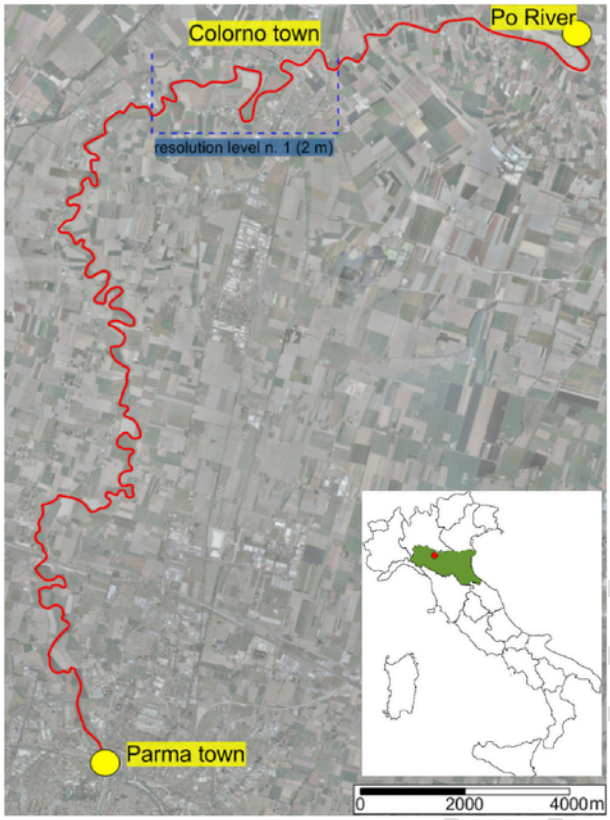


Fig. 18. Parma River reach from Parma to the confluence in the Po River: the locations of the upstream and downstream boundary conditions are indicated.

Table 3
ID, grid type, minimum and maximum cell sizes (Δx_{min}, Δx_{max}), number of cells, compression rate C_R, run times, speed-up S_U and RMSE for the October 2014 Parma flooding event simulations.

ID	Grid type	Δx_{min} (m)	Δx_{max} (m)	# cells (10^6)	Compression rate C_R (−)	Run time (hours)	Speed-up S_U (−)	*RMSE* (m)
16-C	Cartesian	2	2	39.81	1.00	8.73	1.00	0.319
8-C	Cartesian	2	2	39.81	1.00	11.52	1.00	0.326
16-U	BUQ	2	2	1.75	22.76	10.28	0.85	0.316
8-U	BUQ	2	2	1.61	24.67	10.71	1.08	0.323
16-Q	BUQ	2	8	0.64	62.09	3.64	2.40	0.306
8-Q	BUQ	2	16	0.36	109.41	2.60	4.43	0.335

Firstly the conserved variables (η, uh, vh) and the water depth h are reconstructed at the four faces of each cell. For a cell i the reconstructed values at the cell interface i+1/2 on the x coordinate (east edge) are:

$$\eta^L_{i+1/2,x} = \eta_i + \frac{1}{2}\psi_i\left(\eta_i - \eta_{i-1,x}\right) \tag{A.1}$$

$$h^L_{i+1/2,x} = h_i + \frac{1}{2}\psi_i\left(h_i - h_{i-1,x}\right) \tag{A.2}$$

$$uh^L_{i+1/2,x} = uh_i + \frac{1}{2}\psi_i\left(uh_i - uh_{i-1,x}\right) \tag{A.3}$$

$$vh^L_{i+1/2,x} = vh_i + \frac{1}{2}\psi_i\left(vh - vh_{i-1,x}\right) \tag{A.4}$$

where ψ_i is a slope limiter evaluated separately for each considered variable. In the present work the minmod limiter has been adopted. The velocities are calculated as:

$$u^L_{i+1/2,x} = \frac{uh^L_{i+1/2,x}}{h^L_{i+1/2,x}}, v^L_{i+1/2,x} = \frac{vh^L_{i+1/2,x}}{h^L_{i+1/2,x}} \tag{A.5}$$

and forced to zero if the cell is dry. The bed elevation at the same cell interface is calculated as:

$$z^L_{i+1/2,x} = \eta^L_{i+1/2,x} - h^L_{i+1/2,x} \tag{A.6}$$

Then, a single value of bed elevation $z_{i+1/2,x}$ is calculated at each interface. Considering again the interface i+1/2 on the x coordinate:

