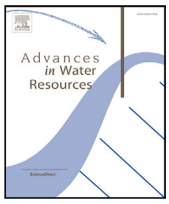
Advances in Water Resources 197 (2025) 104903

Contents lists available at ScienceDirect

Advances in Water Resources

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



Solving the discretised shallow water equations using neural networks

Boyang Chen [a,*], Amin Nadimy [a,b], Claire E. Heaney [a,d], Mohammad Kazem Sharifian [b], Lluis Via Estrem [b], Ludovico Nicotina [c], Arno Hilberts [b], Christopher C. Pain [a,d,e]

[a] Applied Modelling and Computation Group, Department of Earth Science and Engineering, Imperial College London, South Kensington Campus, London, SW7 2AZ, UK
[b] Moody's, 21 Mincing Lane, London, EC3R 7AG, UK
[c] Inigo Insurance Ltd., One Creechurch Place, London, EC3A 5A, UK
[d] Imperial-X, Imperial College London, White City Campus, London, W12 7SL, UK
[e] Data Assimilation Laboratory, Data Science Institute, Imperial College London, South Kensington Campus, London, SW7 2AZ, UK

ARTICLE INFO

Dataset link: GitHub Repository

Keywords:
Shallow water equations
Neural networks
Petrov–Galerkin stabilisation
Finite element method

ABSTRACT

We present a new approach to the discretisation and solution of the Shallow Water Equations (SWE) based on the finite element (FE) method. The discretisation is expressed as the convolutional layer of a neural network whose weights are determined by integrals of the FE basis functions. The resulting system can be solved with explicit or implicit methods. Expressing and solving discretised systems with neural networks has several benefits, including platform-agnostic code that can run on CPUs, GPUs as well as the latest processors optimised for AI workloads; the model is fully differentiable and suitable for performing optimisation tasks such as data assimilation; easy integration with trained neural networks that could represent sub-grid-scale models, surrogate models or physics-informed approaches; and speeding up the development of models due to the available functionality in machine-learning libraries. In this paper, we investigate explicit and semi-implicit methods, and FE discretisations of up to quartic-order elements. A variety of examples is used to demonstrate the neural-network–based SWE solver, ranging from idealised problems with analytical solutions to laboratory experiments, and we finish with a real-world test case based on the 2005 Carlisle flood.

1. Introduction

Hydrodynamic models employ the principles of fluid dynamics to predict the behaviour and interaction of fluids in various environmental and industrial contexts, such as marine science (Dobbelaere et al., 2020), material science (Zeiner and Fischlschweiger, 2023) and flood hazards (Yalcin, 2020), offering valuable insights into complex fluid systems. The importance of flood modelling lies in the desire to understand the dynamics associated with flooding events and to formulate strategies for mitigation. In turn, this aids informed decision-making for effective flood risk management (Spiekermann et al., 2015). High-resolution numerical models of flood inundation are also used in catastrophe risk models, which are widely adopted for insurance portfolio risk assessments (Zanardo and Salinas, 2022). In these use cases, the ability of the models to solve physical equations with high fidelity on large spatial scales and at high spatial resolution is paramount. Hydrodynamic computational models such as TUFLOW (Tuflow, 2023), TRITON (Triton, 2022), LISFLOOD (Shaw et al., 2021) and InfoWorks (Innovyze, 2019) solve the two-dimensional Shallow Water Equations (SWEs) typically by using the finite difference method (Lundgren and Mattsson, 2020; Madsen et al., 2005), the finite volume method (Kesserwani and Sharifian, 2023) or the finite element (FE) method (Lee, 2021; Lozovskiy et al., 2017). The accuracy and computational efficiency of these models are dependent upon many factors, such as the complexity of the chosen equations and the numerical techniques employed. Despite their widespread success in flood modelling studies, the substantial computational demands of detailed hydrodynamic models lessen their suitability for real-time forecasts (Henonin et al., 2013) as well as extensive exploratory and prolonged flood simulations (Wang and Geng, 2013; Löwe et al., 2017; Jamali et al., 2020).

In recent decades, significant research effort has been directed towards 2D models that are dedicated to the creation of numerical models for unsteady shallow flows, with a particular emphasis on computational efficiency and achieving precision in depicting the flow over topographic irregularities and for phenomena such as shocks and hydraulic jumps. When dealing with hydraulic jumps or shocks, the numerical solutions for the dry front typically exhibit

* Corresponding author.
E-mail address: boyang.chen16@imperial.ac.uk (B. Chen).

https://doi.org/10.1016/j.advwatres.2025.104903
Received 12 September 2024; Received in revised form 26 December 2024; Accepted 27 January 2025
Available online 9 February 2025

unphysical oscillations and positional errors that tend to increase over time (Madsen et al., 2005; Akbari and Pirzadeh, 2023). Many successful solvers have been developed to deal with these issues such as Riemann solvers (Minatti and Faggioli, 2023; Bernetti et al., 2008; Kesserwani and Sharifian, 2023) and Godunov schemes (Liang and Marche, 2009; Skoula et al., 2006; LeFloch and Thanh, 2011). Another significant focus for numerical simulations of the shallow water equations is the modelling of wetting and drying. In particular, achieving stability and accuracy in wetting and drying scenarios remains a non-trivial challenge for numerical modellers. In many instances, stable and straightforward solutions are often imprecise, while accurate solutions tend to be computationally expensive and frequently unstable (Liang and Marche, 2009; Bunya et al., 2009; Medeiros and Hagen, 2013). Also contributing to the computational cost of solving the SWEs is the use of high-order elements. Miura and Skamarock (2013) compared computational costs for modelling the transport scheme with linear and quadratic finite-volume methods over spherical icosahedral grids on two different devices, showing an increase in the range of 1.5 to 7 in time from linear to quadratic elements.

Using high-resolution hydrodynamic computational models for large simulation domains is time-consuming and requires very large computational resources, and is therefore regarded as impractical or unfeasible, particularly when aiming for resolutions of less than 10 m (Teng et al., 2017). In recent times, interest has been gathering in using graphics processing units (GPUs) to achieve high-performance computing, also to solve SWE models. For SWEs, the performance of solution processes on CPUs has been compared with that on GPUs and speed-up factors that have been reported lie between 2 and 75 (Smith and Liang, 2013; Xia and Liang, 2016; Lacasta et al., 2015; Sharifian et al., 2023; Buwalda et al., 2023). The drawback is that, should one wish to run models initially coded for CPUs on GPUs, re-implementation is required using a GPU-specific programming framework such as CUDA or OpenCL.

Our contribution is to develop a model capable of efficiently simulating complex flood events designed to run not only on GPUs, but also, on the latest energy-efficient AI processors, inspired by the idea that both numerical discretisations and the convolutional operations of neural networks are discrete convolutions and therefore equivalent (Zhao et al., 2020; Anderson et al., 2022; Chen et al., 2024b). By expressing a system of discretised partial differential equations (PDEs) as a convolutional neural network (CNN), it is possible to solve the system and obtain the same answer (to within solver tolerances) as a typical Fortran or C++ code. No training is required for the CNN as the weights are determined by the particular discretisation scheme that is desired. This idea can be applied to finite volume, finite difference or finite element methods and to explicit, semi-implicit or fully implicit discretisations. Previously, Zhao et al. (2020) used a semi-implicit scheme to model the Navier–Stokes equations (with an explicit scheme for the velocities and an implicit description of the pressure solved with Preconditioned Conjugate Gradient and the Jacobi method) and run the neural-network–based solver on a GPU. Wang et al. (2022) used an explicit sub-iteration method to solve for the velocities and an explicit update to solve a Poisson equation for the pressure correction, running their code on Tensor Processing Units (TPUs). Chen et al. (2024b) used a predictor–corrector method for time-stepping of the velocities and a multigrid method (implemented through a U-Net architecture Olaf Ronneberger and Brox, 2015) to solve an implicit pressure equation. Writing and solving discretised systems as neural networks has several advantages including (1) the codes can run on CPUs, GPUs and AI processors with no modification to the code; (2) models are fully differentiable and suitable for performing optimisation tasks such as data assimilation; (3) these models can be easily integrated with trained neural networks that could represent sub-grid-scale models, surrogate models or physics-informed approaches; and (4) development of models is faster due to the available functionality in machine-learning libraries. The approach of Chen et al. (2024b), referred to as NN4PDEs, has been extended to model multiphase flows (Chen et al., 2024a), demonstrated on a rising bubble and a collapsing water column, for which the results compared very well with other numerical solutions and experimental results. The idea has also been applied to the multi-group neutron diffusion equation (Phillips et al., 2023b) and the Boltzmann transport equation (Phillips et al., 2023a), the latter developing a new method to implement higher-order FE discretisations referred to as ConvFEM, which are also used in this study. Higher-order finite elements do not have the same stencil for each node, which complicates the application of a convolutional FE approach. One solution would be to use multiple channels each with their own distinct filter. By contrast, ConvFEM employs a single set of filters by averaging FE stencils, simplifying its implementation on structured grids. The extension of the NN4PDEs approach to unstructured grids can be achieved with graph neural networks, as demonstrated by Li et al. (2024) for diffusion problems.

In this study, NN4PDEs is extended to solve the SWEs with two numerical schemes: (1) explicit (for both velocities and free surface height); and (2) semi-implicit (explicit time-stepping for velocities and implicit time-stepping for free surface height). Implicit schemes can be solved through multigrid methods within the NN4PDEs framework (Chen et al., 2024b), however, this is not necessary for SWEs, as their discretisations are well conditioned. FE solutions can suffer from spurious oscillations in free surface height if the velocity and free surface height variables have the basis functions of the same order. Furthermore, the advection terms in the governing equations can also lead to spurious oscillations occurring in the velocity field. To address this, an implicit Large Eddy Simulation (LES) scheme is employed to stabilise the solution. Here, the LES scheme relies on a non-linear Petrov–Galerkin method to introduce diffusion into the discretised system of equations, thereby enhancing both diagonal dominance and stability (Donéa, 2003, page 50). The methodology is effective in controlling oscillations during shocks in the free surface height and wetting and drying scenarios. For further stabilisation, the equation governing the free surface height is expressed as a correction to this variable before discretising. Similar methods are used for pressure when solving incompressible flows (Sotiropoulos and Abdallah, 1991). A predictor–corrector scheme (Ascher and Petzold, 1998; Butcher, 2016) is employed in time, which achieves second-order accuracy. Unless otherwise stated, lumped mass matrices are used in the test cases presented here to ensure simplicity and computational efficiency. To validate our method, we conduct benchmark tests on three dam break problems and we showcase the approach by simulating the Carlisle 2005 flood event, executing the code on a GPU.

The remaining sections are as follows. Section 2 describes the shallow water equations, their discretisation, their formulation in terms of discrete convolutions, Petrov–Galerkin stabilisation and the overall solution algorithm. In Section 3, we present the results obtained by applying the method to an idealised case, a laboratory experiment and the Carlisle 2005 flooding test case. Discussion and conclusions can be found in Sections 4 and 5 respectively.

2. Methodology

The conservative form of the shallow water equations is:

$$\frac{\partial \eta}{\partial t} + \frac{\partial (\eta u)}{\partial x} + \frac{\partial (\eta v)}{\partial y} = s_h, \tag{1}$$

$$\frac{\partial (\eta u)}{\partial t} + \frac{\partial}{\partial x}(\eta u^2 + \frac{1}{2} g \eta^2) + \frac{\partial (\eta u v)}{\partial y} + \hat{\sigma}_q u = -g\eta \frac{\partial B}{\partial x}, \tag{2}$$

$$\frac{\partial (\eta v)}{\partial t} + \frac{\partial}{\partial y}(\eta v^2 + \frac{1}{2} g \eta^2) + \frac{\partial (\eta u v)}{\partial x} + \hat{\sigma}_q v = -g\eta \frac{\partial B}{\partial y}, \tag{3}$$

in which η is the water depth, that is, the height of the free surface relative to the bathymetry, B. The free surface height, h, is measured from the same base level as the bathymetry, and thus the water depth and free surface height are related through $\eta = h - B$. See Fig. 1 for a

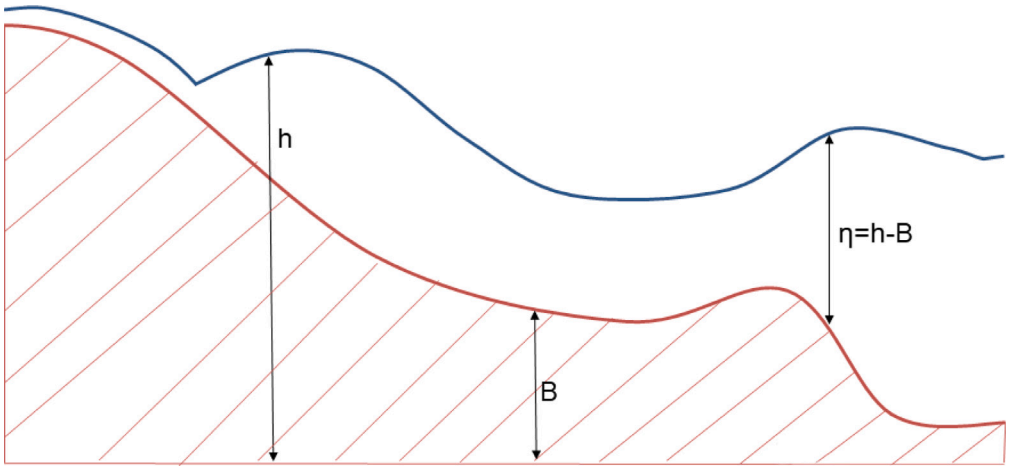


Fig. 1. A schematic diagram of the free surface variables. The height of the free surface as measured from the base level is given by h; the depth of the bathymetry as measured from the base level is given by B; and the water depth is given by $\eta = h - B$.

diagram illustrating these variables. The bed friction term, appearing in Eqs. (2) and (3), is defined through $\hat{\sigma}_q = C_f\sqrt{u^2+v^2}$. The friction function is $C_f = gn_M^2\eta^{-1/3}$, where the Manning's roughness parameter, n_M, is typically $n_M \in [0.01, 0.2]\,\mathrm{s\,m^{-1/3}}$. The velocity components in the x and y directions are u and v respectively, g represents the acceleration due to gravity and s_h is a source or sink of water which in this project is the rainfall.

2.1. Discretisation in time

2.1.1. Discretisation of the continuity equation in time

The continuity equation, Eq. (1), which governs the water depth, η, is discretised in time using a central difference approximation about $t^{n+\frac{1}{2}}$:

$$\frac{h^{n+1}-h^n}{\Delta t} + \frac{\partial(u^{n+\frac{1}{2}}\eta^{n+\frac{1}{2}})}{\partial x} + \frac{\partial(v^{n+\frac{1}{2}}\eta^{n+\frac{1}{2}})}{\partial y} = s_h^{n+1}, \tag{4}$$

in which h^n is the free surface height at time level n, Δt is the time step and the source, s_h, is treated implicitly.

2.1.2. Discretisation of the momentum equations in time

The momentum equations (Eqs. (2) and (3) written in non-conservative form and divided through by the water depth η) are also discretised in time using a central difference approximation about $t^{n+\frac{1}{2}}$ resulting in:

$$\rho_g\left(\frac{\boldsymbol{q}^{n+1}-\boldsymbol{q}^n}{\Delta t} + u^{n+\frac{1}{2}}\frac{\partial \boldsymbol{q}^{n+\frac{1}{2}}}{\partial x} + v^{n+\frac{1}{2}}\frac{\partial \boldsymbol{q}^{n+\frac{1}{2}}}{\partial y}\right) + \sigma_q^{n+1}\boldsymbol{q}^{n+1} = -\nabla h^{n+\frac{1}{2}}, \tag{5}$$

where $\rho_g = \frac{1}{g}$, $\boldsymbol{q}^n = (u^n\ v^n)^T$ denotes the velocity and the bed friction term is expressed as follows:

$$\sigma_q^{n+1} = \frac{\hat{\sigma}_q^{n+1}}{g\eta^{n+1}} = \frac{gn_M^2(\eta^{n+1})^{-1/3}\sqrt{(u^{n+1})^2+(v^{n+1})^2}}{g\eta^{n+1}} = \frac{n_M^2\sqrt{(u^{n+1})^2+(v^{n+1})^2}}{(\eta^{n+1})^{4/3}}. \tag{6}$$

In practice, the following expression is used to put a limit on how small η can be in the denominator to avoid dividing by a number close to zero:

$$\sigma_q^{n+1} = \frac{n_M^2\sqrt{(u^{n+1})^2+(v^{n+1})^2}}{(\max\{\epsilon_\eta, \eta^{n+1}\})^{4/3}}, \tag{7}$$

where ϵ_η is a small number (for our purposes, it is taken as $\epsilon_\eta = 10^{-4}$). The source term from the continuity equation, s_h, would appear in the non-conservative form of the momentum equations, Eq. (5). However, if the momentum equations, themselves, had a source, which had the same velocity as the fluid, the sources would cancel. This is assumed to be the case here.

2.2. Discretisation of the governing equations in space using convolutional layers

2.2.1. Numerical discretisations as discrete convolutions

The matrix vector operations that occur within numerical discretisations can be thought of as discrete convolutions and can therefore be expressed as convolutional layers of a neural network. Such operations can be evaluated by functions within machine learning libraries. A convolutional filter or kernel, $\mathbf{w}$, operating on a 2D tensor $\boldsymbol{\Psi}^n$ can be written as:

$$\boldsymbol{f}(\boldsymbol{\Psi}^n;\mathbf{w})\Big|_{i,j} = \sum_{ii=-\ell}^{\ell}\sum_{jj=-\ell}^{\ell} \mathrm{w}_{ii,jj}\,\boldsymbol{\Psi}^n_{i+ii,j+jj}, \tag{8}$$

where, the filter $\mathbf{w}$, consisting of weights $\mathrm{w}_{ii,jj}$, has dimensions of $2\ell+1$ by $2\ell+1$. The tensor $\boldsymbol{\Psi}^n$ has a of dimension $N_x+2\ell$ by $N_y+2\ell$ where ℓ represents the padding thickness. As well as describing the convolution of a filter ($\mathbf{w}$) with an image whose pixel values are represented by $\boldsymbol{\Psi}^n$, Eq. (8) could also represent a discretised differential operator with stencil $\mathbf{w}$ acting on a field represented by $\boldsymbol{\Psi}^n$. In the case of a FE discretisation, the padding ℓ depends on the order of the elements. For linear, quadratic and cubic elements, ℓ is 1, 2 and 3, respectively. From this point onwards, the function $\boldsymbol{f}$ is used to represent the convolution of a filter with a tensor. Fig. 2 illustrates the equivalence of a finite element discretisation and a convolutional layer.

2.2.2. First-order derivatives

The weights of the filters are determined by integrating the products of the basis functions (N_a) and their derivatives in the standard way, for example, $\int_V N_a\frac{\partial N_b}{\partial x}dxdy$. By replacing the FE nodes a and b with the corresponding node or tensor indices (i,j) and (i',j') respectively, the tensor indices (i,j) of this FEM matrix can be written as:

$$\int_V N_{i,j}\frac{\partial N_{i',j'}}{\partial x}dxdy = \mathrm{w}_{\mathrm{x}(ii,jj)} \quad \text{where } ii = i-i',\ jj = j-j'. \tag{9}$$

Due to compact support, these components are only non-zero for indices $ii \in \{-1,0,1\}$, $jj \in \{-1,0,1\}$ (for linear elements) and the fact that every element is of quadrilateral shape and of the same size (i.e., regular structured quadrilateral elements), means the stencil and therefore these components are the same for each node. This is the reason why they can be represented by the convolutional filter $\mathbf{w}_\mathrm{y}$ of dimension $2\ell+1$ by $2\ell+1$.

