POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Variational data assimilation for porous shallow water equations

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Marco Spadoni, 232014

Advisor:
Prof. Anna Scotti

Co-advisors:
Antoine Rousseau
Pascal Finaud-Guyot

Academic year:
2024-2025

Abstract: This thesis has been carried out at the Inria branch in Montpellier within the LEMON team, an interdisciplinary team working on the design and implementation of accurate and computationally inexpensive models for natural processes occurring in the littoral area [1]. Urban-scale flood models are increasingly being studied in the context of risk prevention, and, to this end, for several years the LEMON team has been developing the so-called porosity models for the shallow water equations to provide rapid, large-scale simulations of these phenomena. Porous shallow water equations extend the classical shallow water model by introducing porosity to represent subgrid-scale effects caused by obstacles such as buildings and vegetation in floodplains. This work explores the use of variational data assimilation to improve the knowledge on the spatial distribution of porosity, and, ultimately, to increase accuracy in the prediction of the flood flow. Variational data assimilation is implemented using a gradient descent optimization approach, which efficiently computes the gradient through the adjoint problem. We present both the forward and adjoint formulations, along with their one-dimensional numerical solvers. Numerical experiments demonstrate the effectiveness of this approach for improving porosity estimation. We emphasize the importance of addressing the well-posedness of the assimilation problem to ensure reliable results.

Key-words: variational data assimilation, porosity, shallow water equations, adjoint problem

1. Introduction

Data assimilation refers to a set of mathematical and computational techniques that merge observational data with numerical models to produce an improved description of a dynamical system. The aim of data assimilation is often to compute the best possible estimate of the model state. Alternatively, we use data assimilation to estimate the model parameters or infer the best characterization of the model forcing terms or controls [2]. The notion of data assimilation was first proposed in the field of numerical weather prediction (NWP), however, its mathematical formulation originates from Bayesian inference (e.g. particle filter [3]), control theory (e.g. Kalman filter [4]), and variational calculus [5]. Data assimilation has now spread to many different research fields including the geosciences, geomechanics, and hydrology [6]. Among the various approaches to data assimilation, in this thesis we focus on the variational method. Indeed, variational data assimilation, being formulated as an optimization problem, can easily accommodate models governed by partial differential equations (PDEs) as constraint by making use of Lagrange multipliers [7]. Variational data assimilation is a major component

of operational systems and scientific studies for the understanding, modeling, forecasting and reconstruction of earth systems [8]. For instance, it is nowadays used by the European Centre for Medium-Range Weather Forecasts (ECMWF) to retrieve initial conditions for weather forecast models [9, 10]. The aim of this thesis is to show that this technique can be applied to improve the application of porous shallow water equations to fast and reliable large-scale simulations of flood events.

Variational data assimilation aims to minimize a cost function defined to quantify the mismatch between the model output and the observations, by adjusting some control variables (states or parameters of the system). In the context of this work the dynamical model is governed by the porous shallow water equations, while the control variable is represented by the porosity distribution in space. Thanks to the Lagrangian formulation we are able to compute the gradient of the cost function with respect to the control variable (namely with respect to porosity). This gradient is then used in a gradient descent algorithm to retrieve the porosity distribution that best aligns the model solution with the observations.

This thesis is structured as follows: in Section 2 we introduce the single porosity shallow water model and we justify its adoption through a test case. In Section 3 we present the numerical discretization for the shallow water equations and we validate the one-dimensional solver. In Section 4 we present the variational data assimilation procedure, first for a simple problem governed by an ordinary differential equation (ODE), then for the one-dimensional porous shallow water equations. In particular, in Section 4.3 we derive the adjoint model for the porous shallow water equations extending the derivation of the adjoint model for the classical shallow water equations performed by Sanders and Katopodes [11], while in Section 4.4 we introduce the numerical solver for the adjoint problem. Finally, in Section 5, we test the data assimilation algorithm by performing some numerical experiments. Section 6 is dedicated to some concluding remarks. Notice that, for simplicity in the technical implementation, we limit our study to the one-dimensional case, however all the methods presented can be naturally extended to the two-dimensional case.