$$z_{i+1/2,x} = max\left[z^L_{i+1/2,x}, z^R_{i+1/2,x}\right] \tag{A.7}$$

and the water depths on the left and right interfaces are corrected to en-

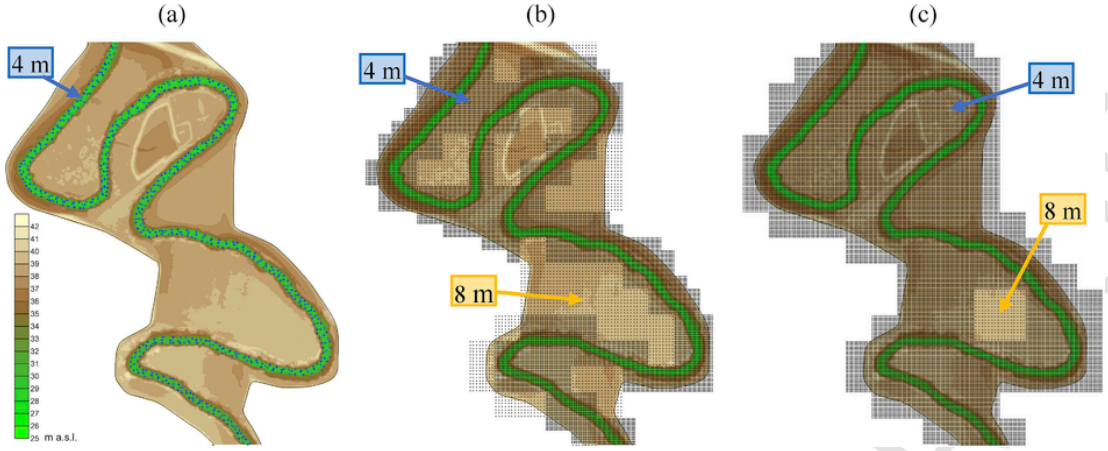


Fig. 19. Detail of the non-uniform BUQ grid generation in a typical meander of the river: the seed points used to set the resolution level in the main channel (a), the resulted grids adopting 8 × 8 (b) and 16 × 16 (c) cell blocks. In background the bathymetry.

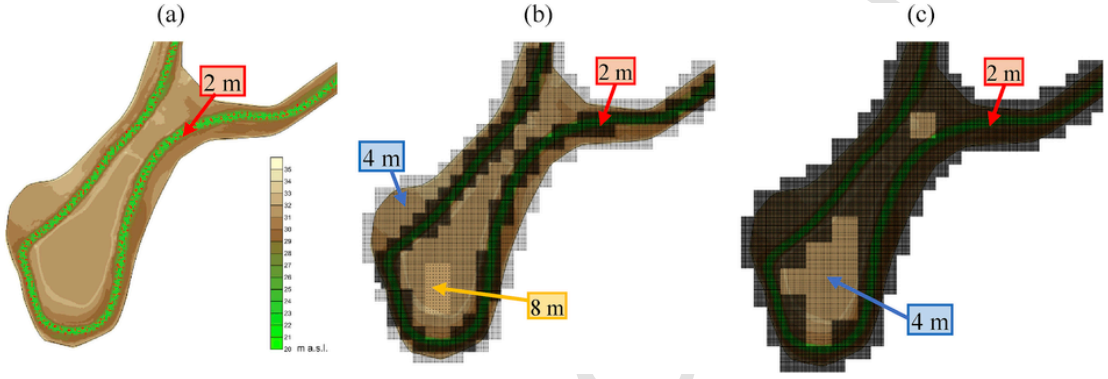


Fig. 20. Detail of the non-uniform BUQ grid generation near Colorno town: the seed points used to set the resolution level in the main channel (a), the resulted grids adopting 8 × 8 (b) and 16 × 16 (c) cell blocks. In background the bathymetry.

sure their positivity:

$$h^{L*}_{i+1/2,x} = max\left[0, \eta^{L}_{i+1/2,x} - z_{i+1/2,x}\right] \;,\; h^{R*}_{i+1/2,x} = max\left[0, \eta^{R}_{i+1/2,x} - z_{i+1/2,x}\right] \tag{A.8}$$

Finally, the Riemann states are calculated as follows:

$$\eta^{L*}_{i+1/2,x} = h^{L*}_{i+1/2,x} + z_{i+1/2,x} \tag{A.9}$$

$$uh^{L*}_{i+1/2,x} = u^{L}_{i+1/2,x} h^{L*}_{i+1/2,x} \tag{A.10}$$

$$vh^{L*}_{i+1/2,x} = v^{L}_{i+1/2,x} h^{L*}_{i+1/2,x} \tag{A.11}$$

$$\eta^{R*}_{i+1/2,x} = h^{R*}_{i+1/2,x} + z_{i+1/2,x} \tag{A.12}$$

$$uh^{R*}_{i+1/2,x} = u^{R}_{i+1/2,x} h^{R*}_{i+1/2,x} \tag{A.13}$$

$$vh^{R*}_{i+1/2,x} = v^{R}_{i+1/2,x} h^{R*}_{i+1/2,x} \tag{A.14}$$

When a non-uniform resolution grid is used, cells with a neighbor at higher resolution have to be updated using two different fluxes to ensure the conservation of mass and momentum (Liang and Borthwick, 2009). In Figure A1 the case of cell *i*, having two cells at higher resolution on the east interface, is shown.