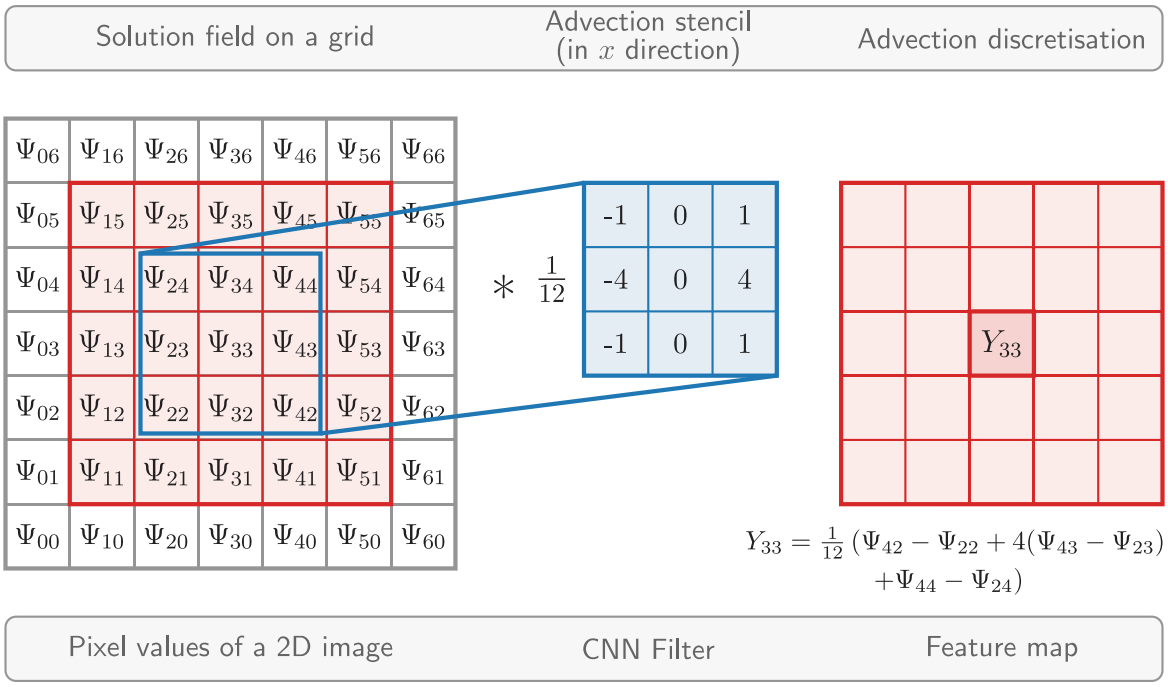


Fig. 2. A diagram illustrating the equivalence between numerical discretisations and convolutional operations in neural networks. On the left side (depicted in red) is a solution field Ψ_{ij} on a 5 by 5 grid ($i, j \in 1, 2, \ldots, 5$). Surrounding the solution field is a set of elements, often referred to as 'ghost cells' through which boundary conditions can be implemented. A 3 by 3 stencil for advection in the x direction arising from a linear finite element discretisation (with $\Delta x = 1 = \Delta y$) is indicated in blue. The stencil is applied to the central group of elements corresponding to $i, j \in 2, 3, 4$. The value of the discretised advection operator acting on the elements is put in the central element on the right (Y_{33}). Equivalently, the red area on the left could be thought of as an image with pixel values represented by Ψ_{ij} surrounded by padding. Weights of a convolutional layer are seen in blue. The result of the convolution between the central part of the image and weights is placed in the central cell of the feature map on the right section.

If Eqs. (9) are evaluated for the discretised advection operator in the x direction, a finite element scheme can be realised using a convolutional layer with a filter possessing weights $\mathbf{w_x}$ defined by:

$$\mathbf{w_x} = \begin{pmatrix} w_{x(-1,-1)} & w_{x(-1,0)} & w_{x(-1,1)} \\ w_{x(0,-1)} & w_{x(0,0)} & w_{x(0,1)} \\ w_{x(1,-1)} & w_{x(1,0)} & w_{x(1,1)} \end{pmatrix} = \frac{\Delta y}{12}\begin{pmatrix} -1 & 0 & 1 \\ -4 & 0 & 4 \\ -1 & 0 & 1 \end{pmatrix}, \tag{10}$$

where Δy is the grid spacing between the nodes in the y direction. Similarly, the weights corresponding to the advection operator in the y direction discretised by linear finite elements are

$$\mathbf{w_y} = \frac{\Delta x}{12}\begin{pmatrix} 1 & 4 & 1 \\ 0 & 0 & 0 \\ -1 & -4 & -1 \end{pmatrix}. \tag{11}$$

The consistent, lumped mass and lumped mass inverse filters for linear FEM are:

$$\mathbf{w_m} = \frac{m_l}{36}\begin{pmatrix} 1 & 4 & 1 \\ 4 & 16 & 4 \\ 1 & 4 & 1 \end{pmatrix}, \ \mathbf{w_{ml}} = m_l\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ \mathbf{w_{ml}^{-1}} = \frac{1}{m_l}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{12}$$

where $m_l = \Delta x \Delta y$. A possible drawback of using higher order FEs (quadratic and above) is that the stencil would be different at each node, which substantially increases the complexity of the NN4PDEs implementation. To overcome this difficulty, a new FEM approach has been developed called the Convolutional Finite Element Method (ConvFEM), which has the same stencil for each node. The basic idea in forming the ConvFEM filters is to add discretisation stencils and find their average (Phillips et al., 2023a). A number of ConvFEM discretisations are applied in this paper: 3 × 3 linear ConvFEM/FEM filters (the linear FEM filters are identical to the linear ConvFEM filters); 5 × 5 quadratic ConvFEM filters; 7 × 7 cubic ConvFEM filters; and 9 × 9 quartic ConvFEM filters.

2.2.3. Second-order derivatives

Second order derivatives appear through diffusive stabilisation terms added to both the equations for free surface height and momentum. The magnitude of the diffusion is determined (see Eq. (A.6) in Appendix A) from a non-linear Petrov–Galerkin method. Supposing the solution field is Ψ^n with diffusion a coefficient k^n, the diffusion equation at t^n can be written as (see the Eq. (13) in Box I):

in which we define $\boldsymbol{f}^{\nabla^2}$, where $\mathbf{w}_{\nabla^2}$ is the filter associated with the discretised Laplacian ∇^2. Forming the diffusion as three Laplacians has the benefit of not having to increase the size of the stencil (or filter).

2.2.4. Discretisation of the continuity equation in space

After applying the product rule to the first order derivatives in the continuity equation, Eq. (4), the FE spatial discretisation (written as convolutional filters that are derived in Section 2.2.1) is applied. This produces a conservative (in η^n) discretised system of equations. For stabilisation, a diffusion operator (analytically $\nabla \cdot (k\nabla\eta)$) is added to the right-hand side of Eq. (4) (see Appendix A for details of its discretisation). The resulting equation can be written as follows:

$$\begin{aligned} \boldsymbol{r}_{\boldsymbol{h}}^{n+\frac{1}{2}} &= \boldsymbol{f}\left(\frac{\boldsymbol{h}^{n+1} - \boldsymbol{h}^n}{\Delta t}; \mathbf{w_{m/ml}}\right) - \boldsymbol{f}\left(\boldsymbol{s}_{\boldsymbol{h}}^{n+1}; \mathbf{w_{ml}}\right) \\ &+ \boldsymbol{u}^{n+\frac{1}{2}} \odot \boldsymbol{f}\left(\boldsymbol{\eta}^{n+\frac{1}{2}}; \mathbf{w_x}\right) + \boldsymbol{v}^{n+\frac{1}{2}} \odot \boldsymbol{f}\left(\boldsymbol{\eta}^{n+\frac{1}{2}}; \mathbf{w_y}\right) \\ &+ \boldsymbol{\eta}^{n+\frac{1}{2}} \odot \left(\boldsymbol{f}\left(\boldsymbol{u}^{n+\frac{1}{2}}; \mathbf{w_x}\right) + \boldsymbol{f}\left(\boldsymbol{v}^{n+\frac{1}{2}}; \mathbf{w_y}\right)\right) \\ &+ \boldsymbol{f}^{\nabla^2}\left(\boldsymbol{\eta}^{n+\frac{1}{2}}, \boldsymbol{k}_{\boldsymbol{\eta}}^{n+\frac{1}{2}}; \mathbf{w}_{\nabla^2}^{\mathbf{PG}}\right), \end{aligned} \tag{14}$$

where $\boldsymbol{r}_h$ is the residual associated with the free surface height, $\boldsymbol{h}^n$, $\boldsymbol{\eta}^n$, $\boldsymbol{u}^n$, $\boldsymbol{v}^n$ and $\boldsymbol{s}_{\boldsymbol{h}}^n$ are tensors containing the nodal values at time level n of the free surface height, water depth, x and y components of velocity and source respectively. The entry-wise multiplication of tensors is represented by the Hadamard product, which is denoted by $\odot$. The function $\boldsymbol{f}(\cdot;\cdot)$ represents a convolution of the first argument with the kernel or filter given in the second argument, see Section 2.2.1.

$$
\begin{aligned}
-\nabla\cdot(k^n\nabla\Psi^n) &= -\frac{1}{2}\left(\nabla^2(k^n\Psi^n)+k^n\nabla^2\Psi^n-\Psi^n\nabla^2k^n\right) && (Discretised in time)\\
&\sim \frac{1}{2}\left(\boldsymbol{f}(\boldsymbol{k}^n\odot\boldsymbol{\Psi}^n;\mathbf{w}_{\nabla^2})+\boldsymbol{k}^n\odot\boldsymbol{f}(\boldsymbol{\Psi}^n;\mathbf{w}_{\nabla^2})-\boldsymbol{\Psi}^n\odot\boldsymbol{f}(\boldsymbol{k}^n;\mathbf{w}_{\nabla^2})\right)=:\boldsymbol{f}^{\nabla^2}(\boldsymbol{\Psi}^n,\boldsymbol{k}^n;\mathbf{w}_{\nabla^2}) && (Discretised in space and time)
\end{aligned}
\tag{13}
$$

Box I.

The terms $\mathbf{w}_\mathbf{x}$ and $\mathbf{w}_\mathbf{y}$ represent the filters (whose entries make up the weights) of the neural network that correspond to the discretisations of the first derivative with respect to x and y. Similarly, the terms $\mathbf{w}_\mathbf{m}$, $\mathbf{w}_\mathbf{ml}$, $\mathbf{w}_\mathbf{ml}^{-1}$, and $\mathbf{w}_{\nabla^2}^{\mathbf{PG}}$ represent filters corresponding to coefficients associated with the consistent mass matrix, lumped mass matrix, its inverse and the Petrov–Galerkin diffusion applied to stabilise the free surface height. The notation $\mathbf{w}_\mathbf{m/ml}$ indicates that one can use either the consistent mass filter or lumped mass filter, with the latter reducing the accuracy in the approximation in time. See Section 2.2.1 for derivations of the convolutional filters. Petrov–Galerkin stabilisation (Donéa, 2003) (see Appendix A) is used here to control oscillations in the free surface height that often occur with central-difference or FEM-based discretisations of the advection term. The stabilisation introduces a diffusion term (with diffusion coefficients $\boldsymbol{k}_\eta^{n+\frac{1}{2}}$) and provides robustness for demanding problems, such as those which include wetting and drying. It can be convenient to use a linear FEM filter for $\mathbf{w}_{\nabla^2}^{\mathbf{PG}}$ rather than a higher order ConvFEM filter, as the latter has a greater tendency to introduce oscillations due to its use of high order FEM polynomials. For idealised problems, the stabilisation term $\mathbf{w}_{\nabla^2}^{\mathbf{PG}}$ was discretised with the same order of ConvFEM filters as the other spatial derivatives (that is $\mathbf{w}_{\nabla^2}^{\mathbf{PG}} = \mathbf{w}_{\nabla^2}$), as the form of this term was found to have little influence on the results. However, for the more demanding test cases, the stabilisation term was discretised with linear elements irrespective of the order of discretisation used for the other terms.

2.2.5. Discretisation of the momentum equations in space

The momentum equations include a term which depends on the bed friction, Eq. (7). To discretise this term, we first need to distribute the value of $\boldsymbol{\eta}^{n+1}$ over the nodes using

$$\hat{\boldsymbol{\eta}}^{n+1}=\boldsymbol{f}\left(\boldsymbol{\eta}^{n+1};\mathbf{w}_\mathbf{ml}{}^{-1}\mathbf{w}_\mathbf{m}\right), \tag{15}$$

where the filter $\mathbf{w}_\mathbf{ml}{}^{-1}\mathbf{w}_\mathbf{m}$ represents the product of two filters associated with the inverse lumped mass matrix and the consistent mass matrix respectively. Distributing the values of water depth over the nodes in this way enables the water to move from the wet to dry areas without being inhibited by excessive bottom drag. On discretising Eq. (7) using the expression in Eq. (15), we obtain

$$\boldsymbol{\sigma}_\boldsymbol{q}^{n+1}=(\boldsymbol{n}_\boldsymbol{M})^2\odot\boldsymbol{c}^{n+1}\oslash(\max\{\mathbf{1}\epsilon_\eta,(1-\alpha_\eta)\boldsymbol{\eta}^{n+1}+\alpha_\eta\hat{\boldsymbol{\eta}}^{n+1}\})^{4/3}, \tag{16}$$

in which the fluid speed tensor $\boldsymbol{c}^{n+1}$ is such that $c_{i,j,k}^{n+1}=\sqrt{(u_{i,j,k}^{n+1})^2+(v_{i,j,k}^{n+1})^2}$. The tensor of nodal values of the Manning coefficient, $\boldsymbol{n}_\boldsymbol{M}$, is assumed to be constant and $\alpha_\eta = 0.1$ is a heuristically obtained parameter.

When discretising the momentum equations (Eq. (5)), we adopt a Petrov–Galerkin stabilisation similar to that used for the free surface height. However, for the momentum equations, the stabilisation terms have the same order elements as the other terms in the equations, hence the weights are denoted by $\mathbf{w}_{\nabla^2}$ and not $\mathbf{w}_{\nabla^2}^{\mathbf{PG}}$. Mass lumping is used for the advection, time and absorption terms. Applying the FE spatial discretisations through convolutional filters leads to:

$$
\begin{aligned}
\boldsymbol{r}_\boldsymbol{u}^{n+\frac{1}{2}} &= \boldsymbol{f}\left(\rho_g\left(\frac{\boldsymbol{u}^{n+1}-\boldsymbol{u}^n}{\Delta t}\right);\mathbf{w}_\mathbf{m/ml}\right)+\boldsymbol{f}\left(\boldsymbol{\sigma}_\boldsymbol{q}^{n+1}\odot\boldsymbol{u}^{n+1};\mathbf{w}_\mathbf{ml}\right)\\
&+\rho_g\left(\boldsymbol{u}^{n+\frac{1}{2}}\odot\boldsymbol{f}(\boldsymbol{u}^{n+\frac{1}{2}};\mathbf{w}_\mathbf{x})+\boldsymbol{v}^{n+\frac{1}{2}}\odot\boldsymbol{f}(\boldsymbol{u}^{n+\frac{1}{2}};\mathbf{w}_\mathbf{y})\right)\\
&+\boldsymbol{f}^{\nabla^2}(\boldsymbol{u}^{n+\frac{1}{2}},\boldsymbol{k}_u^{n+\frac{1}{2}};\mathbf{w}_{\nabla^2})+\boldsymbol{f}(\boldsymbol{h}^{n+\frac{1}{2}};\mathbf{w}_\mathbf{x}),
\end{aligned}
\tag{17}
$$

$$
\begin{aligned}
\boldsymbol{r}_\boldsymbol{v}^{n+\frac{1}{2}} &= \boldsymbol{f}\left(\rho_g\left(\frac{\boldsymbol{v}^{n+1}-\boldsymbol{v}^n}{\Delta t}\right);\mathbf{w}_\mathbf{m/ml}\right)+\boldsymbol{f}\left(\boldsymbol{\sigma}_\boldsymbol{q}^{n+1}\odot\boldsymbol{v}^{n+1};\mathbf{w}_\mathbf{ml}\right)\\
&+\rho_g\left(\boldsymbol{u}^{n+\frac{1}{2}}\odot\boldsymbol{f}(\boldsymbol{v}^{n+\frac{1}{2}};\mathbf{w}_\mathbf{x})+\boldsymbol{v}^{n+\frac{1}{2}}\odot\boldsymbol{f}(\boldsymbol{v}^{n+\frac{1}{2}};\mathbf{w}_\mathbf{y})\right)\\
&+\boldsymbol{f}^{\nabla^2}(\boldsymbol{v}^{n+\frac{1}{2}},\boldsymbol{k}_v^{n+\frac{1}{2}};\mathbf{w}_{\nabla^2})+\boldsymbol{f}(\boldsymbol{h}^{n+\frac{1}{2}};\mathbf{w}_\mathbf{y}),
\end{aligned}
\tag{18}
$$

where $\boldsymbol{k}_u$ and $\boldsymbol{k}_v$ are diffusion coefficients in x and y directions, respectively.

The non-conservative form of the momentum equations is solved here because: (1) this avoids having to divide through by η after solving for the conservative variables $(\eta u)^n$ and $(\eta v)^n$ in order to obtain the velocities u^n and v^n; (2) momentum is no longer conserved when introducing bed friction, which is often dominant; (3) of the desire to apply numerical diffusion (from the Petrov–Galerkin method) directly to the velocities in the discretised form of Eq. (5); (4) it is relatively straightforward to form numerical methods in which a motionless steady state condition remains motionless — see the "C-property" (Vanzo et al., 2016); (5) of the simplicity of the associated implementation which can be important for computational efficiency.

2.2.6. Application of boundary conditions

The boundary conditions are applied through halo nodes (also known as ghost or virtual cells/nodes) that are situated at the perimeter of the domain, using padding (see Fig. 2). This means that the discretisation scheme is the same for nodes next to the boundary as it is for nodes within the domain, which simplifies the implementation of the method. These halos have a width of one node for linear elements, two nodes for quadratic ConvFEM elements, three nodes for cubic ConvFEM elements and so on. Dirichlet boundary conditions are applied at the halo nodes by setting their value equal to the desired Dirichlet boundary condition. When applying a specified normal derivative, the value at the halo nodes is calculated by extrapolating from the nearest node within the domain using the specified gradient. Most commonly, a zero derivative is applied (such as when applying a symmetry boundary condition, see Fig. 3) in which case, this simplifies to copying the nearest nodal value within the domain to the halo nodes.

Thus, for incoming flows (inlets), we specify the values of $\eta_{i,j,k}^n$, $\eta_{i,j,k}^{n+1}$ at the halo nodes based on the desired Dirichlet boundary conditions and for all other boundaries (including outlets) we apply the zero derivative boundary condition to $\eta_{i,j,k}^n$, $\eta_{i,j,k}^{n+1}$. For incoming velocities, the Dirichlet velocity components are specified at the halo nodes and for outgoing flows, a zero normal-derivative boundary condition is applied by copying the nodal values at the boundary to the halo nodes for the velocity component normal to the boundary.

2.3. Solution method for the shallow water equations

This section describes the overall solution method for solving the shallow water equations. Although these equations are highly coupled, for computational efficiency, a segregated solution method is applied, which first solves for the velocity (stage 1) and second the free surface height (stage 2). A correction based on the continuity equation is then applied to the velocity (stage 3). The segregated approach fits within a predictor–corrector method for time, so each stage is solved once in the predictor step and as many times as required in the corrector step(s). The effect of this procedure is to minimise the residuals defined in Eqs. (14), (17) and (18).

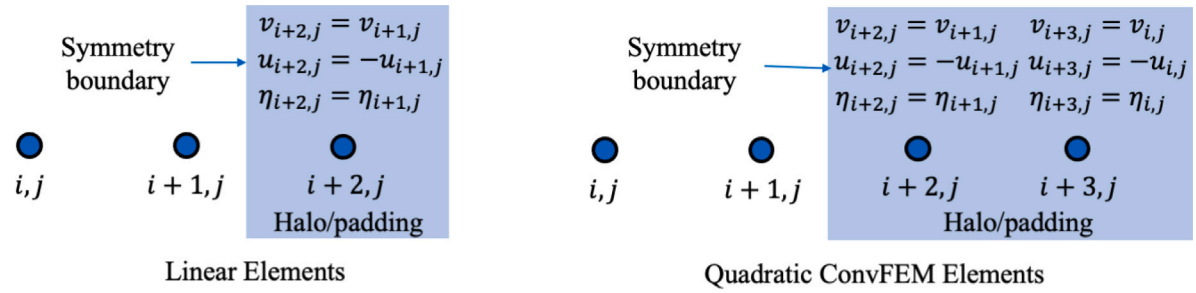


Fig. 3. Diagram showing how the values within the halo regions (shaded blue) are calculated for the linear ConvFEM and quadratic ConvFEM approaches. This example is of a symmetry boundary on the right of the domain. Similar boundary conditions are applied for symmetry boundaries on the left of the domain, and at the top and bottom of the domain. For outlet boundaries where one might use zero normal-to-the-boundary derivative boundary conditions for all the variables, one simply uses the symmetry boundary conditions (as shown in these figures) but without changing the sign of the x velocity component in the halo/padding regions.

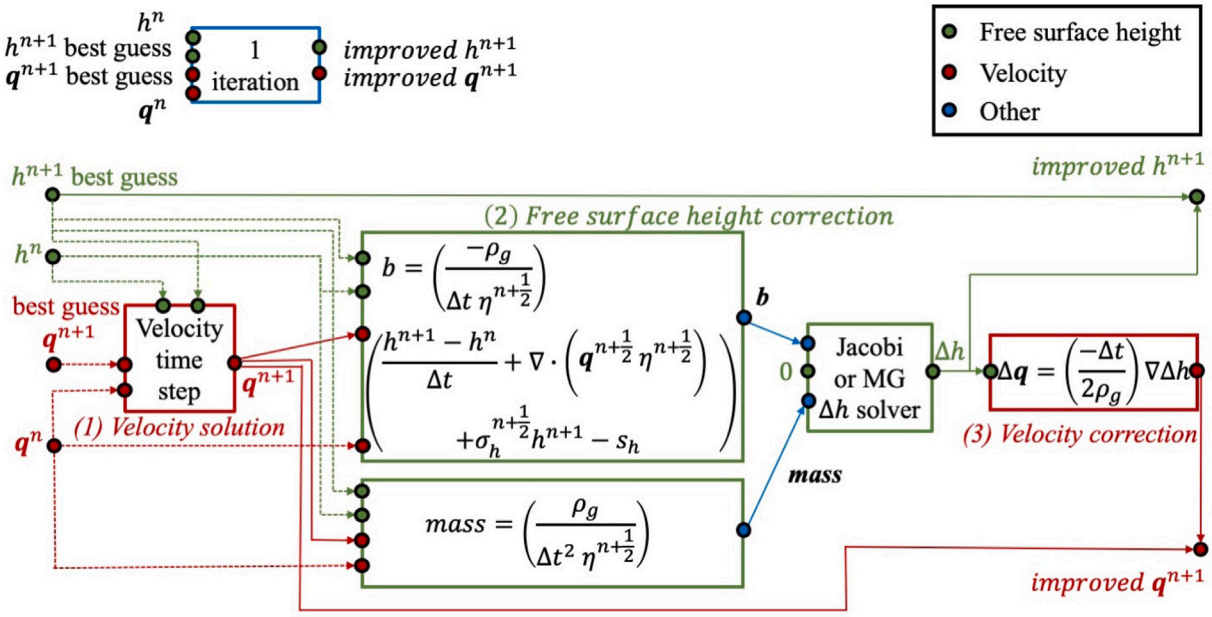


Fig. 4. Top left: diagram of an iteration within a time step of the neural network SWE solver, highlighting the inputs and outputs (of the neural network). The first iteration corresponds to the predictor step, the second iteration to the corrector step. Top right: key of the colours used to represent the inputs and outputs of the neural network. Bottom: This diagram expands the diagram in the top left, indicating the three stages in the solution process: (1) solving for velocity; (2) solving for free surface height correction; (3) the velocity correction derived from the continuity equation. (The variables in this diagram have been discretised in time only, for readability.)