2. Porous shallow water equations

Shallow water models with porosity are designed to simulate urban flood flows orders of magnitude faster than classical shallow-water models thanks to a relatively coarse grid and large time step. This enables flood hazard mapping over far greater spatial extents compared to what can be achieved with classical shallow-water models [12]. Thanks to terrestrial LIDAR data, it is nowadays possible to produce meshes for shallow water simulations whose cell size is in the scale of 10 cm [13]. However, it is not feasible to keep such a detailed scale for the whole floodplain. Indeed, commonly used shallow water equations solver adopts an explicit time stepping and are thus constrained by the well known Courant–Friedrichs–Lewy (CFL) condition. If we consider the two dimensional case, being Δx the discretization length in space, the number of cells N_c scales as Δx^{-2}, while the number of time steps N_t scales as Δx^{-1} due to the CFL condition. The computational cost C of the shallow water solver scales as the product of N_c and N_t. As a result we get $C \sim \frac{1}{\Delta x^3}$. For this reason, the porosity ϕ is introduced in the shallow water model to account for the presence of buildings, vegetation, and other structures in the floodplain, avoiding the need of refining the mesh to the fine scale of such obstacles. In other words, porosity models the obstacles present at the subgrid scale. Physically, porosity represents the fraction of the area available for flow. It ranges from 0 (solid obstruction) to 1 (fully open). Porosity ϕ can be defined in a porous media fashion by making use of a binary density function i, and of the concept of representative elementary volume (REV). Let $\Omega \subset \mathbb{R}^2$ be the domain representing the flood plain. Let $D_b \subset \Omega$ be a portion of the domain occupied by a solid obstacle. We can define $i : \Omega \to \{0, 1\}$ as follows:

$$i(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in D_b \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

Let $\Omega_0 \subset \Omega$ be a REV. We can define the porosity ϕ_0 in the REV as:

$$\phi_0 = \frac{1}{|\Omega_0|} \int_{\Omega_0} i \, d\boldsymbol{x} \tag{2}$$

Notice that the length scale of the REV should be an order of magnitude larger than the scale of the obstructions.

2.1. The single porosity model

The shallow water equations (SWE) are a set of well known hyperbolic partial differential equations that describe the flow of a thin layer of fluid, bounded from below by the bottom topography and from above by a free surface. In the SWE hydrostatic pressure is assumed, moreover vertical velocity and viscous stresses are neglected. The

SWE are derived by performing a depth-integration of the Navier-Stokes equations, as a consequence, being n the spatial dimension of the full Navier-Stokes model, the shallow water model will have a spatial dimension of $n-1$. Among the various porosity models available for SWE, in this work we adopt the single porosity model [14]. Other porosity models are, for instance, the depth dependent porosity model [15] and the dual integral porosity model [16]. The two-dimensional single-porosity SWE read:

$$\frac{\partial}{\partial t}(\phi \boldsymbol{U}) + \frac{\partial}{\partial x}(\phi \boldsymbol{F}) + \frac{\partial}{\partial y}(\phi \boldsymbol{G}) = \boldsymbol{S} \tag{3}$$

$$\text{with} \qquad \boldsymbol{U} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \ \boldsymbol{F} = \begin{pmatrix} hu \\ hu^2 + gh^2/2 \\ huv \end{pmatrix}, \ \boldsymbol{G} = \begin{pmatrix} hv \\ huv \\ hv^2 + gh^2/2 \end{pmatrix}, \ \boldsymbol{S} = \begin{pmatrix} 0 \\ S_{0,x} + S_{f,x} \\ S_{0,y} + S_{f,y} \end{pmatrix},$$

where g is the gravitational acceleration, h is the water depth, u and v are the two velocity components, $S_{0,x}$ and $S_{0,y}$ are the source terms arising from the bottom slopes and porosity variations in the x and y directions, $S_{f,x}$ and $S_{f,y}$ are the source terms arising from friction in the x and y directions, and ϕ is the porosity. Porosity is assumed to depend on the space coordinates only. The topographical source terms are given by

$$S_{0,x} = -\phi g h \frac{\partial z_b}{\partial x} + g \frac{h^2}{2} \frac{\partial \phi}{\partial x} \tag{4}$$

$$S_{0,y} = -\phi g h \frac{\partial z_b}{\partial y} + g \frac{h^2}{2} \frac{\partial \phi}{\partial y} \tag{5}$$

where z_b is the bed elevation with respect to a fixed reference level. The friction terms are given by [14]:

$$S_{f,x} = -\phi g h \frac{(u^2+v^2)^{1/2}}{K^2 h^{4/3}} u - \phi g h s_x (u^2+v^2)^{1/2} u \tag{6}$$

$$S_{f,y} = -\phi g h \frac{(u^2+v^2)^{1/2}}{K^2 h^{4/3}} v - \phi g h s_y (u^2+v^2)^{1/2} v \tag{7}$$

where K is the Strickler coefficient and s_x and s_y are the head loss coefficients accounting for the singular head losses due to the presence of obstacles in the x and y directions, respectively.
A possible use of the porous shallow water equations is the simulation of floods in the presence of urbanized areas that condition the behavior of the flow, but where the details of the flow within the urban areas are not of direct interest [14]. In this section we justify the usefulness of the porous shallow water equations by illustrating a numerical version of the Toce test case [17], a classical benchmark for flooding simulations consisting in physical experiments performed using a scale model of the Toce valley (Italy). The geometry of the problem is represented in Figure 1, the trapezoidal plan shape is 7m long and 3.5 to 5m wide depending on the location.