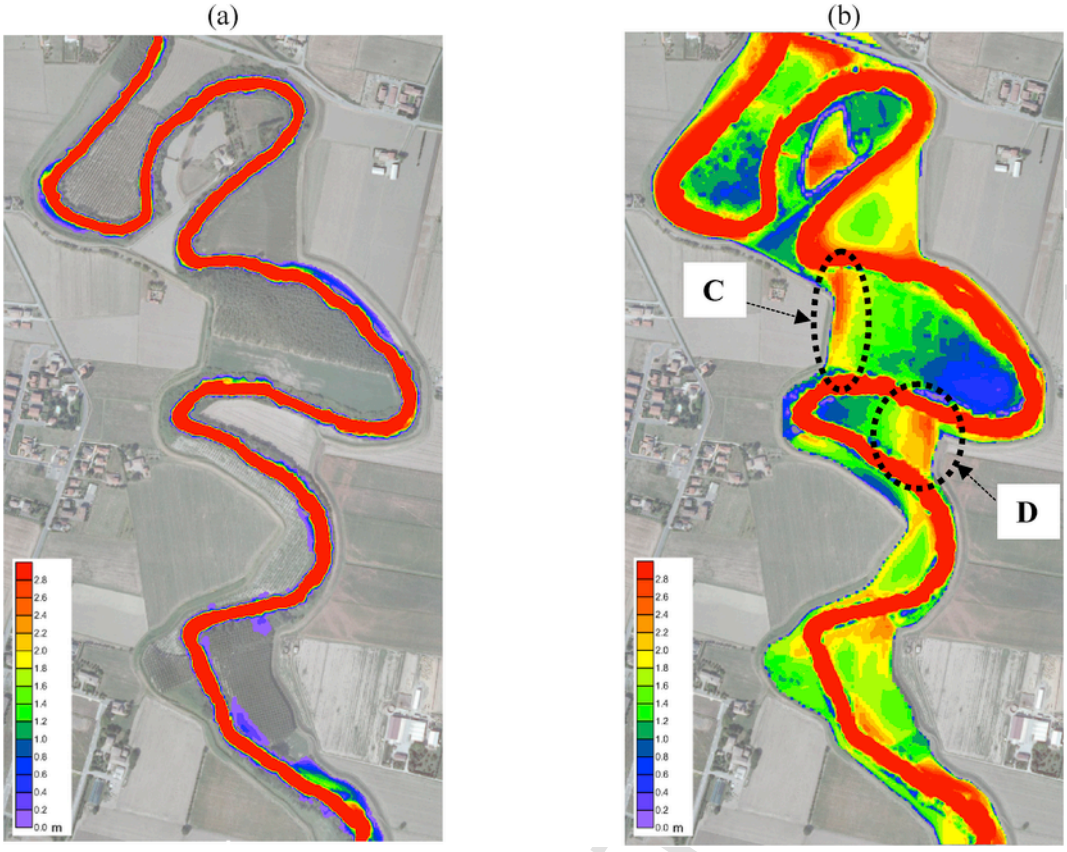


Fig. 21. 8-Q simulation: water depth at low (a) and high (b) discharges in some typical meanders.

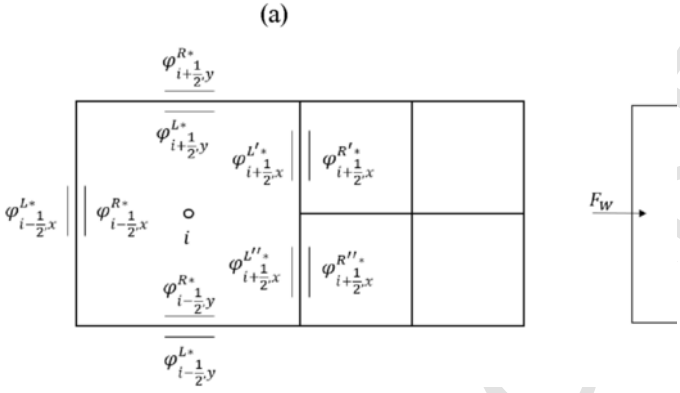


Fig. A1. Fluxes calculation for a cell i with higher resolution neighbors on the east edge: (a) generic reconstructed conserved variables φ (η, uh, vh) at the cell edges, fluxes used to update the conserved variables for cell i (b).