Figs. 4 and 5 show schematic diagrams how the neural network solvers are incorporated in the time integration. The top left diagram in Fig. 4 summarises an iteration within a time step, where the inputs are $\boldsymbol{h}^n$, $\boldsymbol{q}^n$ and the best guesses for $\boldsymbol{h}^{n+1}$ and $\boldsymbol{q}^{n+1}$, and the outputs are improved approximations of $\boldsymbol{h}^{n+1}$ and $\boldsymbol{q}^{n+1}$. In the predictor step, best guesses for $\boldsymbol{h}^{n+1}$ and $\boldsymbol{q}^{n+1}$ will be $\boldsymbol{h}^n$ and $\boldsymbol{q}^n$. In the corrector step, best guesses for these variables will be those calculated at the previous iteration in stage 2 (for free surface height) and stage 3 (for velocities). The main diagram in Fig. 4 shows how stages 1, 2 and 3 are solved within one iteration. The equations in this figure are shown as discretised in time but not space, for readability. Fig. 5 describes how two iterations can be linked together to form a time step made up of a predictor and one corrector step. In the following sections, we describe the predictor–corrector method and summarise the equations solved in stages 1, 2 and 3. For a more detailed derivation, see Appendix B.

2.3.1. Predictor–corrector method

Having discretised the governing equations in time with a central difference approximation about $t^{n+\frac{1}{2}}$, in the next sections (Section 2.2.1 and onwards), we go on to outline the spatial discretisation. The result is a coupled system of discretised equations, the time dependence of which will be solved with a predictor–corrector method that is second-order accurate. For more information, see Chapter 5 in Ascher and Petzold (1998) or Section 244 in Butcher (2016). We illustrate the predictor–corrector approach, here, applied to a single ordinary differential equation (ODE):

$$\frac{dy}{dt} = f(y) \quad \xrightarrow{\text{discretise}} \quad \frac{y^{n+1}-y^n}{\Delta t} = f(y^{n+\frac{1}{2}})$$
$$\text{where } y^{n+\frac{1}{2}} := y(t^{n+\frac{1}{2}}), \tag{19}$$

where y is a function of t, f represents the right-hand side of the ODE, Δt is the time-step size and the time levels are indicated as superscripts. First, a prediction is made for y^{n+1} (often written as $y^{n+1,*}$) by using the approximation $y^{n+\frac{1}{2}} \approx y^n$ (which results in a Forward Euler approximation for $y^{n+1,*}$). For the single corrector step used in this illustration, we approximate $y^{n+\frac{1}{2}}$ using the trapezoidal rule $\frac{1}{2}\left(y^{n+1,*}+y^n\right)$. This can be written as follows:

$$\text{Predictor step:} \quad \frac{y^{n+1,*}-y^n}{\Delta t} = f(y^n) \qquad \textit{Forward Euler}, \tag{20}$$

$$\text{Corrector step:} \quad \frac{y^{n+1}-y^n}{\Delta t} = f\left(\frac{1}{2}\left(y^{n+1,*}+y^n\right)\right) \qquad \textit{Trapezoidal rule}. \tag{21}$$

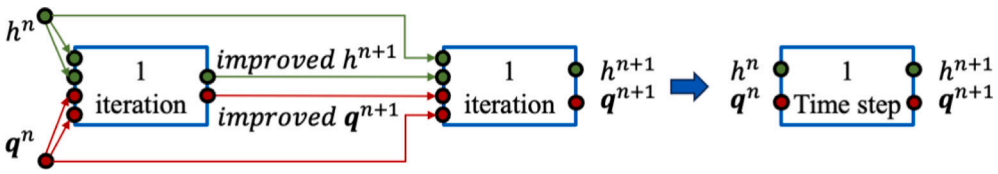


Fig. 5. Left: schematic diagram of the first iteration of a time step; right: how this and a second iteration can be brought together to form one time step. Two iterations are used for illustrative purposes; more iterations can be used as required.

Thus, Eq. (20) is solved for $y^{n+1,*}$, and Eq. (21) is solved for y^{n+1}. If the function f is evaluated with the most recent approximation of y, this is known as a PECE method (Predict–Evaluate–Correct–Evaluate). More iterations can be applied to achieve higher accuracy and greater stability, which is denoted as PE(CE)^M, where M indicates the number of iterations to be applied. Here, $M = 1$ was found to be sufficient.

2.3.2. Stage 1: Velocity solution

To solve for velocity, we wish to reduce the residuals written in Eqs. (17) and (18) to zero. As the future velocity and the bed friction are the dominant terms in the residuals, we treat these terms implicitly by forming the equations below:

$$\boldsymbol{u}^{n+1} \leftarrow -\boldsymbol{f}(\boldsymbol{r}_{\boldsymbol{u}}^{n+\frac{1}{2}}; \mathbf{w}_{ml}^{-1}) \oslash \left(\frac{\rho_g}{\Delta t}\mathbf{1} + \boldsymbol{\sigma}_{\boldsymbol{q}}^{n+1} \right) + \boldsymbol{u}^{n+1}, \tag{22}$$

$$\boldsymbol{v}^{n+1} \leftarrow -\boldsymbol{f}(\boldsymbol{r}_{\boldsymbol{v}}^{n+\frac{1}{2}}; \mathbf{w}_{ml}^{-1}) \oslash \left(\frac{\rho_g}{\Delta t}\mathbf{1} + \boldsymbol{\sigma}_{\boldsymbol{q}}^{n+1} \right) + \boldsymbol{v}^{n+1}, \tag{23}$$

where $\mathbf{1}$ is a tensor, in which each entry is unity $\left(\text{that is } \mathbf{1}\,|_{i,j,k} = 1\right)$. Hadamard entrywise division of tensors of the same size is denoted by $\oslash$ and defined as follows,

$$\hat{\boldsymbol{c}} = \hat{\boldsymbol{a}} \oslash \hat{\boldsymbol{b}} \quad \Longleftrightarrow \quad \hat{c}_{i,j,k} = \frac{\hat{a}_{i,j,k}}{\hat{b}_{i,j,k}}.$$

Having solved for $\boldsymbol{u}^{n+1}$ and $\boldsymbol{v}^{n+1}$, the velocities $\boldsymbol{u}^{n+\frac{1}{2}}$ and $\boldsymbol{v}^{n+\frac{1}{2}}$ are updated. On the right-hand side of Eqs. (22) and (23), the variables $\boldsymbol{u}^{n+1}$ and $\boldsymbol{v}^{n+1}$ refer to the most up-to-date approximation that is available — for the predictor step this will be $\boldsymbol{u}^n$ and $\boldsymbol{v}^n$. Writing Eqs. (22) and (23) in a residual form, enables one to use either the consistent mass matrix (defined by the convolutional filter $\mathbf{w}_m$) or the lumped mass matrix ($\mathbf{w}_{ml}$) for the time term within the residual. As, if the iteration process converges, the residuals will have been forced to zero whichever mass matrix is used. Choosing a diagonal lumped mass matrix has the benefit of being easy to invert, making the iterative process efficient.

2.3.3. Stage 2: Free surface height correction

The free surface correction equation is derived by subtracting out the velocity equation (Eq. (5)) with the previous free surface height at time level n and the new free surface height at time level $n + 1$ that leads to the satisfaction of the continuity equation (Eq. (4)). The divergence of the resulting momentum equation is formed to produce the free surface height correction equation similar to Pavlidis et al. (2016), see Appendix B for details. In continuous form, the resulting iterative correction Δh to the free surface height can be determined by solving:

$$\left(-\nabla\cdot\nabla + \frac{\beta_h \rho_g}{(\Delta t)^2 \eta^{n+\frac{1}{2}}} \right) \Delta h = \frac{\beta_h \rho_g}{\Delta t \eta^{n+\frac{1}{2}}} \left(\frac{h^{n+1} - h^n}{\Delta t} + \nabla\cdot(\boldsymbol{q}^{n+\frac{1}{2}} \eta^{n+\frac{1}{2}}) - s_h^{n+1} \right), \tag{24}$$

in which all the variables on the right-hand side of this expression use the best estimate of the variables h^{n+1}, $\boldsymbol{q}^{n+1}$. Here a parameter $\beta_h = 4$ has been included. If large time steps are being used and the non-linear convergence is struggling, β_h can be reduced. $\beta_h = 1$ corresponds to a more stable, but less accurate, backward Euler time discretisation rather than the predictor–corrector method used here. The discretised form of Eq. (24) is

$$\boldsymbol{f}(\boldsymbol{\Delta h}; \mathbf{w}_{\nabla^2}) + \frac{\beta_h \rho_g}{(\Delta t)^2} \left(\mathbf{1} \oslash \max\{\epsilon_\eta \mathbf{1}, \boldsymbol{\eta}^{n+\frac{1}{2}}\} \right) \odot \boldsymbol{f}(\boldsymbol{\Delta h}; \mathbf{w}_{\mathbf{ml}}) = -\frac{\beta_h \rho_g}{\Delta t} \boldsymbol{r}_{\boldsymbol{h}}^{n+\frac{1}{2}} \oslash \max\{\epsilon_\eta \mathbf{1}, \boldsymbol{\eta}^{n+\frac{1}{2}}\}, \tag{25}$$

in which $\boldsymbol{\eta}^{n+\frac{1}{2}}$ has been replaced by $\max\{\epsilon_\eta \mathbf{1}, \boldsymbol{\eta}^{n+\frac{1}{2}}\}$ in the above to aid stability for wetting and drying occurring at small water depths. For idealised problems, $\epsilon_\eta = 0$, and for non-idealised problems, $\epsilon_\eta = 0.0001$ m (0.1 mm). Eq. (25) is solved for $\boldsymbol{\Delta h}$ using a Jacobi iteration, which is sufficient for Courant numbers (based on the wave speed) of less than $\mathcal{O}(10)$, say. For larger Courant numbers, a multigrid method could also be used to calculate the free surface height correction tensor. See Phillips et al. (2023b) for a description of how the multigrid method can be implemented through neural networks with the U-Net architecture (Olaf Ronneberger and Brox, 2015). Having solved for $\boldsymbol{\Delta h}$, the free surface height is updated according to

$$\boldsymbol{h}^{n+1} \leftarrow \boldsymbol{h}^{n+1} + \boldsymbol{\Delta h}\,.$$

2.3.4. Stage 3: Velocity correction

The velocity correction $\Delta \boldsymbol{q}$ is then calculated, which, in continuous form, is given by

$$\Delta \boldsymbol{q} = -\frac{\Delta t}{2\rho_g} \nabla \Delta h\,. \tag{26}$$

In discretised form, this becomes

$$\boldsymbol{\Delta u} = -\frac{\Delta t}{2\rho_g} \left(\boldsymbol{f}\left(\boldsymbol{f}(\boldsymbol{\Delta h}; \mathbf{w}_{\mathbf{x}}); \mathbf{w}_{\mathbf{ml}}^{-1} \right) \right), \quad \boldsymbol{\Delta v} = -\frac{\Delta t}{2\rho_g} \left(\boldsymbol{f}\left(\boldsymbol{f}(\boldsymbol{\Delta h}; \mathbf{w}_{\mathbf{y}}); \mathbf{w}_{\mathbf{ml}}^{-1} \right) \right). \tag{27}$$

Once $\boldsymbol{\Delta u}$ and $\boldsymbol{\Delta v}$ are obtained, the tensors representing the discrete velocity components can be updated:

$$\boldsymbol{u}^{n+1} \leftarrow \boldsymbol{u}^{n+1} + \boldsymbol{\Delta u}, \quad \boldsymbol{v}^{n+1} \leftarrow \boldsymbol{v}^{n+1} + \boldsymbol{\Delta v}. \tag{28}$$

3. Results

In this section, the NN4PDEs approach is validated against a number of benchmark problems: a circular dam break, a dam break over a triangular-shaped hill and the propagation of flood water across a flood-plain. In addition, the 2005 flooding of Carlisle city centre is simulated using NN4PDEs, and the results are validated against observation data and other numerical results. All the models presented here were run on two types of single GPU: NVIDIA A10 Tensor Core GPU and NVIDIA RTX A100. The details of the simulations and the GPU the simulations are run on are summarised in Table 1.

3.1. Circular-shaped dam break problem

A circular-shaped dam break test (Fernández-Pato et al., 2018) is considered in this section in order to test the accuracy of our NN4PDEs approach for solving the shallow water equations. An analytical solution exists for this problem obtained from the linearised form of the SWEs, with which we compare the performance of a number of computational models generated with NN4PDEs, including varying spatial resolution, varying the Courant number, varying the order of FE discretisation, semi-implicit and explicit solvers, and consistent mass and mass lumping. The dimensions of the computational domain are

Table 1
Summary of benchmark problems solved by NN4PDEs. L, linear ConvFEM; Q, quadratic ConvFEM; C, cubic ConvFEM; Qt, quartic ConvFEM.

Case	Description	GPU	Dimension (m)	Source	Discretisation	Grid spacing (m)	Figures
1	Circular dam break (Fernández-Pato et al., 2018)	A10	200 × 200	No	L, Q, C, Qt	0.4–6.25	Figs. 6–12
2	Triangular hill dam break (Liang and Marche, 2009)	A10	38 × 9	No	L	0.04–0.16	Figs. 13, 14
3	Floodplain flooding (Shaw et al., 2021; Néelz and Pender, 2013)	A10	2000 × 2000	Yes	L, Q	5.0	Figs. 16, 17
4	Carlisle flooding (Kesserwani and Sharifian, 2023; Morales-Hernández et al., 2021)	A100	4700 × 3100	Yes	L, Q	5.0	Figs. 18–22

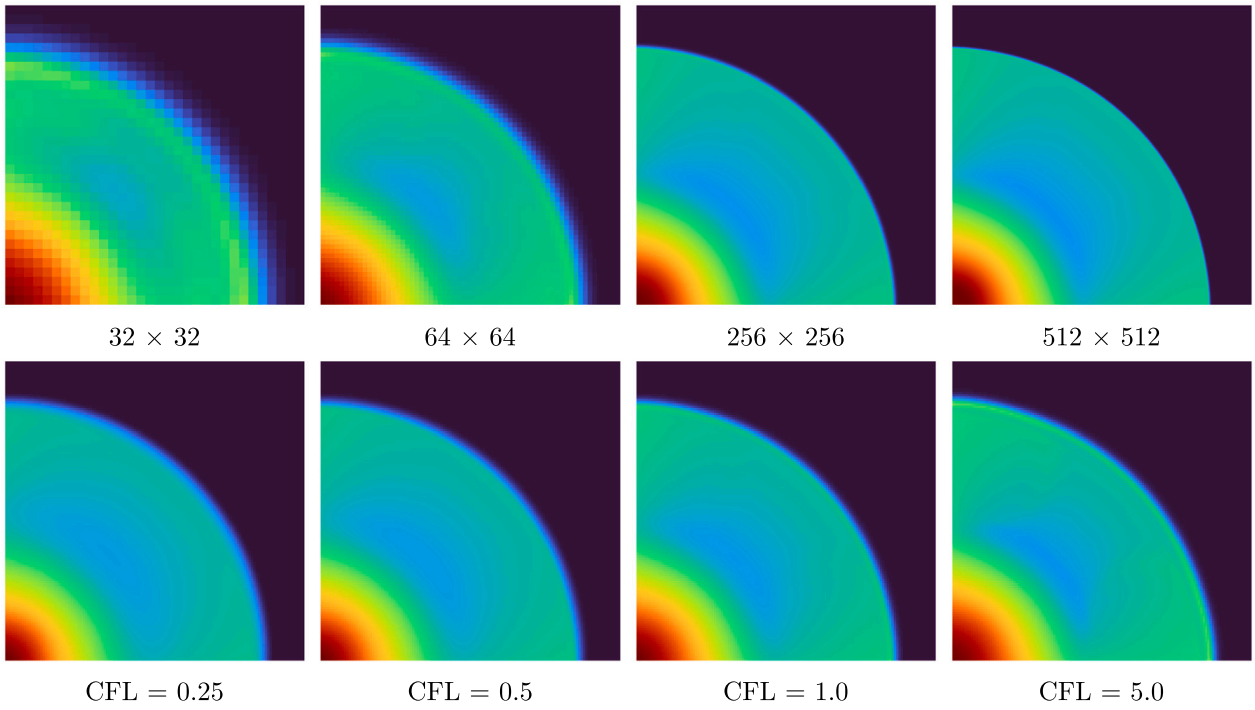


Fig. 6. Free surface height predicted by NN4PDEs at $t = 12\,\mathrm{s}$. The first row compares different mesh resolutions for a Courant number of 1 and the second row compares predictions from different Courant numbers ("CFL") at a resolution of 128^2.

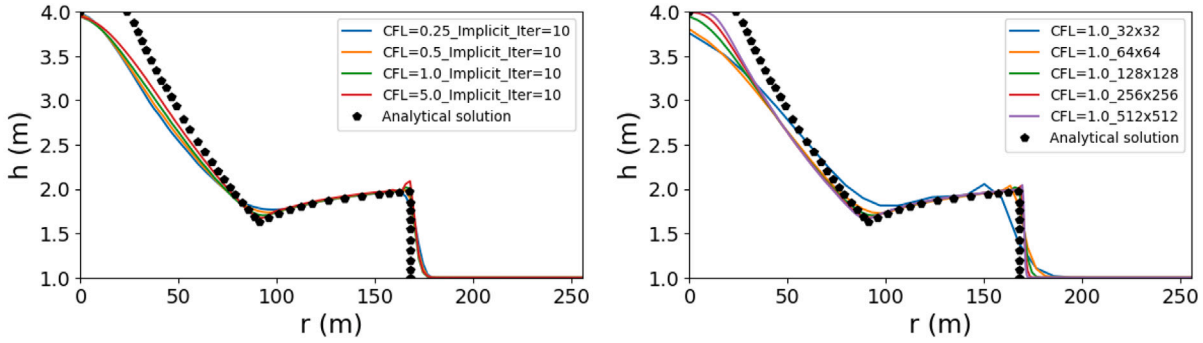
$200\,\mathrm{m} \times 200\,\mathrm{m}$ in the x and y directions. Initially, the water is at rest with a depth of

$$\eta(x, y, t = 0) = \begin{cases} 4 & \text{if } r = \sqrt{x^2 + y^2} \leqslant 100\,\mathrm{m}; \\ 1 & \text{otherwise}. \end{cases} \tag{29}$$

Symmetry boundary conditions are applied on all four boundaries, although the wave information does not reach the right or top boundaries during the simulation. The Courant number is measured assuming a speed of wave propagation of $\sqrt{4g}$ in which 4 is the maximum depth. Fig. 6 illustrates the 2D numerical prediction of free surface height using NN4PDEs with a linear ConvFEM discretisation at $t = 12\,\mathrm{s}$. A comparison of five different mesh resolutions for a Courant number of 1 shows that grid effects are apparent for grid sizes of 32^2 and 64^2, and smooth results are obtained for resolutions of 128^2 (third plot in bottom row), 256^2 and 512^2. In the following discussion, we use 128^2 as this is the lowest resolution for which we obtain smooth results. We also vary the Courant number for a mesh resolution of 128^2, demonstrating the robustness of the solver. As can be seen, the radial symmetry reduces slightly for higher Courant numbers.

Fig. 7 compares numerical results from NN4PDEs with the analytical solution at $t = 12\,\mathrm{s}$ along $(y = x)$. Results for free surface height obtained using four different Courant numbers (0.25, 0.5, 1.0 and 5.0) and a mesh resolution of 128^2 are presented in Fig. 7(a). A slight overshoot in surface height can be seen at the end of the wave in the case with a Courant number of 5. Fig. 7(b) shows the free surface height for five different mesh resolutions (from 32^2 to 512^2) and the Courant number of 1. The results demonstrate good agreement with the analytical solution in all cases. The comparison reveals that coarser grid sizes (32^2 and 64^2) lead to minor deviations at $r = 100\,\mathrm{m}$ and $r = 150\,\mathrm{m}$ in which r is measured from the origin (bottom left corner of the domain) along $y = x$. The discrepancy at $r = 0$ in Fig. 7(b) has been reported by others, for example, Fernández-Pato et al. (2018), who mention the diffusion of the numerical solution as a possible cause.