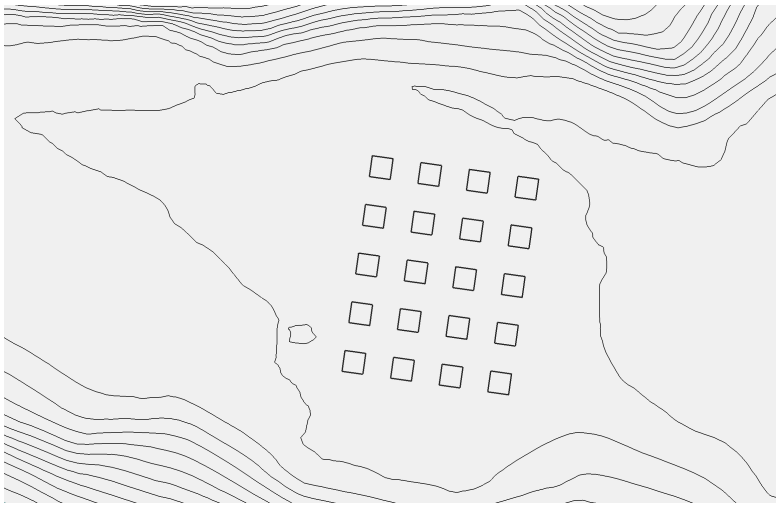

Figure 1: Toce test case: geometry with bed elevation isolines

The urban zone is represented by 20 aligned blocks in a rectangular area. We aim to show that we can replace a classical shallow water model making use of a refined grid, with a porous shallow water model based on a much coarser grid. The tradeoff is to lose the flow details inside the urban area.

We run three different shallow water simulations using the C++ software SW2D [18]. SW2D is finite-volume shallow water solver developed by the LEMON team that includes several porosity models. The three cases are the following:

- `Toce_CR`: we adopt the classical shallow water model (i.e. without porosity) with a fine mesh made of 22322 cells. This simulation will be the reference.
- `Toce_PC`: we adopt the single porosity model with a coarse mesh made of 4800 cells.
- `Toce_CC`: we adopt the classical shallow water model with a coarse mesh made of 4780 cells.

Figure 2 compares the coarse and fine meshes in the urban zone. Notice that when using the classical shallow water model the blocks are represented by impermeable boundaries. On the other hand, when using the porous model we do not provide a detailed discretization of the model geometry between the blocks. In fact in this case the presence of the blocks is just modeled as a subgrid scale property by setting the porosity in the urban area to be equal to 0.44. Porosity is set to 1 elsewhere. The Strickler coefficient K is set to 60 in the whole domain. Figure 3 reports the unit discharge $q = h\boldsymbol{u}$ injected at the upstream boundary, and the solution of