Consistently, if the *C-property* has to be ensured, the slope source term $S_{i,x}$ for cell i has to be calculated as the average of the two terms obtained considering the water surface and the bed elevations reconstructed at the two interfaces i+1/2, along the x direction:

$$S_{i,x} = 0.5\left(S'_{ix} + S''_{ix}\right) \tag{A.15}$$

where

$$S'_{ix} = g\bar{\eta}'_x\left(\frac{z'_{i+1/2,x} - z_{i-1/2,x}}{\Delta x_i}\right) + \bar{S}'_{i-1/2,x} + \bar{S}'_{i+1/2,x} \tag{A.16}$$

$$S''_{ix} = g\bar{\eta}''_x\left(\frac{z''_{i+1/2,x} - z_{i-1/2,x}}{\Delta x_i}\right) + \bar{S}''_{i-1/2,x} + \bar{S}''_{i+1/2,x} \tag{A.17}$$

with $\bar{\eta}'_x = 0.5\left(\eta^{R*}_{i-1/2,x} + \eta^{L'*}_{i+1/2,x}\right)$, $\bar{\eta}''_x = 0.5\left(\eta^{R*}_{i-1/2,x} + \eta^{L''*}_{i+1/2,x}\right)$, and:

$$\bar{S}'_{i-1/2} = g\Delta z_{i-1/2,x}\left[\frac{z'_{i+1/2,x} - \left(z_{i-1/2,x} - \Delta z_{i-1/2,x}\right)}{2\Delta x_i}\right] \tag{A.18}$$

$$\bar{S}'_{i+1/2} = g\Delta z'_{i+1/2,x}\left[\frac{\left(z'_{i+1/2,x} - \Delta z'_{i+1/2,x}\right) - z_{i-1/2,x}}{2\Delta x_i}\right] \tag{A.19}$$

$$\bar{S}''_{i-1/2} = g\Delta z_{i-1/2,x}\left[\frac{z''_{i+1/2,x} - \left(z_{i-1/2,x} - \Delta z_{i-1/2,x}\right)}{2\Delta x_i}\right] \tag{A.20}$$

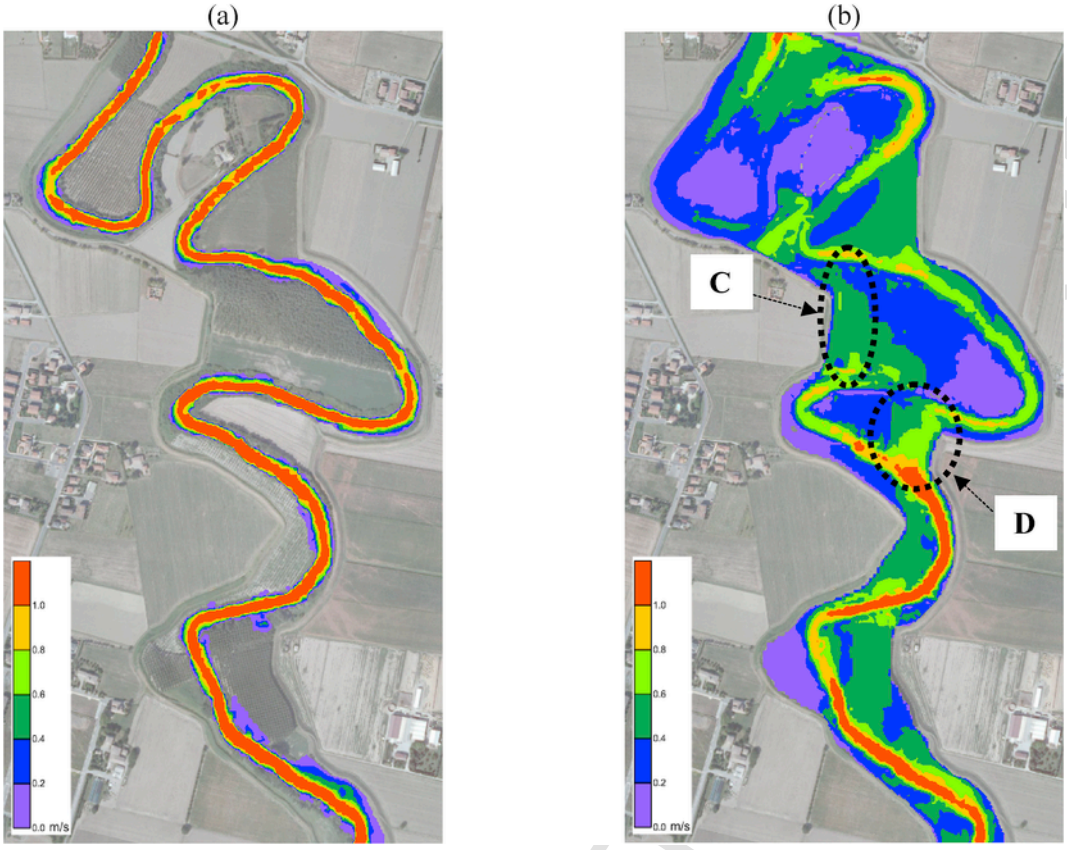


Fig. 22. 8-Q simulation: velocity magnitude at low (a) and high (b) discharges in some typical meanders.