The comparison in Fig. 8 illustrates the performance of four different orders of ConvFEM discretisations (linear, quadratic, cubic and quartic). (The filters can be formed from code in the GitHub repository of Phillips (2022) for any grid spacing.) Two mesh resolutions are considered and the Courant number is set to a small value (0.1) in order to reduce the inaccuracies associated with the time truncation of the scheme. The solid lines correspond to the coarser resolution (32×32 nodes), whilst the dashed lines represent the finer resolution (128×128 nodes). The initial distribution has a depth of water given by Eq. (29) and solution values are extracted along $(y = x)$. All four ConvFEM approaches demonstrate close agreement with the analytical solution. Notably, the coarser mesh resolution reveals more pronounced differences among the models compared to the finer mesh resolution, particularly in terms of oscillations in free surface height at $r = 100\,\mathrm{m}$ and $r = 150\,\mathrm{m}$. On the coarser meshes, the higher-order ConvFEM results seem closer to the analytical and finer grid solutions. The same trend is observed for the higher spatial resolution results shown on the



(a) Courant numbers (0.25, 0.5, 1.0, 5.0) at a resolution of 128^2

(b) Mesh resolutions (from 32^2 to 512^2) for a Courant number of 1

Fig. 7. Comparison of free surface height predicted by NN4PDEs with the analytical solution taken at $t = 12\,\text{s}$. Various Courant numbers and grid spacings are investigated.

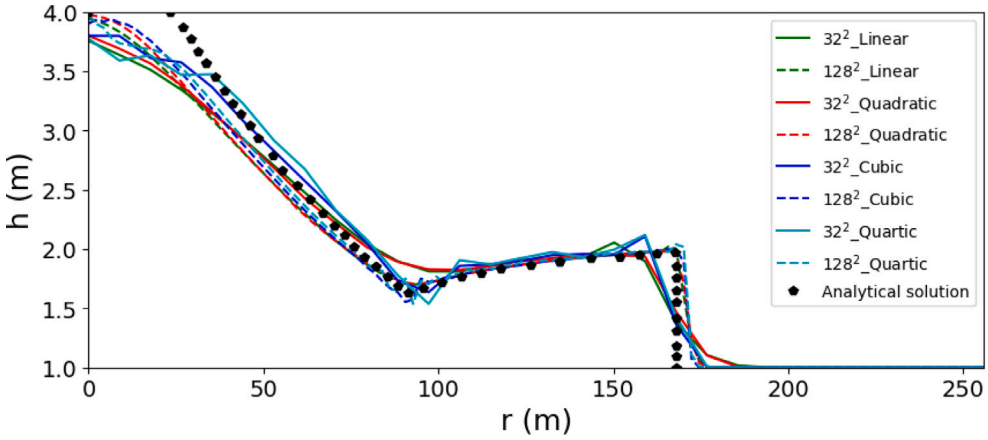


Fig. 8. Comparison of analytical solution with numerical results from NN4PDEs for different orders of ConvFEM discretisation and two different mesh resolutions at $t = 12\,\text{s}$. Shown here is the free surface height for linear, quadratic, cubic and quartic ConvFEM discretisations with resolutions of 32^2 and 128^2 for a Courant number of 0.1.

same plot, with the possible exception of the quartic ConvFEM results. This might be because the results of higher-order FE methods are more susceptible to oscillations, so the magnitude of artificial diffusion applied through the Petrov–Galerkin stabilisation scheme could be adjusted to overcome this issue. Fig. 9 shows the spatial distribution of free surface height represented in both 2D and 3D at $t = 12\,\text{s}$ for linear, quadratic, cubic and quartic elements. Especially in the 3D plots, one can see a reduced radial symmetry for the linear ConvFEM results as compared to the higher-order ConvFEM results. This is simply because the lower-order ConvFEM is less accurate and more prone to produce less accurate results by coarsening the mesh size, see Fig. 7(b). The oscillations observed at the shock front are present in results from all orders of ConvFEM discretisation but tend to increase with the order of ConvFEM. Quadratic, cubic and quartic elements show a more accurate solution near to $r = 100\,\text{m}$ (i.e. show more radial symmetry). These results suggest the use of quadratic ConvFEM as a compromise between accuracy and smoothness.

In Fig. 10, a comparison of the explicit and semi-implicit solvers is presented using two schemes, focusing on the sensitivity of the numerical predictions to the number of Jacobi iterations used for the correction step of the free surface height. The first scheme is fully explicit, solving for both the free surface height and velocity explicitly. The second scheme is semi-implicit, solving for the free surface height implicitly and the velocity explicitly. All results are obtained from a linear ConvFEM model shown at $t = 12\,\text{s}$ and are compared with the analytical solution (indicated by black circles). A mesh resolution of 128^2 is employed, and four Courant numbers are tested: 0.25, 0.5, 1.0 and 5.0. For Courant numbers 0.25, 0.5 and 1.0, both the explicit and semi-implicit schemes yield similar results, with little change observed upon increasing the number of Jacobi iterations used to solve for the free surface height, see Eq. (25). However, when the Courant number is increased to 5, the explicit time-stepping scheme struggles with stability, particular behind the wave front (at $r = 100\,\text{m}$). In contrast, the semi-implicit time-stepping scheme maintains robustness, showing its ability to generate stable numerical predictions even for larger Courant numbers. The iterative Jacobi method also seems to converge after about 10 iterations for a Courant number of 1 or above.

Fig. 11 presents the results for the circular dam break case at $t = 12\,\text{s}$, comparing mass lumping and consistent mass matrix approaches to the treatment of the time terms. The use of the consistent mass matrix takes into account the actual distribution of mass within the element, leading to a more accurate representation of the system's inertia properties. On the other hand, the lumped mass matrix simplifies the mass distribution by concentrating the mass at the nodes of the finite element mesh. This results in a diagonal mass matrix which is easier and more efficient to compute and invert. Five different mesh resolutions (32^2, 64^2, 128^2, 256^2, 512^2) are tested using linear and quadratic ConvFEM models, with the Courant number set to 0.1. In line with the observations made in relation to Fig. 8, finer meshes consistently yield better predictions for each ConvFEM model. Interestingly, when employing quadratic schemes, the differences between lumped and consistent mass results are less pronounced, unlike the linear scheme which exhibits slightly more sensitivity to mass lumping. Root mean square errors (RMSE) are given in Table 2 for the results shown in Fig. 11. The free surface height is averaged over the range $r \in [120, 200]$ at the time $t = 12\,\text{s}$. The RMSE values decrease and then

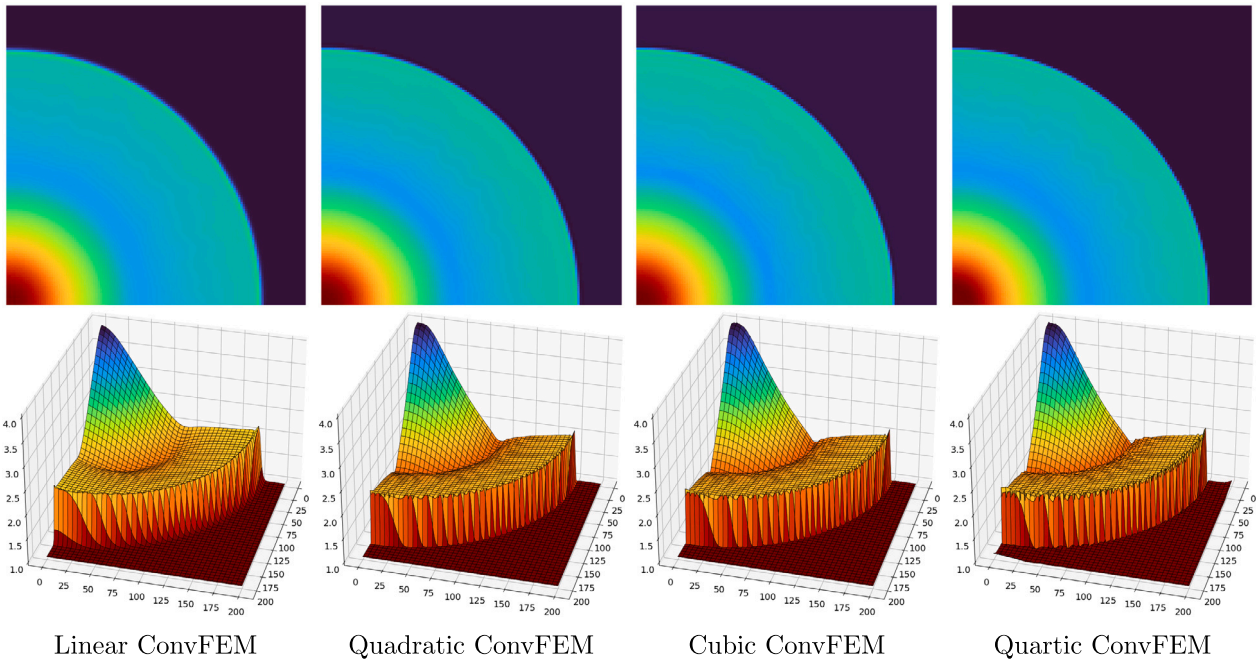


Fig. 9. Prediction of free surface height by NN4PDEs for different orders of ConvFEM at $t = 12\,\mathrm{s}$, a Courant number of 0.1 and a resolution of 128^2. The top row displays 2D contour plots of free surface height, and the bottom row shows 3D surface plots of the free surface height.

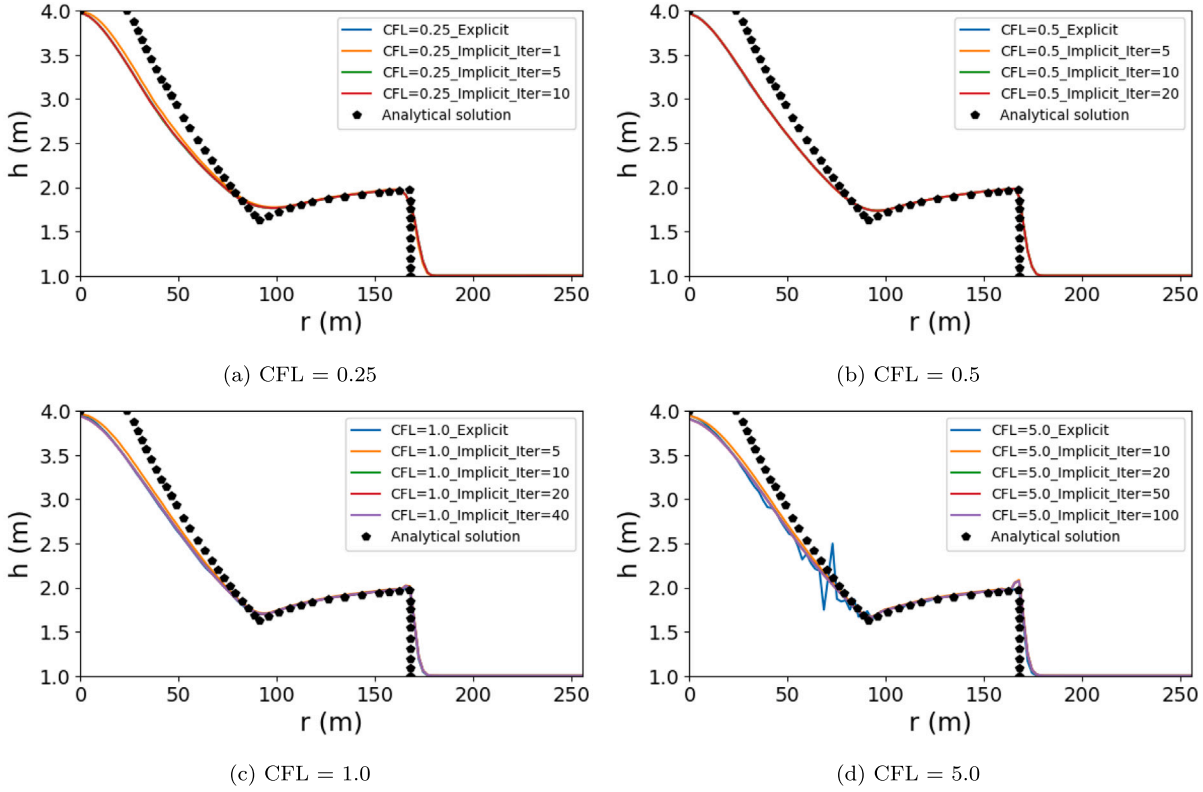


Fig. 10. Comparison of numerical results from NN4PDEs with the analytical solution for the circular dam break case at $t = 12\,\mathrm{s}$. Results are shown for an explicit scheme and a semi-implicit scheme (with different numbers of Jacobi iterations). The results are obtained by the Linear ConvFEM approach and with a mesh resolution of 128^2.

increase slightly as the grid spacing decreases. This is because with a smaller grid size, the curve matches the analytical solution better apart from at the water front where the discontinuity is. For this problem, there is very little difference in accuracy between using mass lumping and consistent mass formulations, there is also very little difference between solutions obtained with the linear and quadratic ConvFEM. This might be because of the dominance of water front (the shock wave), in the presence of which, second-order accuracy is expected to reduce to first-order accuracy in space.

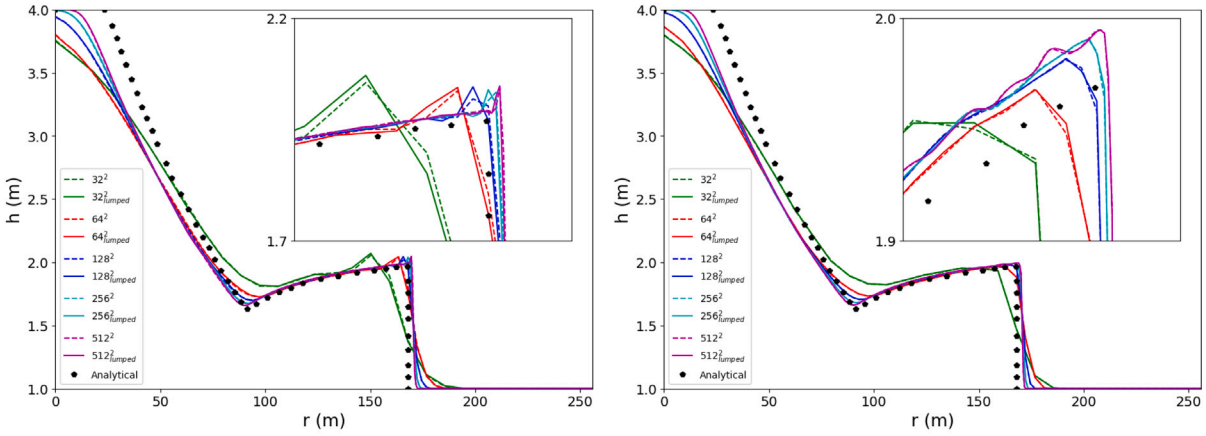


Fig. 11. Comparison of lumped and consistent mass results for the circular dam break case at $t = 12\,\mathrm{s}$ with different mesh resolution (32^2, 64^2, 128^2, 256^2, 512^2) by using two ConvFEM schemes. The results are obtained for a Courant number of 0.1.

Table 2
The RMSE values (deviation of the solutions from the analytical solution graphs shown in Fig. 10) of the simulations of the circular dam break problem shown in Fig. 11 calculated at $t = 12\,\mathrm{s}$ for $r \in [120\,\mathrm{m}, 200\,\mathrm{m}]$ and using a Courant number of 0.1.

	Mass term	32^2	64^2	128^2	256^2	512^2
Linear ConvFEM	Consistent	0.0436	0.0113	0.0186	0.0210	0.0228
	Lumped	0.0438	0.0119	0.0180	0.0202	0.0219
Quadratic ConvFEM	Consistent	0.0424	0.0097	0.0196	0.0222	0.0241
	Lumped	0.0427	0.0099	0.0195	0.0222	0.0240

Additionally, Fig. 12 investigates mass conservation for a quadratic ConvFEM discretisation for three different mesh resolutions with mass lumping. It illustrates the time evolution of normalised mass concentration from $t = 0\,\mathrm{s}$ to $t = 12\,\mathrm{s}$. The quantity $\bar{h}^n/\bar{h}^0$ represents normalised free surface height integrated over the domain, where $\bar{h}^n$ denotes the integrated free surface height over the volume, $\bar{h}^n = \int_V h^n dV$, and $\bar{h}^0$ denotes the initial value of the integral. Values close to 1 indicate that the solver effectively conserves mass and that the mesh resolution does not affect conservation. Fig. 12 shows a small increase in mass of $5{\times}10^{-5}$ as the simulation progresses. This small error in mass may result from round-off errors or the tolerance used in the Jacobi iteration. We note that the formulation is mass-conserving if one exactly solves the wave equation for free surface height, Eq. (25). Otherwise, if the wave equation is only approximately solved through a fixed number of iterations, mass will only be conserved approximately. Our experience suggests that even using a small number of Jacobi iterations (e.g., 1 to 10) is sufficient for accurate mass conservation.

3.2. Laboratory-scale dam break over a triangular-shaped hill

The domain consists of a channel $38\,\mathrm{m}$ long which contains a triangular-shaped hill (see Fig. 13(a)). The corresponding physical experiments are detailed in Liang and Marche (2009), see in particular Figure 10 of Liang and Marche (2009), and are modelled here in a 2D square domain measuring $38\,\mathrm{m}$ by $9\,\mathrm{m}$, but with symmetry boundary conditions in the y direction so as to emulate a 1D model. The results are collected along the line given by $y = 4.5\,\mathrm{m}$. A symmetry boundary condition is also used at the left-hand side boundary and an open boundary condition (zero derivative of all fields) is used on the right-hand side boundary. Linear ConvFEM is used in the simulations along with a Manning coefficient of $n_M = 0.0125\,\mathrm{s\,m^{-1/3}}$. The width of the initial column of water is $15.5\,\mathrm{m}$ and its height is $0.75\,\mathrm{m}$ as shown in Fig. 13(a). The water column collapses due to gravity. The free surface height h is measured at the gauges also shown in Fig. 13(a). Time histories of the experimental results and the simulation results are shown at these gauges in Fig. 13(b)–(h). We note the close match between the experiments and the simulations for all three different grid spacings of $\Delta x = 0.16\,\mathrm{m}$, $\Delta x = 0.08\,\mathrm{m}$ and $\Delta x = 0.04\,\mathrm{m}$. The spatial distribution of water at various time levels is shown in Fig. 14 along with the water velocity shown in Fig. 15. The plots show a wave propagating towards the triangular-shaped hill. Some of the water is then recedes back down the hill leading to a sloshing motion. Some of the water spills over the hill, wetting the far side of the hill initially, but, as water falls under gravity, the hill becomes dry again. This problem tests the wetting and drying, and the front modelling capabilities of the proposed NN4PDEs model. In this figure, one can observe that the predictions of NN4PDEs for free surface height are similar to the values measured in the experiment.

3.3. Propagation of flood water across a floodplain

This scenario mimics the failure of an embankment through breaching or overtopping, also modelled computationally by Néelz and Pender (2013). The simulation has a wide and level floodplain measuring $2000\,\mathrm{m}$ by $2000\,\mathrm{m}$, with a source of water uniformly placed in the centre of the domain and spanning $20\,\mathrm{m}$ by $20\,\mathrm{m}$. The peak flow is $20\,\mathrm{m^3\,s^{-1}}$ and approximately 5 h of real time is simulated. The grid spacing is $5\,\mathrm{m}$ and the Manning coefficient is $n_M = 0.05\,\mathrm{s\,m^{-1/3}}$. For the first 5 min, the flow rate is set to zero. For the next 55 min it increases linearly up to a value of $20\,\mathrm{m^3\,s^{-1}}$. It remains constant at $20\,\mathrm{m^3\,s^{-1}}$ for 180 min and then decreases linearly to zero over 60 min. The primary aim of modelling this test case is to evaluate the capability of NN4PDEs to simulate the speed of flood wave propagation and make predictions about transient velocities and water depths with substantial wetting. This evaluation is particularly relevant for the modelling of fluvial and coastal inundation that arises from embankment breaches. The right-hand half of the domain is shown in Fig. 16 indicating in (a) the location of sensors 1, 3, 5 and 6 (taken from Shaw et al. (2021)) as well as our results at two-time levels. The spatial variation after one hour agrees well with that of Shaw et al. (2021) as can be seen by comparing plots (a) and (b) of Fig. 17. The results are also shown at a time of 3 h into the simulation. Fig. 17 presents the speed of the flow and the water depth at the four sensor points, and one can see that there is a close match between NN4PDEs results (red and blue lines represent linear and quadratic ConvFEM models) and the discontinuous Galerkin (DG) results (black lines) of Shaw et al. (2021) helping to validate the NN4PDEs approach.