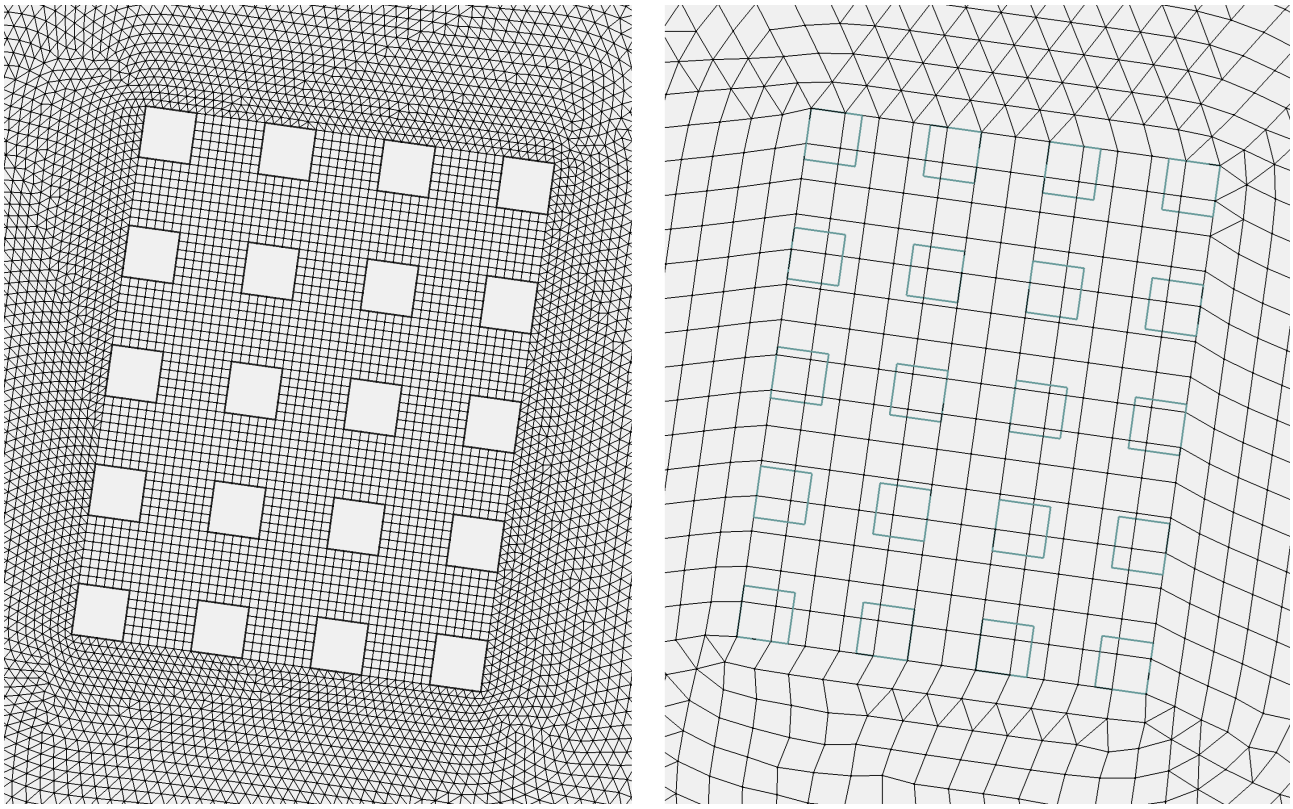

Figure 2: Toce test case: mesh detail in the urban zone. Fine mesh (left), coarse mesh (right). The blue lines in the coarse mesh represent the position of the buildings and are just for reference

the three different tests at time $t = 20s$. Figure 4 reports the water depth difference between the reference solution given by `Toce_CR` and the solution given by `Toce_PC` and `Toce_CC`, respectively. We notice that, when compared to the classical coarse model, the porosity model better captures the features of the flow upstream and downstream of the urban area. Of course we cannot expect to capture the flow behavior inside the urban area: this is a inevitable consequence of the choice of the model. The concept of porosity is in fact intended to represent the macroscopic properties of the geometry.

3. 1D-porous shallow water model

In this work we adopt a slightly different formulation of the SWE than the one presented in Section 2.1: first, we consider just the one-dimensional case. Moreover, instead of solving for h and u, we solve for the variables h and q, where $q = hu$. The 1D-SWE with porosity then read:

$$\begin{cases} \dfrac{\partial(\phi h)}{\partial t} + \dfrac{\partial(\phi q)}{\partial x} = 0 & \text{(8a)} \\ \dfrac{\partial(\phi q)}{\partial t} + \dfrac{\partial}{\partial x}\left(\phi \dfrac{q^2}{h} + \phi g \dfrac{h^2}{2}\right) = S_{0,x} + S_{f,x} & \text{(8b)} \end{cases}$$