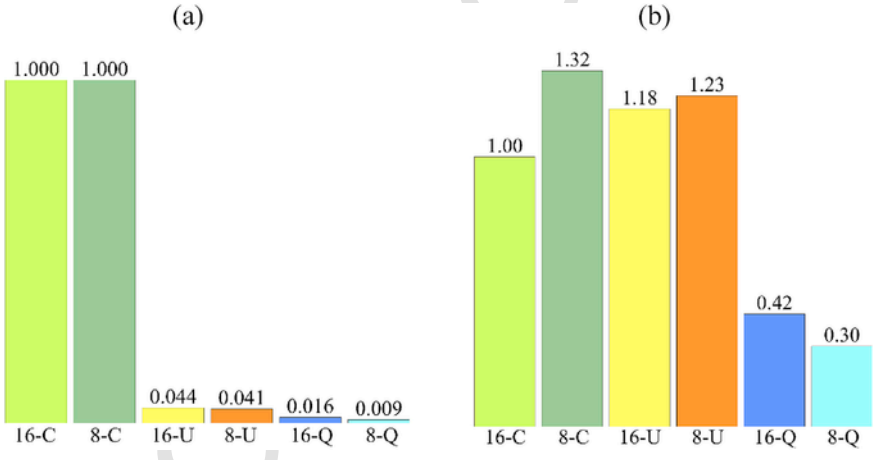


Fig. 23. October 2014 Parma flooding test case: number of cells (a) and non-dimensional computational times (b) rescaled on 16-C simulation.

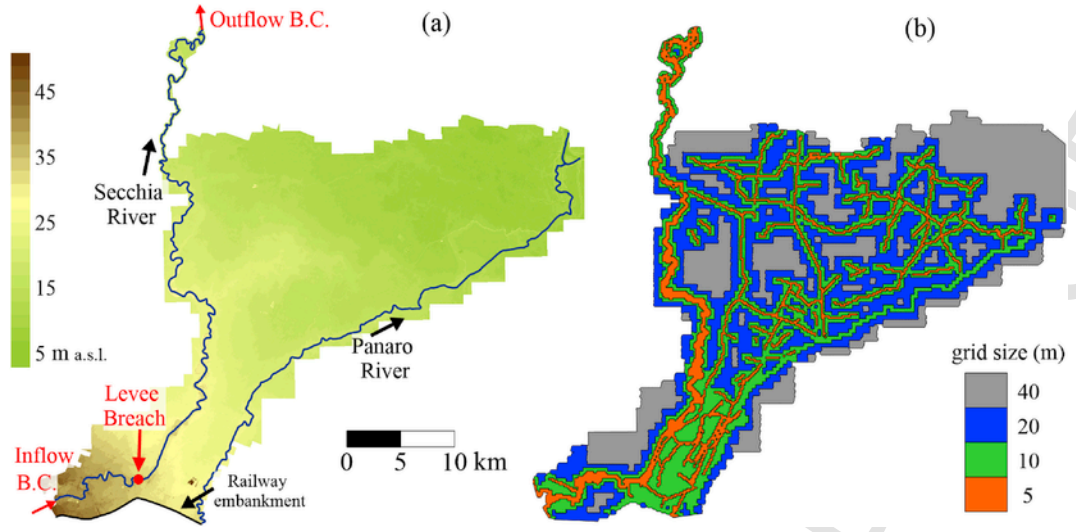


Fig. 24. Levee break on the Secchia River: bathymetry (a) and grid size distribution (b).

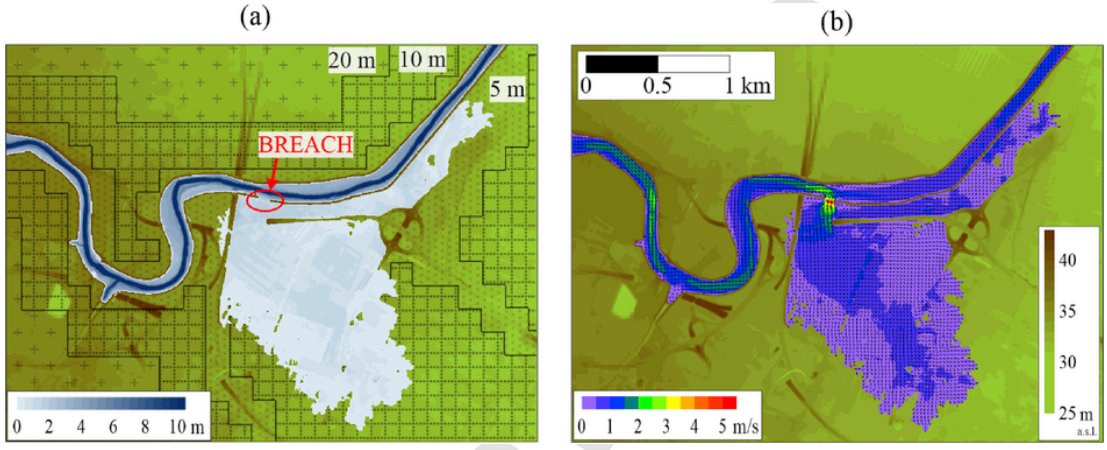


Fig. 25. Flow field around the levee breach: water depth (a) and velocity (b) contour maps.