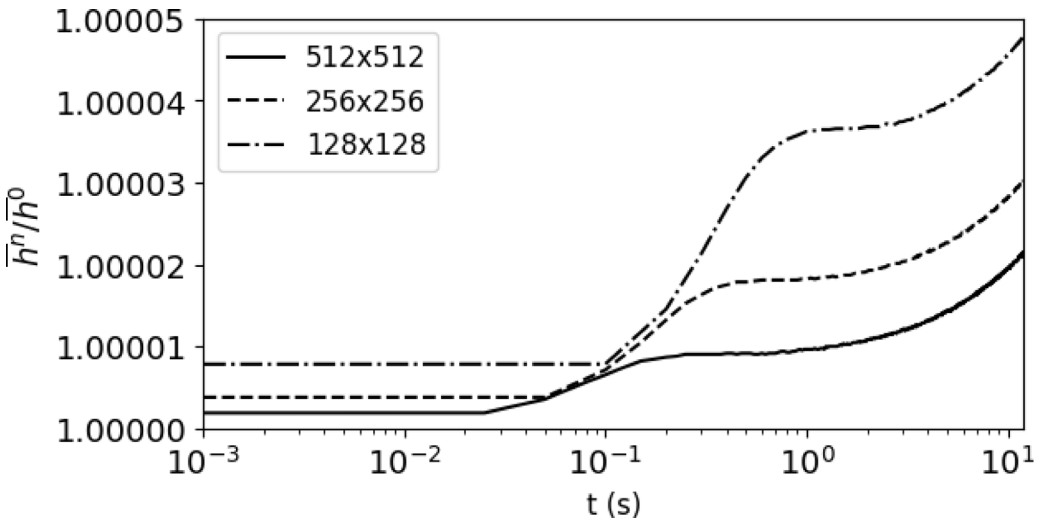


Fig. 12. Time history of the normalised mass concentration on the interval $0 \leqslant t \leqslant 12$ with different mesh resolutions (128^2, 256^2, 512^2). The results are obtained with a Courant number of 0.1 and with mass lumping.

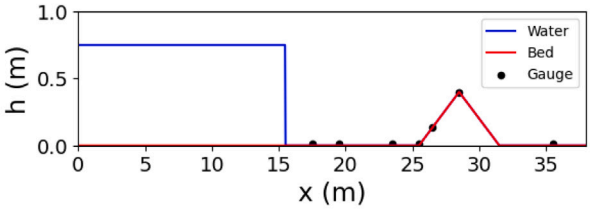
3.4. Flooding in Carlisle

The Carlisle 2005 flood event was a significant natural disaster that occurred in early January 2005 in Carlisle, Cumbria, England. This catastrophic flood was the result of a combination of prolonged heavy rainfall and saturated ground conditions, exacerbated by the confluence of the Rivers Eden, Petteril, and Caldew. The intense weather conditions led to the rivers breaching their banks, causing widespread flooding across the city. The impacts of the flood were severe, with thousands of homes and businesses inundated, major disruptions to transportation and infrastructure, and extensive damage to properties. The event also resulted in tragic loss of life and highlighted the need for improved flood defence mechanisms in the region.

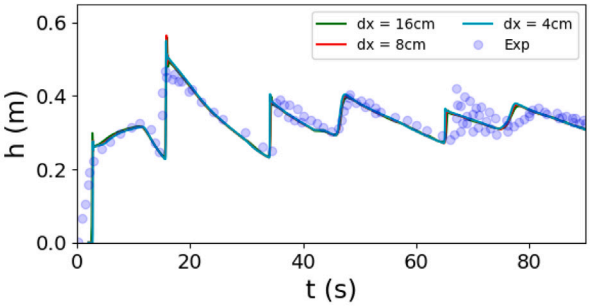
In this section, we simulate the evolution of a city-scale flood originating from multiple river inflows. This problem was investigated by Kesserwani and Sharifian (2023) who provided detailed information about the problem setup and compared results from several computational models. To validate NN4PDEs, we will use in-situ water depth measurements given in Kesserwani and Sharifian (2023) that were collected at the three gauges during the flood. Morales-Hernández et al. (2021) have obtained numerical results for this test case using TRITON, with which we shall also compare our results. The computational domain has 951 nodes and 611 nodes in the x and y directions respectively over a domain of 4.75 km by 3.05 km (the same as that used by Morales-Hernández et al. (2021) in their TRITON model). A uniform grid spacing of 5 m has been used with a time-step size of 0.5 s. The Manning coefficient n_M is set to $0.055\,\mathrm{s\,m^{-1/3}}$. The study area, illustrated in Fig. 18, spans approximately $14.5\,\mathrm{km^2}$ within the city of Carlisle. Over three days, flooding is initiated by three inflow hydrographs at the upstream points of the Rivers Eden, Petteril and Caldew (Figs. 18(a) and 18(b)), with a maximum flow rate of just over $1200\,\mathrm{m^3/s}$ (as shown in Fig. 18(c)). The unit of s_h in Eq. (1) is m/s and in order to use the flow rate values from Fig. 18(c) in the calculations, it should be multiplied by the area in which the source is applied, which in this case is $25\,\mathrm{m^2}$ for each cell with a source in it. The River Eden is approximately 11 cells wide, while the Rivers Caldew and Petteril are both 6 cells wide. The time history of water depth is recorded at 15 sampling points, including at the three gauges at Sheepmount, Botcherby Bridge and Denton Holme.

The results obtained from the NN4PDEs solver exhibit good agreement with the observed values. Fig. 19 shows time histories from NN4PDEs measured at 15 sampling points during the flood simulation and compares these with predictions from the open source 2D hydrodynamics flood model, TRITON, obtained by Kesserwani and Sharifian (2023). Three of the sampling points coincide with the three gauges at which we have observational data, and this is also included in the plots where available. Very good agreement is demonstrated with this observed data. At certain sampling points, disparities between the results of NN4PDEs and the TRITON results can be seen. The largest differences are between the quadratic ConvFEM and TRITON, occurring at the bus depot and at the Substation, see Fig. 19. We attribute these differences partly to the fact that the there is a grid resolution issue associated with the bathymetry, so the location of the sampling points may not be precise. This assertion can be substantiated by the fact that for some points, there is a good match at sampling points where the surrounding topography does not vary very much, such as Sheepmount, Botcherby Bridge and Denton Holme. However, for other points with highly heterogeneous surrounding topography, even slight discrepancies in the location of the sampling points can significantly impact the output (for example, near buildings). However, if one compares the spatial distribution of the linear and quadratic ConvFEM results flooding depths near these two points (Fig. 20), they appear to be almost identical. This is further investigated in Appendix C. Comparing spatial distributions of NN4PDEs with results from TRITON, RMSEs for linear and quadratic ConvFEM at Sheepmount are 0.785 and 0.679; at Botcherby bridge are 0.406 and 0.208; and at Denton Holme are 0.278 and 0.56. These three locations were selected because real flood event measurements are available at these points, enabling a meaningful comparison. The range of values exhibited by NN4PDEs, shown in the plots of Fig. 19, are consistent with those appearing in Kesserwani and Sharifian (2023) where results are presented from a wide range of different computational models. Kesserwani and Sharifian (2023) present observation results at only three of the gauges, located at Sheepmount, Botcherby Bridge and Denton Hulme, and for these locations, our results are similar to the TRITON results. Another potential source of discrepancy for our results is the water mass volume rate, which can only be estimated (see Fig. 18(c)).

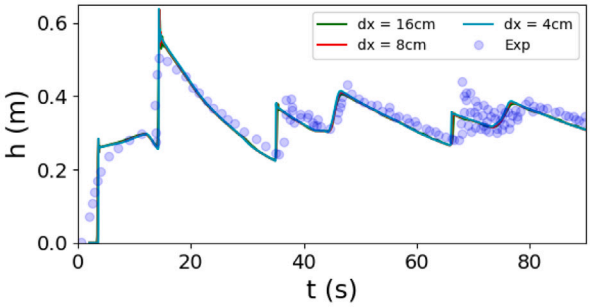
The spatial variation of the water depth during the flooding event is shown at various time levels in Fig. 20 along with the speed of the water in Fig. 21 and in two focused regions at 42 h into the flood event in Fig. 22. Notice in the latter, there are small differences in which regions are flooded and which are not, comparing the linear and quadratic discretisations. However, generally speaking, both linear and quadratic ConvFEM approaches produce similar results. The simulation results also reveal that the maximum flooded area was $6.14\,\mathrm{km^2}$ out of



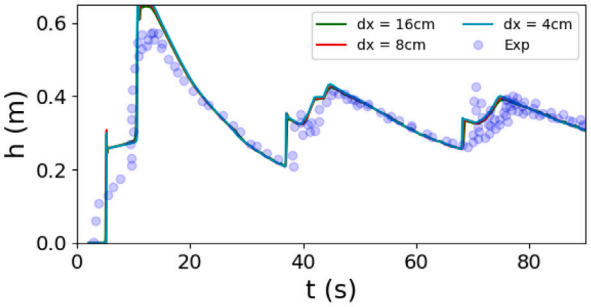
(a) Initial conditions and location of gauges.



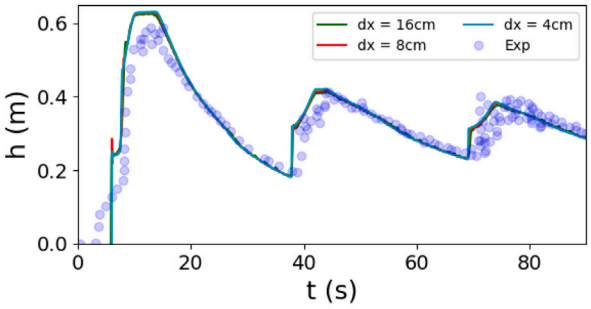
(b) Water depth from gauge at x =17.5 m.



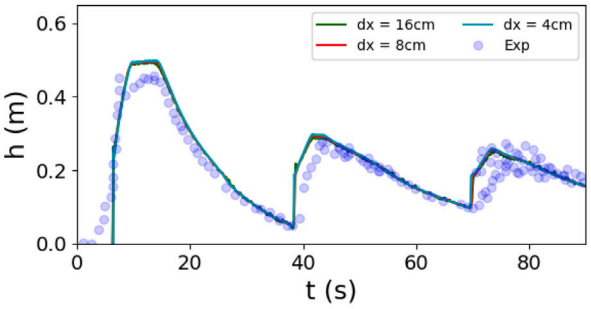
(c) Water depth from gauge at x =19.5 m.



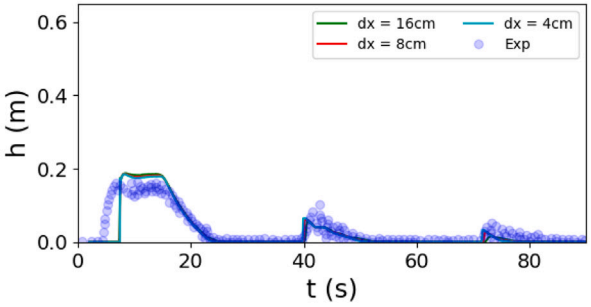
(d) Water depth from gauge at x =23.5 m.



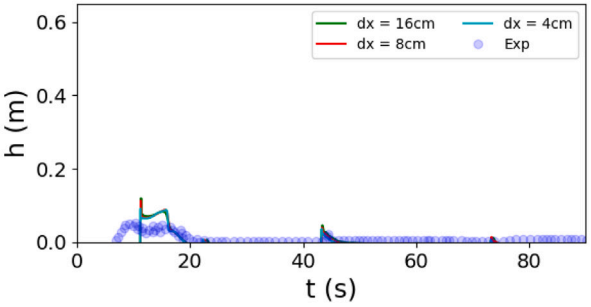
(e) Water depth from gauge at x =25.5 m.



(f) Water depth from gauge at x =26.5 m.

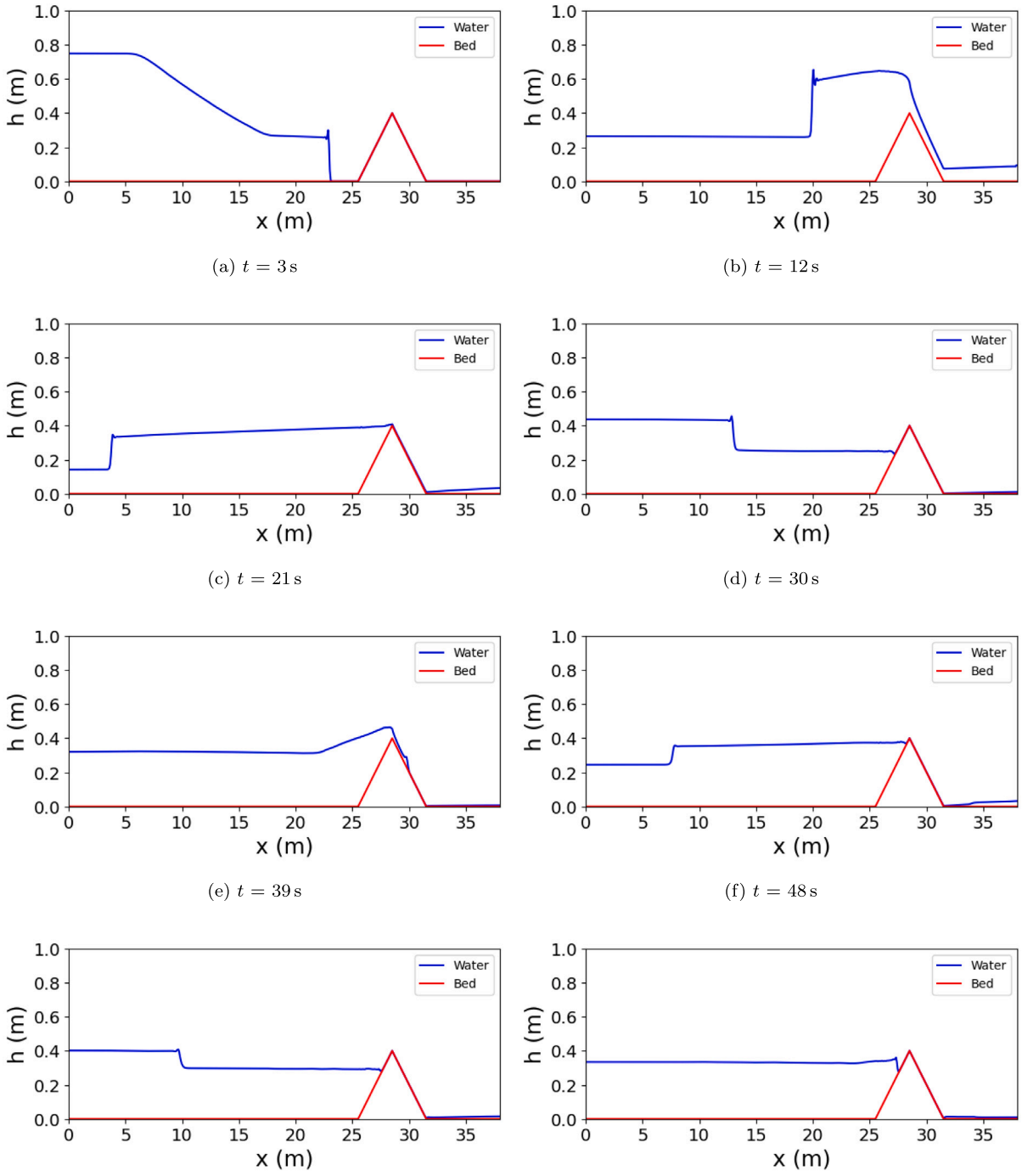


(g) Water depth from gauge at x =28.5 m.



(h) Water depth from gauge at x =35.5 m.

Fig. 13. Comparison of time histories of the predictions obtained from NN4PDEs with experimental measurements for the dam break over a triangular-shaped hill. Three different mesh resolutions (0.16 m, 0.08 m and 0.04 m) are used.



(a) $t = 3\,\mathrm{s}$ (b) $t = 12\,\mathrm{s}$

(c) $t = 21\,\mathrm{s}$ (d) $t = 30\,\mathrm{s}$

(e) $t = 39\,\mathrm{s}$ (f) $t = 48\,\mathrm{s}$

(g) $t = 57\,\mathrm{s}$ (h) $t = 66\,\mathrm{s}$

Fig. 14. Snapshots of water surface in the dam break over a triangular-shaped hill along the middle plane in the y direction from $t = 3\,\mathrm{s}$ to $t = 66\,\mathrm{s}$ in $9\,\mathrm{s}$ intervals.

$14.5\,\mathrm{km}^2$, occurring after $48\,\mathrm{h}$, which constitutes 42.3% of the total area and approximately affected 1600 buildings.

3.5. Computational speed of NN4PDEs

The computational speed of NN4PDEs can be approximated by hypothesising that the computational time will be proportional to the number of FE nodes and also proportional to the product of the number of nodes and the number of Jacobi iterations for the free surface height. To obtain the total time for a simulation, we multiply by the number of time levels and number of iterations within a time step. This can be written as

$$\text{simulation duration} = (1.3)^{\mathrm{p}-1}\mathcal{N}_{\mathcal{I}}\mathcal{T}\mathcal{N}\left(M_s + JF_s\right) \tag{30}$$

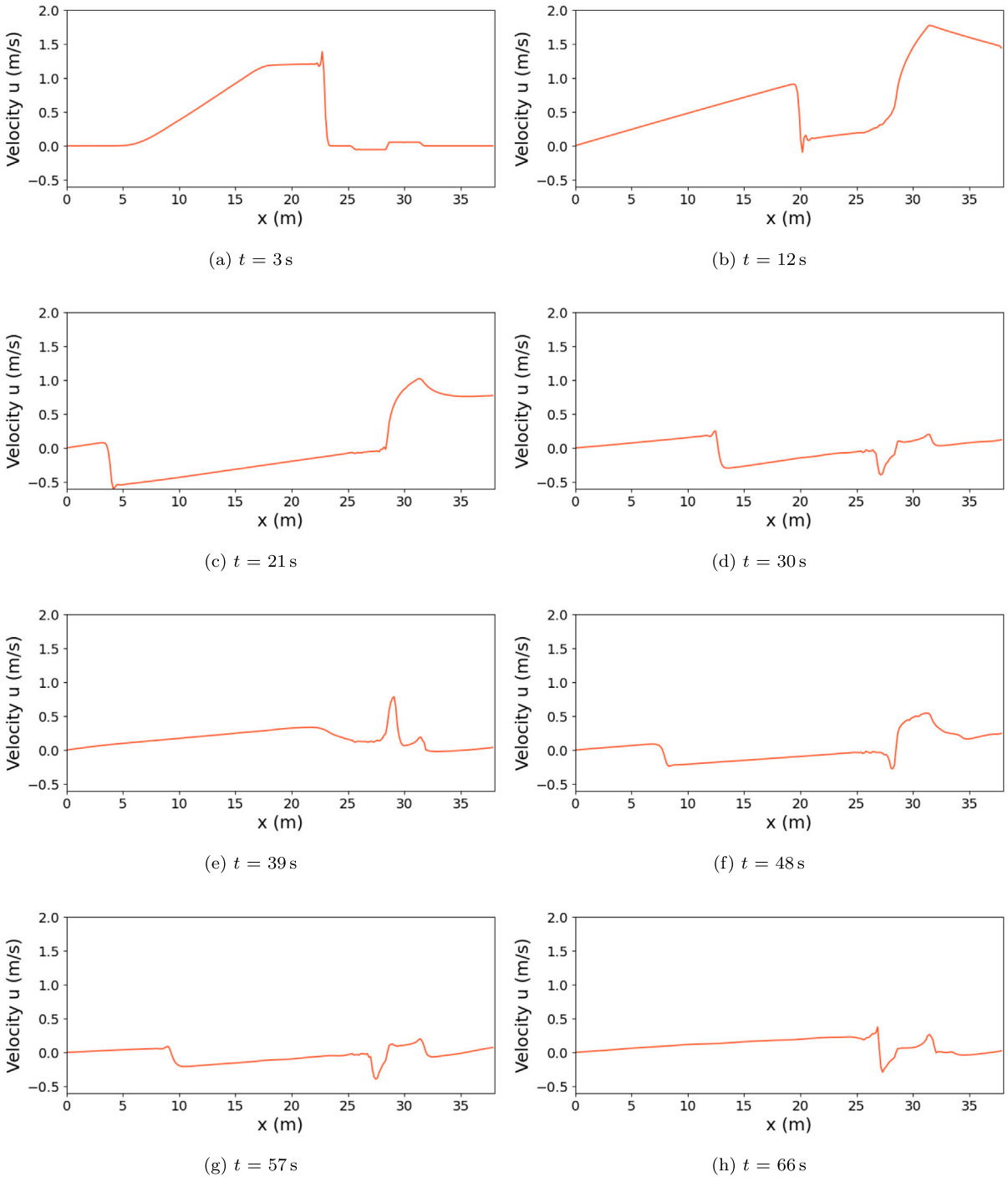
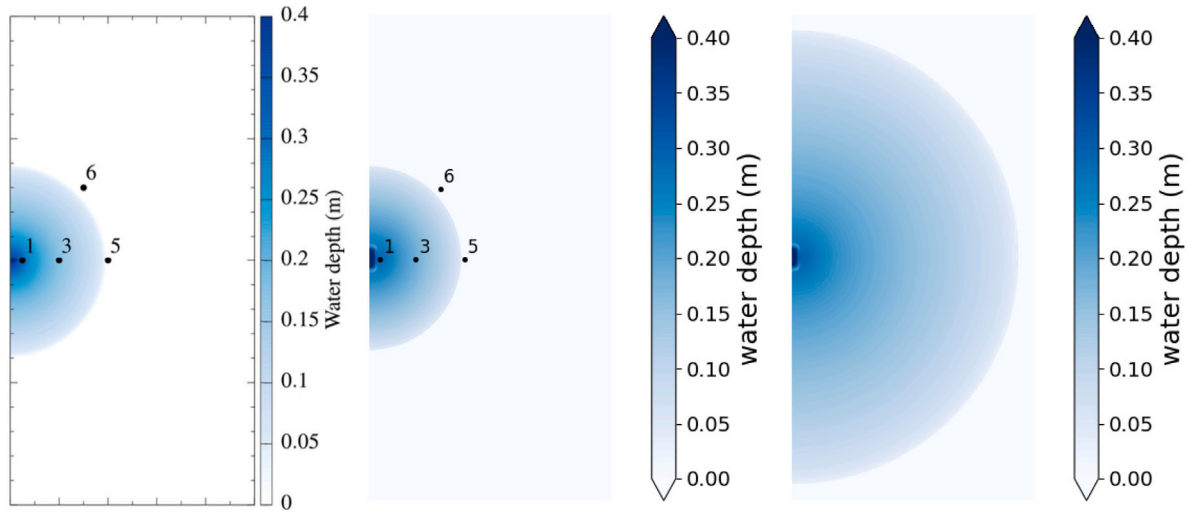


Fig. 15. Snapshots of the water velocity in the dam break over a triangular-shaped hill along the middle plane in the y direction from $t = 3\,\mathrm{s}$ to $t = 66\,\mathrm{s}$ in $9\,\mathrm{s}$ intervals.

in which p is the polynomial order (p = 1 refers to linear and p = 2 refers to quadratic), $\mathcal{N}_I$ is the number of iterations within a time step (the default is $\mathcal{N}_I = 2$ unless otherwise stated and is able to produce a second order in time scheme), $\mathcal{T}$ is the number of time steps, $\mathcal{N} = N_x \times N_y$ is the number of FE nodes, J is the number of Jacobi iterations used for the free surface height, and M_s and F_s are the two constants of proportionality which depend on the computer being used. Here we use a NVIDIA A100 Tensor Core GPU and in which case $M_s = 2.76 \times 10^{-9}$ s and $F_s = 0.92 \times 10^{-10}$ s. These numbers have been obtained by running two simulations of Carlisle flooding test case, (first with $J = 1$ and $\mathcal{T} = 1000$, and second with $J = 10$ and $\mathcal{T} = 1000$) and then solving the resulting simultaneous equations for M_s and F_s. The Carlisle flooding test case shown in Fig. 19 was run for a period of 68 h with a time step of 0.8 s ($\mathcal{T} = 306\,000$) and $\mathcal{N} = 951 \times 611 = 581\,061$ FE nodes with quadratic elements (p = 2), $\mathcal{N}_I = 2$ iterations within the time step and $J = 1$ Jacobi iteration (for free surface height). In this case, Eq. (30) predicts a time of 24.02 mins and the actual simulation time was 21.28 mins. For the case where only one iteration was used within the time step ($\mathcal{N}_I = 1$), Eq. (30) predicts a time of 12.01 mins and the actual simulation time was 10.64 mins. For comparison purposes the finite-volume TRITON code takes 7 mins for the same problem, albeit a different discretisation.