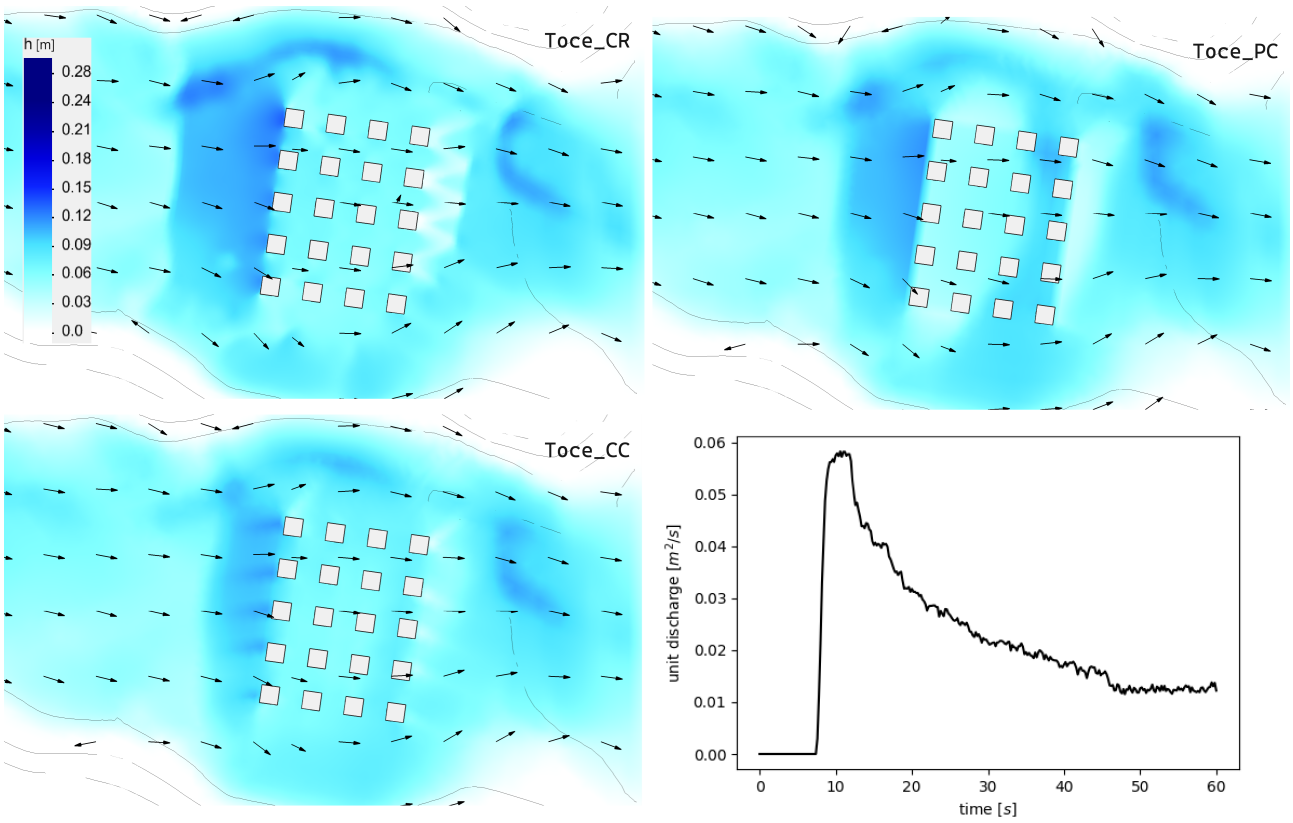

Figure 3: Toce test case: water depth h [m] (color) and unit discharge $h\boldsymbol{u}$ [m^2/s](vectors) at time $t = 20s$ for `Toce_CR` (top left), `Toce_PC` (top right), `Toce_CC` (bottom left). Injected unit discharge at the boundary (bottom right)

Let us define the Froude number as:

$$\text{Fr} = \frac{|u|}{\sqrt{gh}} \tag{9}$$

We say that the shallow water flow is subcritical if Fr < 1. On the other hand if Fr > 1 we say that the flow is supercritical. If Fr ≈ 1 the flow is said to be critical. Froude number provides information on the direction of information propagation: waves celerities in SWE are in fact given by $\lambda_{1-2} = u \pm \sqrt{gh}$. If $Fr < 1$, it means that $|u| < \sqrt{gh}$. As a consequence λ_{1-2} will have opposite sign. Physically, this translates to the ability of information to propagate upstream against the flow direction. On the other hand, if $Fr > 1$, it means that $|u| > \sqrt{gh}$ so both λ_{1-2} will be positive and thus information can only propagates downstream. For this reason the Froude number is crucial to determine where boundary conditions are needed in order for the shallow water problem to be well-posed. For subcritical flows two boundary conditions are required: one at the inflow and one at the outflow. For critical flows two inflow and zero outflow boundary conditions are required. Different boundary condition choices can be made for shallow water equations [19–22], such as:

- Solid wall: if we consider a boundary node belonging to a solid wall, we can prescribe the no-slip boundary condition $u = 0$. Since we will always consider wet conditions, namely $h > 0$, the no-slip boundary condition translates to $q = 0$.
- Water depth given: suppose we know a priori what water level we will have at the boundary. We can prescribe it: $h = h_b$.
- Transparent boundary conditions: they are designed to allow waves to exit the computational domain without reflecting back into the interior. They are adopted in those cases where the computational domain does not cover the entire physical space (for instance in ocean modeling or river simulations). More details on transparent boundary conditions are reported in [23].

In this work different boundary conditions are used. In particular, when dealing with dam-break tests the solid wall boundary condition at inflow and outflow is adopted. When dealing with other cases, the water depth at inflow and outflow is fixed.