$$\bar{S}''_{i+1/2} = g\Delta z''_{i+1/2,x}\left[\frac{\left(z''_{i+1/2,x} - \Delta z''_{i+1/2,x}\right) - z_{i-1/2,x}}{2\Delta x_i}\right] \quad \text{(A.21)}$$

where

$$\Delta z'_{i+1/2,x} = max\left[0, -\left(\eta^{L'}_{i+1/2,x} - z'_{i+1/2,x}\right)\right] \quad \text{(A.22)}$$

$$\Delta z_{i-1/2,x} = max\left[0, -\left(\eta^{R}_{i-1/2,x} - z_{i-1/2,x}\right)\right] \quad \text{(A.23)}$$

$$\Delta z''_{i+1/2,x} = max\left[0, -\left(\eta^{L''}_{i+1/2,x} - z''_{i+1/2,x}\right).\right] \quad \text{(A.24)}$$

In this way the *C-property* is guaranteed for BUQ grids, also in presence of wet/dry fronts.

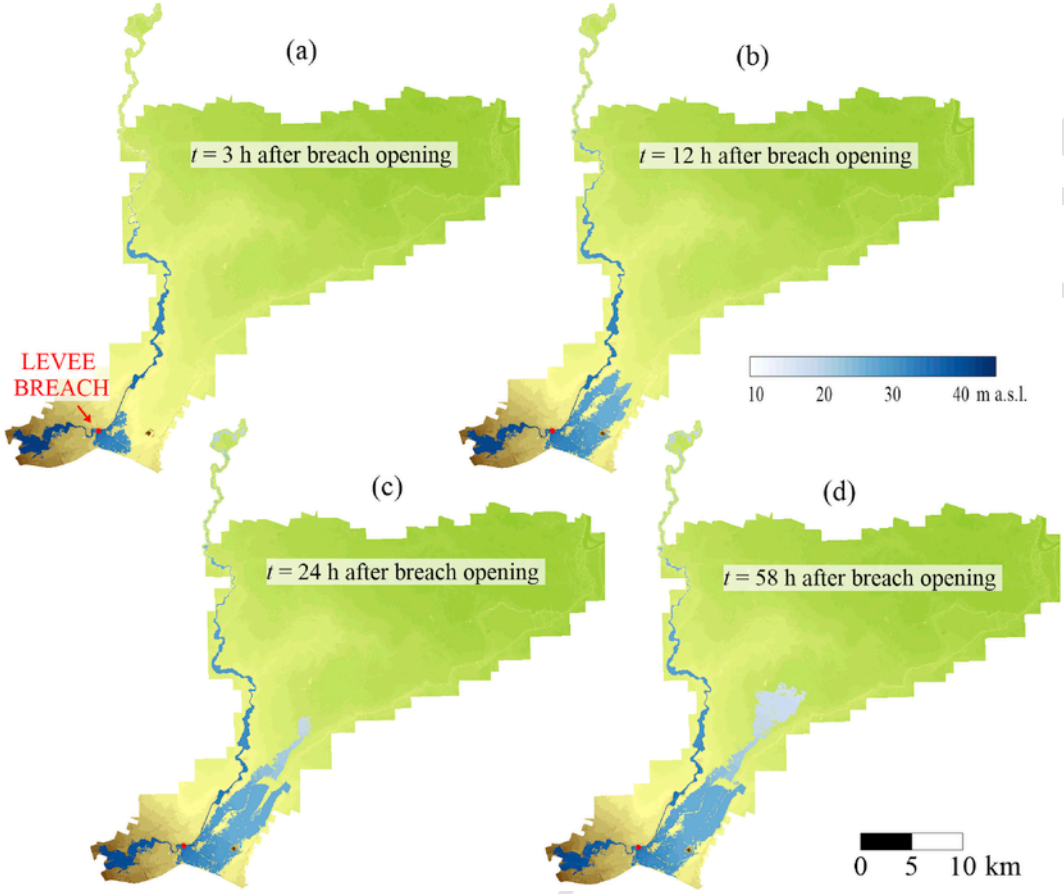


Fig. 26. Water surface elevation maps of the flooding event at selected times: 3 h (a), 12 h (b), 24 h (c) and 58 h (d) after the breach opening.

References

Alfieri, L., Feyen, L., Dottori, F., Bianchi, A., 2015. Ensemble flood risk assessment in Europe under high end climate scenarios. Glob. Environ. Change 35, 199–212.