(a) LISFLOOD after one hour [8] (b) NN4PDEs after one hour (c) NN4PDEs after three hours

Fig. 16. Semi-circular flood water depth after one hour and three hours. The position of the gauges is shown in (a) and (b).

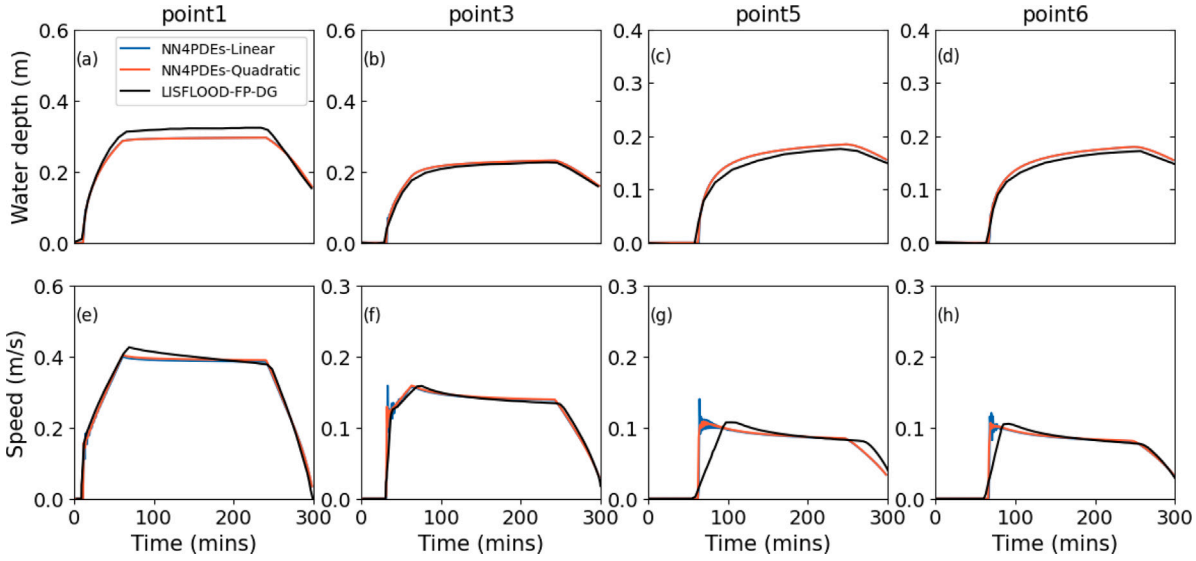


Fig. 17. Predictions of water depth and velocity speed at gauge points 1, 3, 5 and 6 using NN4PDEs with linear and quadratic discretisations. The two numerical solutions often overlay one another. The positions of the gauges are shown in Fig. 16. Results obtained by Néelz and Pender (2013) using LISFLOOD are shown for comparison.

4. Discussion

Implicit methods are particularly useful in situations where it is not so important to track waves across the domain or where water inertia is not important e.g., in some flood plain scenarios where the dynamics evolve slowly over time. In this work, a discretised wave equation is solved for free surface height, see Eq. (25). Looking at this equation it can be noticed that the diagonal term becomes dominant when the time step is small. This enables the Jacobi or multigrid solution method to converge very rapidly. In fact multigrid methods are hardly needed, and one or a few multigrid levels with a Jacobi update achieve adequate convergence of the solution. So, in the case of small time steps, this approach acts in a similar way to a fully explicit solver. However, if the Courant number of the free surface wave is greater than one, such as could occur in deep water, the scheme may remain stable. If this Courant number is less than 10, say, then only a small number of multigrid levels would be required to achieve convergence. For larger Courant numbers, the full power of the multigrid method can be used. So the implicit wave equation approach provides flexibility in terms of time step choice without incurring the reduction in speed associated with an implicit approach over an explicit approach as long as the multigrid solver is tuned for the problem being solved.

In addition, looking at the balance between the lumped mass term and the diffusion term on the left-hand side of Eq. (25), it can be noticed that a consistent mass term starts to emerge from the addition of the two terms up to a Courant number of 2 based on the free surface wave speed $c = \sqrt{g\eta}$. This emergence is due to the fact that a diffusion operator can be formed from the difference between a lumped mass and consistent mass matrix, so that, at a Courant number of 2, the left-hand side matrix of Eq. (25) can be seen as a consistent mass matrix. After that point, the real power of the implicit approach would emerge in terms of stability and accuracy. It should also be noted that the residuals in Eq. (25), as well as Eqs. (22) and (23), can contain a consistent mass matrix operating on the time term, which enhances the time accuracy of the approach. This only becomes noticeable when two or more iterations of the whole system are applied, see Fig. 5. Generally, two iterations (one predictor step and one corrector step) are used here to achieve second-order accuracy in time as well as having

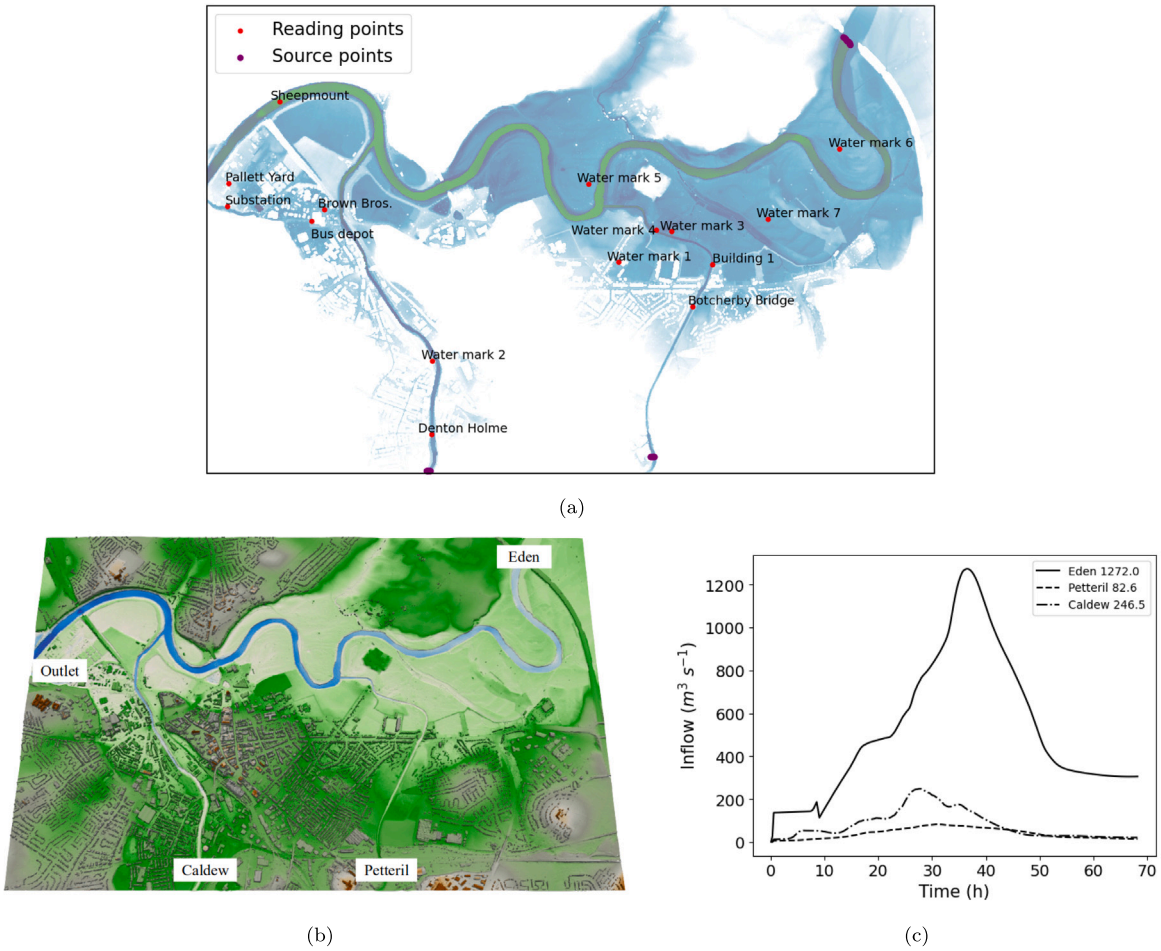


Fig. 18. Carlisle 2005 flooding: (a) the flood extent at the maximum Eden inlet rate (1200 m/s) including the positions and names of the sampling points; (b) the three rivers and topology of the area being modelled; (c) influx (volume rate of water) into Carlisle area from the Rivers Eden, Petteril and Caldew versus time.

options for consistent time mass matrices.

The model developed in this study can be run on GPUs, and is capable of using high-order finite elements and semi-implicit or explicit time integration. Stabilisation is achieved through a Petrov–Galerkin formulation which reduces the unphysical oscillations that can occur due to advection processes, shocks, wetting and drying. The result is an implicit LES model for turbulent flows. Other methods such as Riemann solvers and Godunov schemes could be implemented within the NN4PDEs framework, although this was not necessary for the test cases we chose. The successful validation of NN4PDEs against the Carlisle flooding in 2005, highlights the potential of this model, which runs with competitive speed on GPUs and achieves 5 m resolution for a domain of 4.75 km by 3.05 km.

Since neural network models are typically developed using the same machine-learning library (e.g., PyTorch) they are relatively easy to integrate, especially if they are exploiting similar neural network methods such as convolutional neural networks. There are a number of reasons why one might want to couple neural networks to produce a single neural network. Some reasons are: (1) Combining Sub-Grid-Scale models that are based on (trained) neural networks with an NN4PDEs model (with analytically defined network weights such as the model used here) to form a single neural network that may have advantages such as computational speed and differentiability. (2) Being able to combine NN4PDEs within a neural network to help form more accurate solutions or conservation of certain key variables (e.g., water depth). (3) Being able to differentiate the model (form gradients with respect to controls) so that the model may be used, together with the optimisation methods also embedded in neural networks, to assimilate data, solve inverse problems or optimise designs or control problems. (4) Form priors from AI for data assimilation or design or control problems based on physically realistic solutions and then optimise within these priors by coupling the priors to the NN4PDEs model. This might be useful in order to form initial conditions for a flooding model, for example. (5) Being able to use the vast array of software available in AI as well as its capabilities, e.g., mixed arithmetic for increased speed. (6) Being able to integrate different AI models together with relative ease compared with existing problems is particularly important for integrating different types of physics.

5. Conclusions and future work

We present a new FEM-based method for solving the Shallow Water Equations (SWEs) which uses a non-linear Petrov–Galerkin scheme for stabilisation. The numerical discretisation is written as a convolutional neural network (CNN) in which the weights are determined by the discretisation itself and not through training. This approach is referred to as NN4PDEs and has previously been applied to single-phase flow, multiphase flow, neutron diffusion and Boltzmann transport problems. Our focus here is the validation of the approach for solving SWEs, as well as investigating benchmark problems with analytical solutions, laboratory experiments, and real-life flooding events.

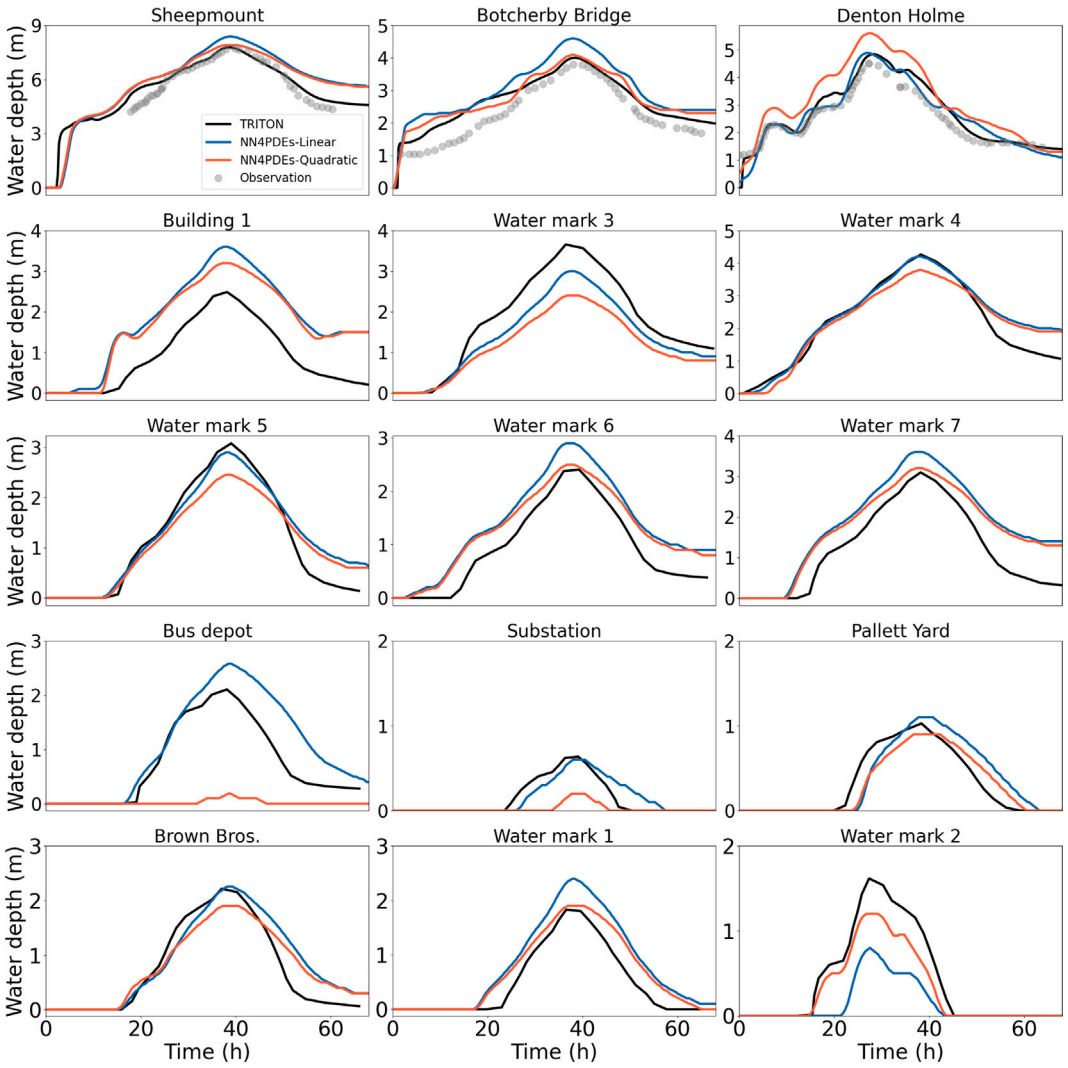


Fig. 19. Numerical prediction of water depth versus time using linear and quadratic ConvFEM discretisations. Time histories are shown at the sampling points (shown in Fig. 18(a)). A comparison is made with results from TRITON, obtained by Morales-Hernández et al. (2021), indicated by black lines. Grey circles represent the observations taken at the three gauges located at Sheepmount, Botcherby Bridge and Denton Holme.

For the circular dam break problem, we compare the NN4PDEs results with the analytical solution, showing that quadratic elements result in a more accurate solution than linear elements, but that cubic and quartic elements can lead to an oscillatory solution. We also show that for Courant numbers up to 1, a fully explicit solver is capable of stable results. For Courant numbers larger than this, having an implicit scheme for the free surface height is required to obtain a stable solution. Jacobi iterations were found to be sufficient to solve the implicit scheme. The effect of mass lumping was also investigated and was shown to have little impact on mass conservation.

The triangular-shaped hill demonstrates that NN4PDEs can model wetting and drying, and the results compare very well with data from a laboratory experiment. The floodplain test case presents a scenario involving with substantial wetting, particularly relevant for the modelling of fluvial and coastal inundation that can arise from embankment breaches. Results obtained from NN4PDEs agree well with other computational results from the literature. Finally, the Carlisle flooding event of 2005 is a real-world test case on which to demonstrate NN4PDEs. The results showed close agreement with another computer simulation and with measurements of water depth taken at three gauges.

Broadly speaking, the NN4PDEs approach generates results that compare favourably with analytical solutions, other numerical results and real data. The models presented here were run on one GPU, but the code could be run on multiple GPUs or AI processors, in case of the latter, we would expect to see a significant speed-up. A variable resolution code will be developed to have high resolution where needed and to coarsen the grid in areas that are less affected by the flood. The code will be parallelised to run the model on multiple GPUs.

Key limitations of the present work are: (1) The need to use structured grids with uniform grid spacing. This will be relaxed, in future work, by having regions (subdomains) of higher resolution and other regions of lower resolution. We are also developing a version of this approach which uses fully unstructured FEM meshes that allow greater flexibility in resolution (Li et al., 2024). (2) The 2D nature of the SWE model precludes the resolution of non-hydrostatic effects (vertical inertia) that may be important in fast deep water or around objects. This may be overcome in future work by using 3D NN4PDEs flood models, and will also enable the easy coupling of drainage systems and the air flow above the flood even including rain and cloud formation.

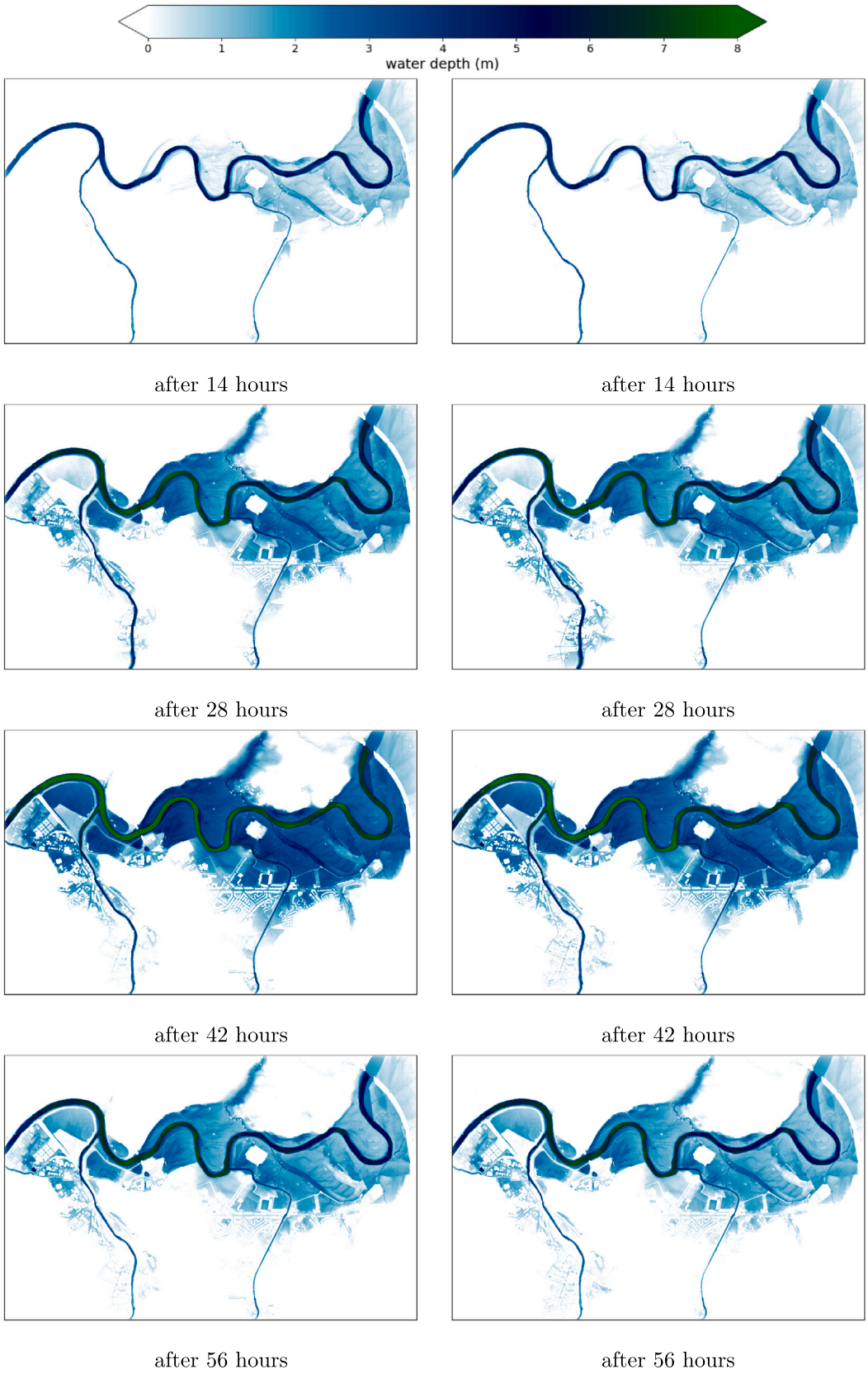


Fig. 20. Spatial variation of water depth in the flooded area after 14, 28, 42 and 56 h as predicted by linear (left) and quadratic (right) ConvFEM models.