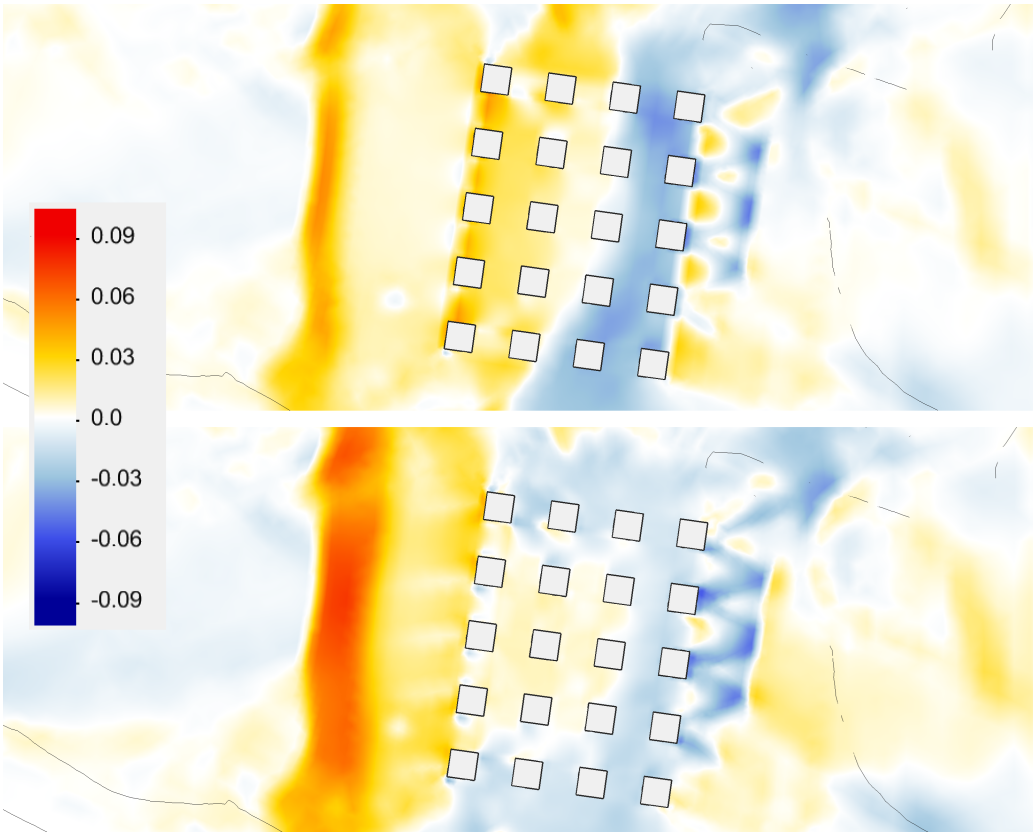

Figure 4: Toce test case: water depth difference [m] between `Toce_CR` and `Toce_PC` (top) and between `Toce_CR` and `Toce_CC` (bottom)

3.1. Numerical discretization

A 1D solver for the one dimensional SWE with porosity has been developed in the scope of this thesis. The solver adopts an explicit time discretization and the Harten-Lax-van Leer (HLL) Riemann solver for the computation of the flux [24, 25]. The numerical scheme is here presented.
The 1D-SWE with porosity can be written in compact form as:

$$\frac{\partial}{\partial t}(\phi \boldsymbol{U}) + \frac{\partial}{\partial x}(\phi \boldsymbol{F}) = \boldsymbol{S} \tag{10}$$

where

$$\boldsymbol{U} = \begin{pmatrix} h \\ q \end{pmatrix}, \ \boldsymbol{F} = \begin{pmatrix} q \\ q^2/h + gh^2/2 \end{pmatrix}, \ \boldsymbol{S} = \begin{pmatrix} 0 \\ S_{0,x} + S_{f,x} \end{pmatrix}.$$

The space domain is discretized into segments (cells) of uniform length Δx. We suppose that the numerical solution is piecewise constant so that a single value is needed for each cell. This will lead to discontinuities of the solution across the boundaries of each cell. For this reason, we need to solve a Riemann problem at each of these boundaries.
The discrete problem in space and time, using finite volumes and explicit Euler, reads:

$$\boldsymbol{U}_i^{n+1} = \boldsymbol{U}_i^n - \frac{\Delta t}{\phi_i \Delta x}((\phi \boldsymbol{F})_{i+\frac{1}{2}}^n - (\phi \boldsymbol{F})_{i-\frac{1}{2}}^n) + \frac{\Delta t}{\phi_i} \boldsymbol{S}_i^n \tag{11}$$