An, H., Yu, S., 2014. An accurate multidimensional limiter on quadtree grids for shallow water flow simulation. J. Hydraul. Res. 52 (4), 565–574.

Aureli, F., Maranzoni, A., Mignosa, P., 2004. Two dimensional modeling of rapidly varying flows by finite volume schemes. River Flow 2004. Department of Hydraulic and Environmental Engineering. In: Second International Conference on Fluvial Hydraulics. vol. 2. Federico II University, pp. 837–847.

Begnudelli, L., Sanders, B.F., 2006. Unstructured grid finite-volume algorithm for shallow-water flow and scalar transport with wetting and drying. J. Hydraul. Eng. 132.4, 371–384.

Bennett, N.D., Croke, B.F.W., Guariso, G., Guillaume, J.H.A., Hamilton, S.H., Jakeman, A.J., Marsili-Libelli, S., Newham, L.T.H., Norton, J.P., Perrin, C., Pierce, S.A., Robson, B., Seppelt, R., Voinov, A.A., Fath, B.D., Andreassian, V., 2013. Characterising performance of environmental models. Environ. Model. Softw. 40, 1–20.

Borthwick, A.G.L., Marchant, R.D., Copeland, G.J.M., 2000. Adaptive hierarchical grid model of water-borne pollutant dispersion. Adv. Water Resour. 23 (8), 849–865.

Brodtkorb, A.R., Saetra, M.L., Altinakar, M., 2012. Efficient shallow water simulations on GPUs: implementation, visualization, verification, and validation. Comput. Fluids 55, 1–12.

Brufau, P., García-Navarro, P., Vázquez-Cendón, M.E., 2004. Zero mass error using unsteady wetting–drying conditions in shallow flows over dry irregular topography. Int. J. Numer. Meth. Fluids 45, 1047–1082.

Caleffi, V., Valiani, A., Zanni, A., 2003. Finite volume method for simulating extreme flood events in natural channels. J. Hydraul. Res. 41, 167–177.

Casulli, V., 1990. Semi-implicit finite difference methods for the two-dimensional shallow water equations. J. Comp. Phys. 86 (1), 56–74.

Caviedes-Voullieme, D., Morales-Hernández, M., López-Marijuan, I., García-Navarro, P., 2014. Reconstruction of 2D river beds by appropriate interpolation of 1D cross-sectional information for flood simulation. Environ. Model. Softw. 61, 206–228.

Costabile, P., Macchione, F., 2015. Enhancing river model set-up for 2-D dynamic flood modelling. Environ. Model. Softw. 67, 89–107.

Crespo, A.J.C., Domínguez, J.M., Rogers, B.D., Gómez-Gesteira, M., Longshaw, S., Canelas, R., Vacondio, R., Barreiro, A., García-Feal, O., 2015. DualSPHysics: open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH). Comput. Phys. Commun. 187, 204–216.

Crossley, A., Lamb, R., Waller, S., 2010. Fast solution of the shallow water equations using GPU technology. In: Proceedings of the British Hydrological Society 3rd International Symposium, Newcastle, UK, 13–19 July 2010.

Dazzi, S., Maranzoni, A., Mignosa, P., 2016. Local time stepping applied to mixed flow modelling. J. Hydraulic Res. http://dx.doi.org/10.1080/00221686.2015.1132276.

De la Asunción, M., Castro, M.J., Fernández-Nieto, E., Mantas, J.M., Acosta, S.O., González-Vida, J.M., 2013. Efficient GPU implementation of a two waves TVD-WAF method for the two-dimensional one layer shallow water system on structured meshes. Comput. Fluids 80, 441–452.

De Moel, H., Jongman, B., Kreibich, H., Merz, B., Penning-Rowsell, E., Ward, P.J., 2015. Flood risk assessments at different spatial scales. Mitig. Adapt. Strategies Glob. Change 1–26.

Greaves, D.M., Borthwick, A.G.L., 1999. Hierarchical tree-based finite element mesh generation. Int. J. Numer. Meth. Eng. 45 (4), 447–471.

Juez, C., Lacasta, A., Murillo, J., García-Navarro, P., 2016. An efficient GPU implementation for a faster simulation of unsteady bed-load transport. J. Hydraulic Res. 54 (3), 275–288. http://dx.doi.org/10.1080/00221686.2016.1143042.

Kalyanapu, A.J., Shankar, S., Pardyjak, E.R., Judi, D.R., Burian, S.J., 2011. Assessment of GPU computational enhancement to a 2D flood model. Environ. Model. Softw. 26 (8), 1009–1016.

Kurganov, K., Petrova, G., 2007. A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system. Commun. Math. Sci. 5 (1), 133–160.

Lacasta, A., Morales-Hernández, M., Murillo, J., García-Navarro, P., 2014. An optimized GPU implementation of a 2D free surface simulation model on unstructured meshes. Adv. Eng. Softw. 78, 1–15. http://dx.doi.org/10.1016/j.advengsoft.2014.08.007.