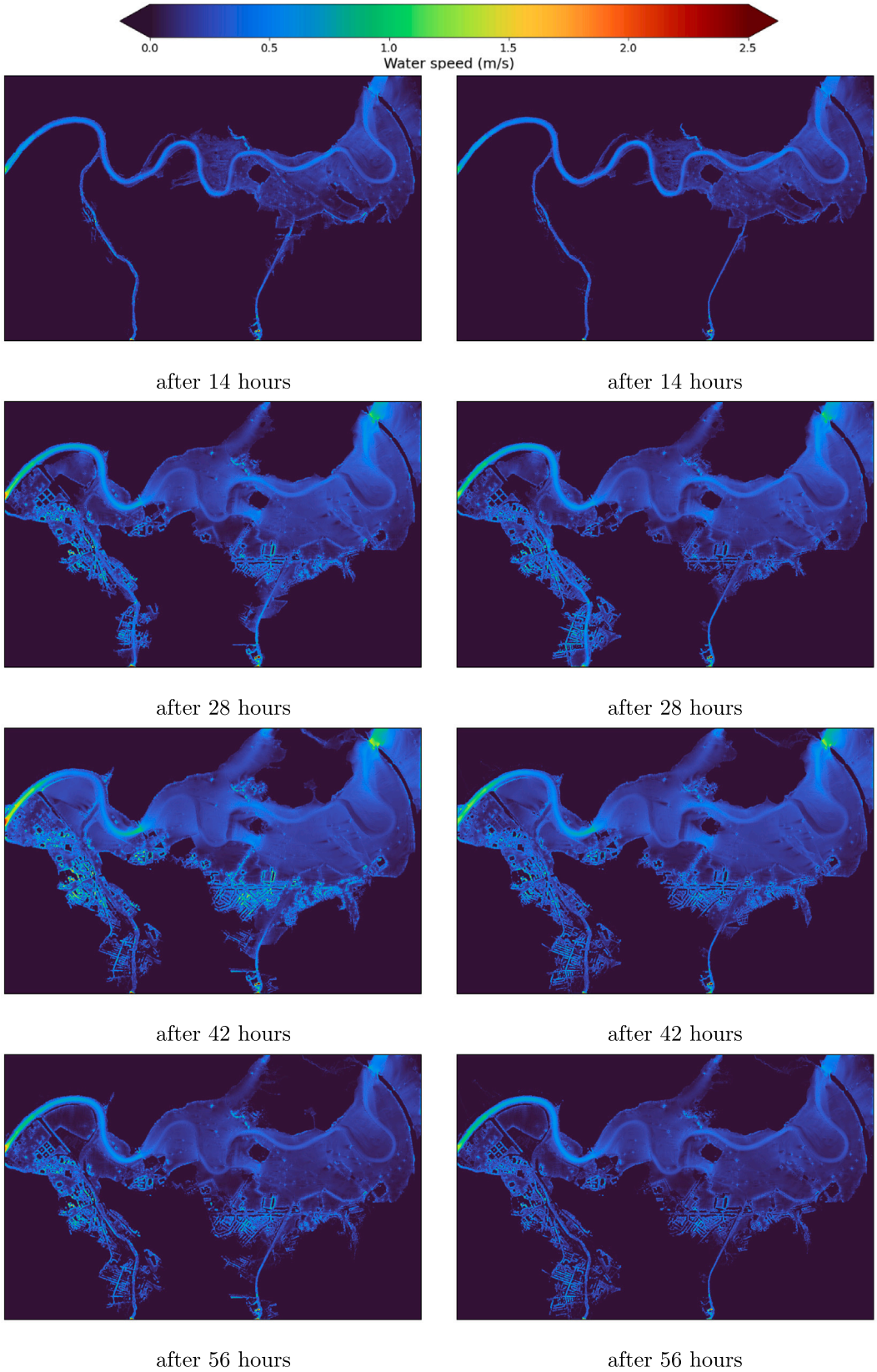


Fig. 21. Spatial variation of water speed in the flooded area after 14, 28, 42 and 56 h as predicted by linear (left) and quadratic (right) ConvFEM models.

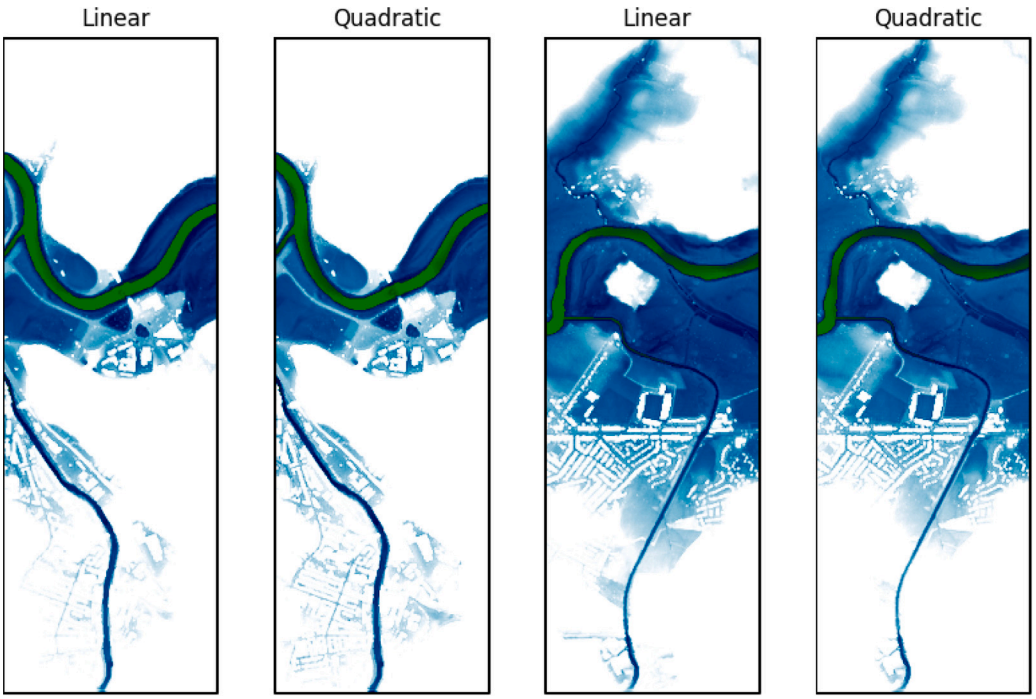


Fig. 22. Focusing on two areas in the Carlisle domain, and comparing the linear and quadratic ConvFEM simulation water depths at 42 h into the flooding event.

CRediT authorship contribution statement

Boyang Chen: Writing – review & editing, Supervision, Software, Methodology. **Amin Nadimy:** Writing – review & editing, Software, Methodology. **Claire E. Heaney:** Writing – review & editing, Writing – original draft, Supervision, Software, Methodology. **Mohammad Kazem Sharifian:** Writing – review & editing, Supervision, Resources. **Lluis Via Estrem:** Writing – review & editing, Supervision, Resources. **Ludovico Nicotina:** Writing – review & editing, Supervision, Resources. **Arno Hilberts:** Writing – review & editing, Supervision, Resources. **Christopher C. Pain:** Writing – review & editing, Writing – original draft, Software, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to acknowledge the following EPSRC grants: WavE-Suite, "New Generation Modelling Suite for the Survivability of Wave Energy Convertors in Marine Environments" (EP/V040235/1); the PREMIERE programme grant, "AI to enhance manufacturing, energy, and healthcare" (EP/T000414/1); INHALE, "Health assessment across biological length scales" (EP/T003189/1); AI-Respire, "AI for personalised respiratory health and pollution (EP/Y018680/1); ECO-AI, "Enabling CO_2 capture and storage using AI" (EP/Y005732/1). Support for CEH from Imperial-X's Eric and Wendy Schmidt Centre for AI in Science (a Schmidt Sciences programme) is gratefully acknowledged.

Appendix A. Non-linear Petrov–Galerkin method for the velocity components and the free surface height field

The approach outlined here finds the minimum of three diffusion coefficients, calculated for each field, from non-linear Petrov–Galerkin methods as described in Donéa (2003, page 185, final two equations). We choose the minimum in order to affect the discretisation as little as possible. Suppose that $\boldsymbol{\Psi}^{n+\frac{1}{2}}$ is either a velocity component (either $\boldsymbol{u}^{n+\frac{1}{2}}$ or $\boldsymbol{v}^{n+\frac{1}{2}}$) or the depth field $\boldsymbol{\eta}^{n+\frac{1}{2}}$. Defining $\boldsymbol{\Psi}_{\boldsymbol{x}}^{n+\frac{1}{2}} = \boldsymbol{f}(\boldsymbol{\Psi}^{n+\frac{1}{2}}; \mathbf{w}_{\mathbf{x}})$ and $\boldsymbol{\Psi}_{\boldsymbol{y}}^{n+\frac{1}{2}} = \boldsymbol{f}(\boldsymbol{\Psi}^{n+\frac{1}{2}}; \mathbf{w}_{\mathbf{y}})$, the first diffusion coefficient is calculated as follows:

$$\boldsymbol{k}_{\boldsymbol{\Psi}\boldsymbol{abs}}^{n+\frac{1}{2}} = \alpha_{kabs} |\boldsymbol{a}_{\boldsymbol{\Psi}}^{n+\frac{1}{2}}| \odot \boldsymbol{\mathcal{H}}_{\boldsymbol{\Psi}}{}^{n+\frac{1}{2}} \oslash (\epsilon_k \mathbf{1} + \frac{1}{2}(|\boldsymbol{\Psi}_{\boldsymbol{x}}^{n+\frac{1}{2}}| + |\boldsymbol{\Psi}_{\boldsymbol{y}}^{n+\frac{1}{2}}|)), \quad \text{(A.1)}$$

$$\text{with} \quad \boldsymbol{\mathcal{H}}_{\boldsymbol{\Psi}}{}^{n+\frac{1}{2}} = (\Delta x |\boldsymbol{\Psi}_{\boldsymbol{x}}^{n+\frac{1}{2}}| + \Delta y |\boldsymbol{\Psi}_{\boldsymbol{y}}^{n+\frac{1}{2}}|) \oslash (\epsilon_k \mathbf{1} + |\boldsymbol{\Psi}_{\boldsymbol{x}}^{n+\frac{1}{2}}| + |\boldsymbol{\Psi}_{\boldsymbol{y}}^{n+\frac{1}{2}}|), \quad \text{(A.2)}$$

where $|\cdot|$ indicates that the modulus is applied to all entries in the tensor and $\boldsymbol{a}_{\boldsymbol{\Psi}}^{n+\frac{1}{2}}$ is a residual or error associated with the advection terms. Here, for simplicity, a mixed mass approach is used,

$$\boldsymbol{a}_{\boldsymbol{\Psi}}^{n+\frac{1}{2}} = \alpha_a \boldsymbol{f}\left(\boldsymbol{u}^{n+\frac{1}{2}} \odot \boldsymbol{f}(\boldsymbol{\Psi}^{n+\frac{1}{2}}; \mathbf{w}_{\mathbf{x}}) + \boldsymbol{v}^{n+\frac{1}{2}} \odot \boldsymbol{f}(\boldsymbol{\Psi}^{n+\frac{1}{2}}; \mathbf{w}_{\mathbf{y}});\ m_l{}^{-1}\mathbf{w}_{\mathbf{m}} - \mathbf{I}\right), \quad \text{(A.3)}$$

where $m_l = \Delta x \Delta y$ is the lumped mass and thus $m_l^{-1} = \frac{1}{\Delta x \Delta y}$, $\mathbf{w}_{\mathbf{m}}$ is the consistent mass filter associated with the finite element matrices and $\mathbf{I}$ is the identity filter (on multiplication of a tensor results in the same vector). If the grid sizes in x and y are equal ($\Delta x = \Delta y$) and constant, Eq. (A.2) can be simply replaced with $\boldsymbol{\mathcal{H}}_{\boldsymbol{\Psi}}{}^{n+\frac{1}{2}} = \Delta x \mathbf{1}$ in which $\mathbf{1}$ is the tensor containing all ones. To prevent the diffusion coefficient from becoming too large, ϵ_k is set to a small value. Here, we choose $\epsilon_k = 10^{-7}$. The second diffusion coefficient is given by

$$\boldsymbol{k}_{\boldsymbol{\Psi}\boldsymbol{square}}^{n+\frac{1}{2}} = \alpha_{ksquare}(\boldsymbol{a}_{\boldsymbol{\Psi}}^{n+\frac{1}{2}} \odot \boldsymbol{a}_{\boldsymbol{\Psi}}^{n+\frac{1}{2}}) \odot \boldsymbol{\mathcal{H}}_{\boldsymbol{\Psi}}{}^{n+\frac{1}{2}} \oslash \left(\epsilon_k \mathbf{1} + \frac{1}{2}(|\boldsymbol{u}^{n+\frac{1}{2}} \odot \boldsymbol{\Psi}_{\boldsymbol{x}}^{n+\frac{1}{2}}| + |\boldsymbol{v}^{n+\frac{1}{2}} \odot \boldsymbol{\Psi}_{\boldsymbol{y}}^{n+\frac{1}{2}}|) \times (\boldsymbol{\Psi}_{\boldsymbol{x}}{}^{n+\frac{1}{2}} \odot \boldsymbol{\Psi}_{\boldsymbol{x}}{}^{n+\frac{1}{2}} + \boldsymbol{\Psi}_{\boldsymbol{y}}{}^{n+\frac{1}{2}} \odot \boldsymbol{\Psi}_{\boldsymbol{y}}{}^{n+\frac{1}{2}})\right). \quad \text{(A.4)}$$

The coefficients α_{kabs} and $\alpha_{ksquare}$ are defined as $\alpha_{kabs} = \frac{1}{8\times 2^{\mathrm{p}}}$ and $\alpha_{ksquare} = \frac{8}{2^{\mathrm{p}}}$ respectively, where p is the polynomial order of the finite element expansion, i.e., p = 1 for linear 3 × 3 filters, p = 2 for quadratic 5 × 5 filters and so on. The coefficient α_a is determined heuristically and was set to the value 2 here. The diffusion coefficient

Table B.3
A diagram showing how stages 1, 2 and 3 (resulting from segregating the coupled equations), fit within the predictor–corrector method for time level $n+1$. Here, we show two corrector steps. The variables on the left are what is being solved for and the variables on the right are the best guess used in the calculations. Solutions at the previous time levels are also needed (h^n, u^n and v^n). The figures in parentheses indicate the iteration index.

			Solve for		Based on	
			h	q	h	q
Time level $n+1$	Predictor	Stage 1		$\tilde{q}^{n+1,(1)}$	$h^{n+1,(0)}$	$q^{n+1,(0)}$
		Stage 2	$h^{n+1,(1)}$		$h^{n+1,(0)}$	$\tilde{q}^{n+1,(1)}$
		Stage 3		$q^{n+1,(1)}$	$h^{n+1,(1)}$	$\tilde{q}^{n+1,(1)}$
	Corrector	Stage 1		$\tilde{q}^{n+1,(2)}$	$h^{n+1,(1)}$	$q^{n+1,(1)}$
		Stage 2	$h^{n+1,(2)}$		$h^{n+1,(1)}$	$\tilde{q}^{n+1,(2)}$
		Stage 3		$q^{n+1,(2)}$	$h^{n+1,(2)}$	$\tilde{q}^{n+1,(2)}$
	Corrector	Stage 1		$\tilde{q}^{n+1,(3)}$	$h^{n+1,(2)}$	$q^{n+1,(2)}$
		Stage 2	$h^{n+1,(3)}$		$h^{n+1,(2)}$	$\tilde{q}^{n+1,(3)}$
		Stage 3		$q^{n+1,(3)}$	$h^{n+1,(3)}$	$\tilde{q}^{n+1,(3)}$

should not exceed the stability criteria for the explicit treatment of diffusion (Wood, 1990), hence, the third diffusion coefficient is

$$\boldsymbol{k}_{\boldsymbol{max}}^{n+\frac{1}{2}} = \frac{\Delta x \Delta y}{4\Delta t}\mathbf{1}, \tag{A.5}$$

which has the same value for all three fields. Thus the diffusion coefficient for each field is given by:

$$\boldsymbol{k}_{\boldsymbol{\Psi}}^{n+\frac{1}{2}} = \rho_\Psi \min\{\boldsymbol{k}_{\boldsymbol{max}}^{n+\frac{1}{2}}, \boldsymbol{k}_{\boldsymbol{\Psi abs}}^{n+\frac{1}{2}}, \boldsymbol{k}_{\boldsymbol{\Psi square}}^{n+\frac{1}{2}}\} + \mu_\Psi \mathbf{1}, \tag{A.6}$$

$$\text{or} \quad k_{\Psi\, i,j}^{n+\frac{1}{2}} = \rho_\Psi \min\{k_{max\, i,j}^{n+\frac{1}{2}}, k_{\Psi abs\, i,j}^{n+\frac{1}{2}}, k_{\Psi square\, i,j}^{n+\frac{1}{2}}\} + \mu_\Psi\,, \tag{A.7}$$

where i and j identify the node. Note the dynamic viscosity or diffusivity could define μ_Ψ (although here $\mu_\Psi = 0$ is used). Also, when $\boldsymbol{\Psi}^{n+\frac{1}{2}}$ is one of the velocity components then $\rho_\Psi = \frac{1}{g}$, but when $\boldsymbol{\Psi}^{n+\frac{1}{2}}$ represents the water depth, $\rho_\Psi = 1$.

Appendix B. Derivation of corrections for velocity and free surface height

During one iteration of the solution procedure for one time level, three equations are solved. First, we will solve for velocity (stage 1). From this, we solve for the free surface height (stage 2) and we then solve for a correction to velocity based on the latest value of free surface height.

In this section we derive the equations used in stages 2 and 3, using notation associated with the time discretised system, because we form the equations in the continuum and then discretise in space (see main text for the spatial discretisation). Table B.3 shows the notation used in the iterative process of the predictor–corrector algorithm, indicating the variables solved for at a particular stage and the variables that are used in the calculation. The tilde sign over the velocity indicates a first approximation to this variable (calculated in stage 1), which will be corrected (in stage 3) based on the updated free surface height, at which point the tilde sign is omitted.