where the superscripts refer to the time discretization and the subscripts to the space discretization. Notice that the half integer location $i + \frac{1}{2}$ refers to right interface of the i-th cell. The fluxes $(\phi \boldsymbol{F})_{i+\frac{1}{2}}^n$ and $(\phi \boldsymbol{F})_{i-\frac{1}{2}}^n$ are computed by implementing the HLL Riemann solver described in the following.
Let us denote the two discontinuous states of the Riemann problem as $\boldsymbol{U}_L$ (left state) and $\boldsymbol{U}_R$ (right state). In the SWE the wave celerities are given by $\lambda_{1,2} = q/h \pm \sqrt{gh}$. We can then estimate the wave speeds as $S_L = \min(q_L/h_L - \sqrt{gh_L}, q_R/h_R - \sqrt{gh_R})$ and $S_R = \max(q_L/h_L + \sqrt{gh_L}, q_R/h_R + \sqrt{gh_R})$ [26, 27].
The physical fluxes on the two sides of the interface are:

$$(\phi \boldsymbol{F})_L = \begin{pmatrix} \phi_L q_L \\ \phi_L q_L^2/h_L + \frac{1}{2}\phi_L g h_L^2 \end{pmatrix}, \; (\phi \boldsymbol{F})_R = \begin{pmatrix} \phi_R q_R \\ \phi_R q_R^2/h_R + \frac{1}{2}\phi_R g h_R^2 \end{pmatrix}$$

The HLL flux is finally computed as:

$$\tilde{\boldsymbol{F}} = \begin{cases} (\phi \boldsymbol{F})_L & \text{if } S_L \geq 0 \\ \frac{S_R(\phi \boldsymbol{F})_L - S_L(\phi \boldsymbol{F})_R + S_L S_R(\boldsymbol{U}_R - \boldsymbol{U}_L)}{S_R - S_L} & \text{if } S_L < 0 < S_R \\ (\phi \boldsymbol{F})_R & \text{if } S_R \leq 0 \end{cases} \tag{12}$$

Putting all together we get the following scheme:

$$\boldsymbol{U}_i^{n+1} = \boldsymbol{U}_i^n - \frac{\Delta t}{\phi_i \Delta x}\left(\tilde{\boldsymbol{F}}(\boldsymbol{U}_i^n, \boldsymbol{U}_{i+1}^n) - \tilde{\boldsymbol{F}}(\boldsymbol{U}_{i-1}^n, \boldsymbol{U}_i^n)\right) + \frac{\Delta t}{\phi_i}\boldsymbol{S}_i^n \tag{13}$$

Since an explicit scheme is used, the solver is subject to a stability constraint that yields a maximum admissible computational time step, namely the CFL condition. In our case the CFL condition reads $\Delta t \leq \frac{\Delta x}{\lambda}$, where λ is the wave celerity. For this reason Δt is updated at each time step as follows:

$$\Delta t^{n+1} = \frac{\Delta x}{\max_i(|q_i^n/h_i^n| + \sqrt{g h_i^n})} \tag{14}$$

3.2. Solver validation

In this section some validation tests are provided. Notice that these test cases are performed under the ideal hypothesis of frictionless condition ($S_f = 0$). We consider the dam break test case [18] with the parameters reported in Table 1: in particular we set a uniform porosity $\phi = 0.5$ in the whole domain and we adopt a uniform mesh whose size is $\Delta x = 0.5m$. We denote with h_{num} and q_{num} the numerical solution given by our solver, and with h_{ex} and q_{ex} the exact analytical solution. Figure 5 reports the computed and exact water depth h and unit discharge q at time $t = 30s$. The results show good agreement between the numerical and analytical solutions.