Lacasta, A., Juez, C., Murillo, J., García-Navarro, P., 2015a. An efficient solution for hazardous geophysical flows simulation using GPUs. Comput. Geosciences 78 (0), 63–72. http://dx.doi.org/10.1016/j.cageo.2015.02.010.

Lacasta, A., Morales-Hernández, M., Murillo, J., García-Navarro, 2015b. GPU implementation of the 2D shallow water equations for the simulation of rainfall/runoff events. Environ. Earth Sci. 74 (11), 7295–7305. http://dx.doi.org/10.1007/s12665-015-4215-z.

Liang, Q., Borthwick, A.G.L., Stelling, G., 2004. Simulation of dam- and dyke-break hydrodynamics on dynamically adaptive quadtree grids. Int. J. Numer. Meth. Fluids 46, 127–162.

Liang, Q., Zang, J., Borthwick, A.G., Taylor, P.H., 2007. Shallow flow simulation on dynamically adaptive cut cell quadtree grids. Int. J. Numer. Meth. Fluids 53 (12), 1777–1799.

Liang, Q., Du, G., Hall, J.W., Borthwick, A.G., 2008. Flood inundation modeling with an adaptive quadtree grid shallow water equation solver. J. Hydraul. Eng. 134 (11), 1603–1610.

Liang, Q., Borthwick, A.G.L., 2009. Adaptive quadtree simulation of shallow flows with wet-dry fronts over complex topography. Comput. Fluids 38, 221–234.

Liang, Q., Marche, F., 2009. Numerical resolution of well-balanced shallow water equations with complex source terms. Adv. Water Resour. 32, 873–884.

Liang, Q., 2011. A structured but non-uniform Cartesian grid-based model for the shallow water equations. Int. J. Numer. Meth. Fluids 66, 537–554.

Lynch, D.R., Gray, W.G., 1979. A wave equation model for finite element tidal computations. Comput. Fluids 7, 207–228.

NVIDIA CUDA, 2007. Compute Unified Device Architecture Programming Guide.

Rogers, B., Fujihara, M., Borthwick, A.G., 2001. Adaptive Q-tree Godunov-type scheme for shallow water equations. Int. J. Numer. Meth. Fluids 35 (3), 247–280.

Sætra, M.L., Brodtkorb, A.R., Lie K, A., 2014. Efficient GPU-implementation of adaptive mesh refinement for the shallow-water equations. J. Sci. Comp. 63 (1), 23–48.

Sanders, B.F., Bradford, S.F., 2006. Impact of limiters on accuracy of high-resolution flow and transport models. J. Eng. Mech. 132, 87–98.

Sanders, B.F., Schubert, J.E., Detwiler, R.L., 2010. Parbrezo: a parallel, unstructured grid, godunov-type, shallow-water code for high-resolution flood inundation modeling at the regional scale. Adv. Water Resour. 33 (12), 1456–1467. ISSN 0309–1708.

Soares Frazão, S., Sillen, X., Zech, Y., 1999. Dam-break flow through sharp bends physical model and 2D boltzmann model validation. In: Proc., CADAM Meeting Wallingford, U.K., 2–3 March 1998. European Commission, Brussels, Belgium, pp. 151–169.

Soares Frazão, S., Zech, Y., 2002. Dam break in channels with 90° bend. J. Hydraul. Eng. 128, 956–968.

Sousa, F.A., Dos Reis, R.J.N., Pereira, J.C.F., 2012. Simulation of surface fire fronts using fireLib and GPUs. Environ. Model. Softw. 38, 167–177.

Toro, E., 1999a. Shock Capturing Methods for Free Surface Shallow Water Flows. Wiley, New York.

Toro, E., 1999b. Riemann Solvers and Numerical Methods for Fluid Dynamics. Springer.

Vacondio, R., Rogers, B.D., Stansby, P.K., Mignosa, P., 2012. SPH modeling of shallow flow with open boundaries for practical flood simulation. J. Hydraul. Eng. 138, 530–541.

Vacondio, R., Dal Palù, A., Mignosa, P., 2014a. GPU-enhanced Finite Volume Shallow Water solver for fast flood simulations. Environ. Model. Softw. 57, 60–75.

Vacondio, R., Aureli, F., Ferrari, A., Mignosa, P., Dal Palù, A., 2016. Simulation of the January 2014 flood on the Secchia River using a fast and high-resolution 2D parallel shallow-water numerical scheme. Nat. Hazards 80 (1), 103–125. http://dx.doi.org/10.1007/s11069-015-1959.

Vázquez-Cendón, M.E., 1999. Improved treatment of source terms in upwind schemes for the shallow water equations in channels with irregular geometry. J. Comput. Phys. 148 (2), 497–526.