Suppose we have the solution for time level n and wish to solve for the next time level. Assuming a general predictor–corrector method with multiple iterations, PE(CE)^M where $M > 1$, the variables at the beginning of a time level are initialised as follows:

$$u^{n+1,(0)} = u^n\,, \quad v^{n+1,(0)} = v^n\,, \quad h^{n+1,(0)} = h^n\,. \tag{B.1}$$

Suppose also that we have applied $m-1$ iterations thus far and wish to solve for the mth iteration. To obtain an approximation to the velocity (stage 1), the following equation is solved

$$\rho_g\left(\frac{\tilde{\boldsymbol{q}}^{n+1,(m)} - \boldsymbol{q}^n}{\Delta t} + u^{n+\frac{1}{2}}\frac{\partial \boldsymbol{q}^{n+\frac{1}{2}}}{\partial x} + v^{n+\frac{1}{2}}\frac{\partial \boldsymbol{q}^{n+\frac{1}{2}}}{\partial y}\right) + \sigma_q^{n+1,(m-1)}\tilde{\boldsymbol{q}}^{n+1,(m)} = -\frac{1}{2}\nabla(h^n + h^{n+1,(m-1)})\,, \tag{B.2}$$

where $u^{n+\frac{1}{2}} = \frac{1}{2}\left(u^n + u^{n+1,(m-1)}\right)$, similarly for $v^{n+\frac{1}{2}}$ and $\boldsymbol{q}^{n+\frac{1}{2}} \equiv (u^{n+\frac{1}{2}}, v^{n+\frac{1}{2}})^T$ and in which $\sigma_q^{n+1,(m-1)} = \sigma_q(q^{n+1,(m-1)}, \eta^{n+1,(m-1)})$. Note that the term involving bed friction is treated implicitly. At the end of the iteration, the free surface height (stage 2) and corrected velocity (stage 3) will satisfy

$$\rho_g\left(\frac{\boldsymbol{q}^{n+1,(m)} - \boldsymbol{q}^n}{\Delta t} + u^{n+\frac{1}{2}}\frac{\partial \boldsymbol{q}^{n+\frac{1}{2}}}{\partial x} + v^{n+\frac{1}{2}}\frac{\partial \boldsymbol{q}^{n+\frac{1}{2}}}{\partial y}\right) + \sigma_q^{n+1,(m-1)}\tilde{\boldsymbol{q}}^{n+1,(m)} = -\frac{1}{2}\nabla(h^n + h^{n+1,(m)})\,, \tag{B.3}$$

where the advection terms and σ_q^{n+1} are calculated as in Eq. (B.2), meaning that the term involving bed friction is no longer fully implicit. By subtracting Eq. (B.2) from (B.3), we obtain the velocity correction equation:

$$\rho_g\left(\frac{\boldsymbol{q}^{n+1,(m)} - \tilde{\boldsymbol{q}}^{n+1,(m)}}{\Delta t}\right) = -\frac{1}{2}\nabla(h^{n+1,(m)} - h^{n+1,(m-1)})\,, \tag{B.4}$$

which is solved in stage 3. The divergence of this equation is:

$$\rho_g\left(\frac{\nabla\cdot\boldsymbol{q}^{n+1,(m)} - \nabla\cdot\tilde{\boldsymbol{q}}^{n+1,(m)}}{\Delta t}\right) = -\frac{1}{2}\nabla\cdot\nabla(h^{n+1,(m)} - h^{n+1,(m-1)}). \tag{B.5}$$

Eq. (4), which describes the free surface height, can be expanded as follows:

$$\frac{h^{n+1,(m)} - h^n}{\Delta t} + \frac{1}{2}\eta^{n+\frac{1}{2}}\nabla\cdot\boldsymbol{q}^{n+1,(m)} + \frac{1}{2}\eta^{n+\frac{1}{2}}\nabla\cdot\boldsymbol{q}^n + (\nabla\eta^{n+\frac{1}{2}})\cdot\tilde{\boldsymbol{q}}^{n+\frac{1}{2}} - s_h^{n+1} = 0\,, \tag{B.6}$$

where $\tilde{u}^{n+\frac{1}{2}} = \frac{1}{2}\left(u^n + \tilde{u}^{n+1,(m)}\right)$, similarly for $v^{n+\frac{1}{2}}$. Upon rearranging, Eq. (B.6) becomes:

$$\nabla\cdot\boldsymbol{q}^{n+1,(m)} = -\frac{1}{\frac{1}{2}\eta^{n+\frac{1}{2}}}\left(\frac{h^{n+1,(m)} - h^n}{\Delta t} + \frac{1}{2}\eta^{n+\frac{1}{2}}\nabla\cdot\boldsymbol{q}^n + (\nabla\eta^{n+\frac{1}{2}})\cdot\tilde{\boldsymbol{q}}^{n+\frac{1}{2}} - s_h^{n+1}\right). \tag{B.7}$$

Substituting the expression for $\nabla\cdot\boldsymbol{q}^{n+1,(m)}$ from Eq. (B.7) into Eq. (B.5), we obtain:

$$-\frac{1}{2}\nabla\cdot\nabla(h^{n+1,(m)} - h^{n+1,(m-1)}) = -\frac{\rho_g}{\frac{1}{2}\Delta t\eta^{n+\frac{1}{2}}}\left(\frac{h^{n+1,(m)} - h^n}{\Delta t} + \frac{1}{2}\eta^{n+\frac{1}{2}}\nabla\cdot(\boldsymbol{q}^n + \tilde{\boldsymbol{q}}^{n+1,(m)}) + (\nabla\eta^{n+\frac{1}{2}})\cdot\tilde{\boldsymbol{q}}^{n+\frac{1}{2},(m)} - s_h^{n+1}\right). \tag{B.8}$$

By using $\tilde{\boldsymbol{q}}^{n+\frac{1}{2},(m)} = \frac{1}{2}(\boldsymbol{q}^n + \tilde{\boldsymbol{q}}^{n+1,(m)})$ and defining $\Delta h^{n+1,(m)} := h^{n+1,(m)} - h^{n+1,(m-1)}$:

$$-\frac{1}{2}\nabla\cdot\nabla\Delta h^{n+1,(m)} = -\frac{\rho_g}{\frac{1}{2}\Delta t\eta^{n+\frac{1}{2}}} \times\left(\frac{h^{n+1,(m)} - h^n}{\Delta t} + \nabla\cdot(\eta^{n+\frac{1}{2}}\tilde{\boldsymbol{q}}^{n+\frac{1}{2},(m)}) - s_h^{n+1}\right). \tag{B.9}$$

Rearranging and subtracting a term involving $h^{n+1,(m-1)}$ to both sides of the equation results in:

$$-\frac{1}{2}\nabla\cdot\nabla\Delta h^{n+1,(m)} + \frac{\rho_g}{\frac{1}{2}\Delta t\eta^{n+\frac{1}{2}}}\left(\frac{h^{n+1,(m)} - h^{n+1,(m-1)}}{\Delta t}\right) = -\frac{\rho_g}{\frac{1}{2}\Delta t\eta^{n+\frac{1}{2}}}\left(\frac{h^{n+1,(m-1)} - h^n}{\Delta t} + \nabla\cdot(\eta^{n+\frac{1}{2}}\tilde{\boldsymbol{q}}^{n+\frac{1}{2},(m)}) - s_h^{n+1}\right). \tag{B.10}$$

Thus, simplifying this equation, the final wave equation is obtained,

$$\left(-\nabla\cdot\nabla + \frac{4\rho_g}{(\Delta t)^2\eta^{n+\frac{1}{2}}}\right)\Delta h^{n+1,(m)}$$

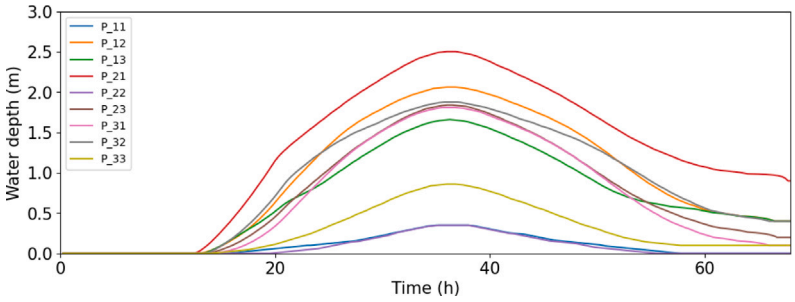


Fig. C.23. Numerical prediction of water depth at 9 ConvFEM nodes nearest to the bus depot in the Carlisle test case (using the quadratic ConvFEM discretisation).

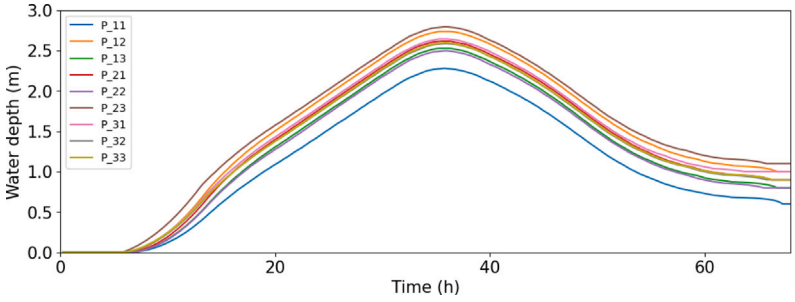


Fig. C.24. Numerical prediction of water depth at 9 ConvFEM nodes nearest to Water mark 5 in the Carlisle test case (using the quadratic ConvFEM discretisation).

$$= -\frac{4\rho_g}{\Delta t \eta^{n+\frac{1}{2}}}\left(\frac{h^{n+1,(m-1)} - h^n}{\Delta t} + \nabla\cdot(\eta^{n+\frac{1}{2}}\tilde{\boldsymbol{q}}^{n+\frac{1}{2},(m)}) - s_h^{n+1}\right), \qquad \text{(B.11)}$$

the solution of which can be used to update the free surface height: $h^{n+1,(m)} = h^{n+1,(m-1)} + \Delta h^{n+1,(m)}$.

Thus, given an approximation to the velocity calculated at the beginning of an iteration (stage 1), the free surface height can be calculated by solving Eq. (B.11). This value can be used to calculate the correction for the velocity given by Eq. (B.4).

Appendix C. Uncertainty in predicting water depth readings in carlisle flooding

A 3×3 array of sensors comprising of the 9 nearest ConvFEM nodes (with a uniform spacing of $\Delta x = 5\,\text{m}$) are deployed at two locations, to investigate the uncertainty in water depth readings. The locations considered are the bus depot, where there is a steep bathymetry gradient, and Water mark 5, where the gradient is minimal. The variation in water depth predicted at the array of sensors in the vicinity of the bus depot (Fig. C.23) is much larger than that seen at Water mark 5 (Fig. C.24). At the bus depot, the predictions can vary by up to 2.5 m whereas at water mark 5, the variation in depth is about 0.5 m. Thus, the substantial difference (over 1 m) between the predicted water depth of the different models shown in Fig. 19 is attributed to the steep bathymetry.

Data availability

NN4PDEs is implemented using PyTorch and is available from GitHubRepository.

References

Akbari, M., Pirzadeh, B., 2023. Preserving stationary discontinuities in two-layer shallow water equations with a novel well-balanced approach. J. Hydroinformatics 25 (5), 1979–2003.

Anderson, J.R., Ihme, M., Wang, Q., Chen, Y.-F., 2022. A TensorFlow simulation framework for scientific computing of fluid flows on tensor processing units. Comput. Phys. Comm. 274, 108292.

Ascher, U.M., Petzold, L.R., 1998. Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. SIAM.

Bernetti, R., Titarev, V.A., Toro, E.F., 2008. Exact solution of the Riemann problem for the shallow water equations with discontinuous bottom geometry. J. Comput. Phys. 227 (6), 3212–3243.

Bunya, S., Kubatko, E.J., Westerink, J.J., Dawson, C., 2009. A wetting and drying treatment for the Runge–Kutta discontinuous Galerkin solution to the shallow water equations. Comput. Methods Appl. Mech. Engrg. 198 (17–20), 1548–1562.

Butcher, J.C., 2016. Numerical Methods for Ordinary Differential Equations, third ed. John Wiley & Sons.

Buwalda, F.J., De Goede, E., Knepflé, M., Vuik, C., 2023. Comparison of an explicit and implicit time integration method on gpus for shallow water flows on structured grids. Water 15 (6), 1165.

Chen, B., Heaney, C.E., Gomes, J.L.M.A., Matar, O.K., Pain, C.C., 2024a. Solving the discretised multiphase flow equations with interface capturing on structured grids using machine learning libraries. Comput. Methods Appl. Mech. Engrg. 426, 116974. http://dx.doi.org/10.1016/j.cma.2024.116974.

Chen, B., Heaney, C.E., Pain, C.C., 2024b. Using AI libraries for incompressible computational fluid dynamics. http://dx.doi.org/10.48550/arXiv.2402.17913, ArXiv Preprint arxiv:2402.17913.

Dobbelaere, T., Muller, E.M., Gramer, L.J., Holstein, D.M., Hanert, E., 2020. Coupled epidemio-hydrodynamic modeling to understand the spread of a deadly coral disease in florida. Front. Mar. Sci. 7, 591881.

Donéa, J., 2003. Finite Element Methods for Flow Problems. John Wiley & Sons, http://dx.doi.org/10.1002/0470013826.

Fernández-Pato, J., Morales-Hernández, M., García-Navarro, P., 2018. Implicit finite volume simulation of 2D shallow water flows in flexible meshes. Comput. Methods Appl. Mech. Engrg. 328, 1–25.

Henonin, J., Russo, B., Mark, O., Gourbesville, P., 2013. Real-time urban flood forecasting and modelling — A state of the art. J. Hydroinformatics 15 (3), 717–736, URL:.

Innovyze, 2019. InfoWorks ICM. Retrieved from https://www.innovyze.com/.

Jamali, B., Bach, P.M., Deletic, A., 2020. Rainwater harvesting for urban flood management — An integrated modelling framework. Water Res. 171, 115372, URL: https://d10.1016/j.watres.2019.115372.

Kesserwani, G., Sharifian, M.K., 2023. (Multi) wavelet-based godunov-type simulators of flood inundation: Static versus dynamic adaptivity. Adv. Water Resour. 171, 104357. http://dx.doi.org/10.1016/j.advwatres.2022.104357.

Lacasta, A., Morales-Hernández, M., Murillo, J., García-Navarro, P., 2015. GPU implementation of the 2D shallow water equations for the simulation of rainfall/runoff events. Environ. Earth Sci. 74, 7295–7305.

Lee, H., 2021. Discontinuous Galerkin discretization of shallow water equations in implicit primal formulations for turbulent stresses. J. Mech. Sci. Technol. 35 (6), 2471–2479.

LeFloch, P.G., Thanh, M.D., 2011. A godunov-type method for the shallow water equations with discontinuous topography in the resonant regime. J. Comput. Phys. 230 (20), 7631–7660.

Li, L., Xiang, J., Chen, B., Heaney, C.E., Dargaville, S., Pain, C.C., 2024. Implementing the discontinuous-Galerkin finite element method using graph neural networks. Neural Netw. 107061. http://dx.doi.org/10.1016/j.neunet.2024.107061.

Liang, Q., Marche, F., 2009. Numerical resolution of well-balanced Shallow water equations with complex source terms. Adv. Water Resour. 32 (6), 873–884.

Löwe, R., Urich, C., Sto. Domingo, N., Mark, O., Deletic, A., Arnbjerg-Nielsen, K., 2017. Assessment of urban pluvial flood risk and efficiency of adaptation options through simulations — A new generation of urban planning tools. J. Hydrol. 550, 355–367, URL: https://doi.org/10.1016/j.jhydrol.2017.05.009.

Lozovskiy, A., Farthing, M., Kees, C., 2017. Evaluation of Galerkin and Petrov-Galerkin model reduction for finite element approximations of the shallow water equations. Comput. Methods Appl. Mech. Engrg. 318, 537–571.

Lundgren, L., Mattsson, K., 2020. An efficient finite difference method for the shallow water equations. J. Comput. Phys. 422, 109784.

Madsen, P.A., Simonsen, H.J., Pan, C.-H., 2005. Numerical simulation of tidal bores and hydraulic jumps. Coast. Eng. 52 (5), 409–433.

Medeiros, S.C., Hagen, S.C., 2013. Review of wetting and drying algorithms for numerical tidal flow models. Internat. J. Numer. Methods Fluids 71 (4), 473–487.

Minatti, L., Faggioli, L., 2023. The exact Riemann solver to the Shallow water equations for natural channels with bottom steps. Comput. & Fluids 254, 105789.

Miura, H., Skamarock, W.C., 2013. An upwind-biased transport scheme using a quadratic reconstruction on spherical icosahedral grids. Mon. Weather Rev. 141 (2), 832–847.

Morales-Hernández, M., Sharif, M.B., Kalyanapu, A., Ghafoor, S.K., Dullo, T.T., Gangrade, S., Kao, S.-C., Norman, M.R., Evans, K.J., 2021. TRITON: A multi-GPU open source 2D hydrodynamic flood model. Environ. Model. Softw. 141, 105034.

Néelz, S., Pender, G., 2013. Benchmarking the latest generation of 2D hydraulic modelling packages (SC120002). p. 40, Retrieved from Environment Agency, Section 4.5.

Olaf Ronneberger, P.F., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention. MICCAI, In: LNCS, vol. 9351, Springer, pp. 234–241. http://dx.doi.org/10.48550/arXiv.1505.04597.

Pavlidis, D., Gomes, J.L.M.A., Xie, Z., Percival, J.R., Pain, C.C., Matar, O.K., 2016. Compressive advection and multi-component methods for interface-capturing. Internat. J. Numer. Methods Fluids 80 (4), 256–282. http://dx.doi.org/10.1002/fld.4078.

Phillips, T.R.F., 2022. Neural network transport solver. https://github.com/trfphillips/Neural-Network-Transport-Solver.

Phillips, T.R.F., Heaney, C.E., Chen, B., Buchan, A.G., Pain, C.C., 2023a. Solving the discretised Boltzmann transport equations using neural networks: Applications in neutron transport. http://dx.doi.org/10.48550/arXiv.2301.09991, ArXiv Preprint arXiv:2301.09991.

Phillips, T.R.F., Heaney, C.E., Chen, B., Buchan, A.G., Pain, C.C., 2023b. Solving the discretised neutron diffusion equations using neural networks. Internat. J. Numer. Methods Engrg. 124 (21), 4659–4686. http://dx.doi.org/10.1002/nme.7321.

Sharifian, M.K., Kesserwani, G., Chowdhury, A.A., Neal, J., Bates, P., 2023. LISFLOOD-FP 8.1: new GPU-accelerated solvers for faster fluvial/pluvial flood simulations. Geosci. Model. Dev. Discuss. 16 (9), 2391–2413, URL: https://doi.org/10.5194/gmd-16-2391-2023.

Shaw, J., Kesserwani, G., Neal, J., Bates, P., Sharifian, M.K., 2021. LISFLOOD-FP 8.0: the new discontinuous Galerkin shallow-water solver for multi-core CPUs and GPUs. Geosci. Model. Dev. 14 (6), 3577–3602. http://dx.doi.org/10.5194/gmd-14-3577-2021.

Skoula, Z., Borthwick, A., Moutzouris, C., 2006. Godunov-type solution of the shallow water equations on adaptive unstructured triangular grids. Int. J. Comput. Fluid Dyn. 20 (9), 621–636.

Smith, L.S., Liang, Q., 2013. Towards a generalised GPU/CPU shallow-flow modelling tool. Comput. & Fluids 88, 334–343.

Sotiropoulos, F., Abdallah, S., 1991. The discrete continuity equation in primitive variable solutions of incompressible flow. J. Comput. Phys. 95 (1), 212–227. http://dx.doi.org/10.1016/0021-9991(91)90260-R.

Spiekermann, R., Kienberger, S., Norton, J., Briones, F., Weichselgartner, J., 2015. The disaster-knowledge matrix–reframing and evaluating the knowledge challenges in disaster risk reduction. Int. J. Disaster Risk Reduct. 13, 96–108.

Teng, J., Jakeman, A.J., Vaze, J., Croke, B.F., Dutta, D., Kim, S., 2017. Flood inundation modelling: A review of methods, recent advances and uncertainty analysis. Environ. Model. Softw. 90, 201–216.

2022. TRITON:Two-Dimensional Runoff Inundation Toolkit for Operational Needs. Oak Ridge National Lab., Accessed 01-December-2023.

Tuflow, 2023. TUFLOW user manual – build 2016-03-AA . URL: https://wiki.tuflow.com/w/index.php?title=Main_Page&oldid=34793. (Accessed 28 November 2023).

Vanzo, D., Siviglia, A., Toro, E.F., 2016. Pollutant transport by shallow water equations on unstructured meshes: Hyperbolization of the model and numerical solution via a novel flux splitting scheme. J. Comput. Phys. 321, 1–20. http://dx.doi.org/10.1016/j.jcp.2016.05.023.

Wang, Z.-L., Geng, Y.-F., 2013. Two-dimensional shallow water equations with porosity and their numerical scheme on unstructured grids. Water Sci. Eng. 6 (1), 91–105.

Wang, Q., Ihme, M., Chen, Y.-F., Anderson, J., 2022. A TensorFlow simulation framework for scientific computing of fluid flows on tensor processing units. Comput. Phys. Comm. 274, 108292. http://dx.doi.org/10.1016/j.cpc.2022.108292.

Wood, W.L., 1990. Practical time-stepping schemes. Oxford Applied Mathematics and Computing Science Series, Clarendon, Oxford.

Xia, X., Liang, Q., 2016. A GPU-accelerated smoothed particle hydrodynamics (SPH) model for the shallow water equations. Environ. Model. Softw. 75, 28–43.

Yalcin, E., 2020. Assessing the impact of topography and land cover data resolutions on two-dimensional HEC-RAS hydrodynamic model simulations for urban flood hazard analysis. Nat. Hazards 101 (3), 995–1017.

Zanardo, S., Salinas, J.L., 2022. An introduction to flood modeling for catastrophe risk management. Wiley Interdiscip. Rev.: Water 9 (1), e1568.

Zeiner, T., Fischlschweiger, M., 2023. Chapter 15 — Diffusion and transport through nanoscale polymer-based coatings. In: Thomas, S., George, J.S. (Eds.), Polymer-Based Nanoscale Materials for Surface Coatings. Elsevier, pp. 291–321. http://dx.doi.org/10.1016/B978-0-32-390778-1.00022-0, URL: https://www.sciencedirect.com/science/article/pii/B9780323907781000220.

Zhao, X.-Z., Xu, T.-Y., Ye, Z.-T., Liu, W.-J., 2020. A TensorFlow-based new high-performance computational framework for CFD. J. Hydrodyn. 32 (4), 735–746. http://dx.doi.org/10.1007/s42241-020-0050-0.