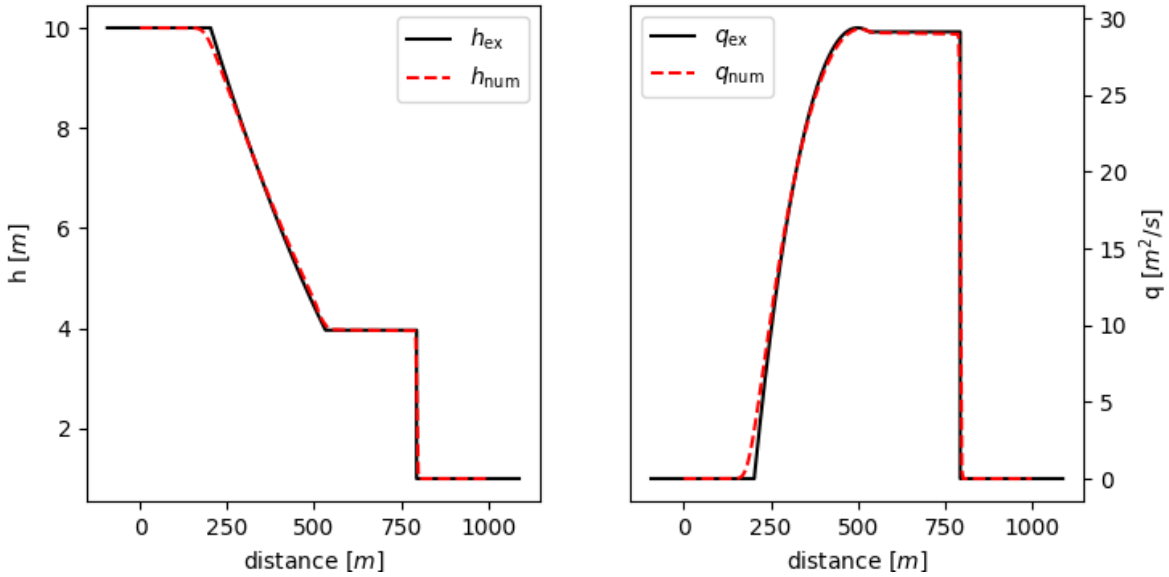

Figure 5: h and q at time $t = 30s$, dam break with uniform porosity $\phi = 0.5$, mesh size $\Delta x = 5m$

We then consider a dam break test case with varying porosity whose parameters are reported in Table 2. Now ϕ linearly varies from 0 to 1 in $[0, L]$. In this case an analytical solution is not available, however in [14] a reference solution is presented. We denote such reference solution with h_{ref} and q_{ref}. The numerical and reference solutions at times $t = 4s$ and $t = 10s$ are reported in Figure 6.
Moreover, a mesh convergence study is performed to ensure that the results do not depend on the number of nodes or, in other words, that the solution converges with respect to the mesh size. Figure 7 reports h and q at time $T = 30s$ in the same dam-break case using 250, 500, and 1000 mesh nodes. The first row refers to the case of uniform porosity $\phi = 0.5$, the second row refers to the case of linearly varying porosity from 0 to 1.

Table 1: Parameters of the one-dimensional dam break test case with uniform porosity

Symbol	Meaning	Value
g	Gravitational acceleration	$9.81 m/s^2$
$h_{0,L}$	Initial water depth on the left-hand side of the dam	10 m
$h_{0,R}$	Initial water depth on the right-hand side of the dam	1 m
L	Length of the domain	1000 m
x_0	Location of the dam	500 m
Δx	Cell size	5 m
ϕ	Uniform porosity	0.5
z_b	Uniform bed elevation	0.0 m

Table 2: Parameters of the one-dimensional dam break test case with varying porosity

Symbol	Meaning	Value
g	Gravitational acceleration	$9.81 m/s^2$
$h_{0,L}$	Initial water depth on the left-hand side of the dam	10 m
$h_{0,R}$	Initial water depth on the right-hand side of the dam	1 m
L	Length of the domain	150 m
x_0	Location of the dam	50 m
Δx	Cell size	1 m
ϕ	Linearly varying porosity	0 - 1
z_b	Uniform bed elevation	0.0 m

Visual inspection suggests that the solution does not depend on the size of the mesh. A quantitative analysis is also performed. In Figure 8 the following quantities are represented with respect to the number of cells used to produce the numerical solution $(h_{\text{num}}, q_{\text{num}})$:

$$e_h = \frac{\int_0^L |h_{\text{num}}(x,T) - h_{\text{ref}}(x,T)|dx}{\int_0^L |h_{\text{ref}}(x,T)|dx} \tag{15}$$

$$e_q = \frac{\int_0^L |q_{\text{num}}(x,T) - q_{\text{ref}}(x,T)|dx}{\int_0^L |q_{\text{ref}}(x,T)|dx} \tag{16}$$

We refer to the first quantity as *relative error h*, while to the second as *relative error q*. h_{ref} and q_{ref} refer to the analytical solution when this is available (namely in the case of uniform porosity), otherwise (namely in the case of linearly varying porosity) they refer to the numerical solution produced using a very fine mesh made of 4000 cells. We consider the solution to be sufficiently accurate when the relative error for both h and q falls below a 2% threshold: to this end, for the case of uniform porosity we need at least 250 cells, while for the case of varying porosity, we need approximately 500 cells.

4. Variational data assimilation

Variational data assimilation is a technique used to optimize model predictions by integrating observational data into numerical models. Variational data assimilation has been developed in the context of numerical weather prediction (NWP) [9, 10] in order to estimate more precise initial conditions to feed the deterministic models. NWP is in fact an initial state problem: accurate initial conditions are needed in order to make a sensible prediction of the future state.

The principle of variational data assimilation is relatively simple: we have a set of observations over a time interval, together with a model that describes the evolution of the quantities of interest. We first define a scalar function which measures the "distance" between the solution given by the model and the observations. We then seek for the best model parameters (initial conditions, boundary conditions, given parameters etc.) which minimize such function. To do so a gradient descent algorithm is implemented. Each step requires the knowledge of some approximation of the partial derivatives. Bearing in mind that in real-world applications the

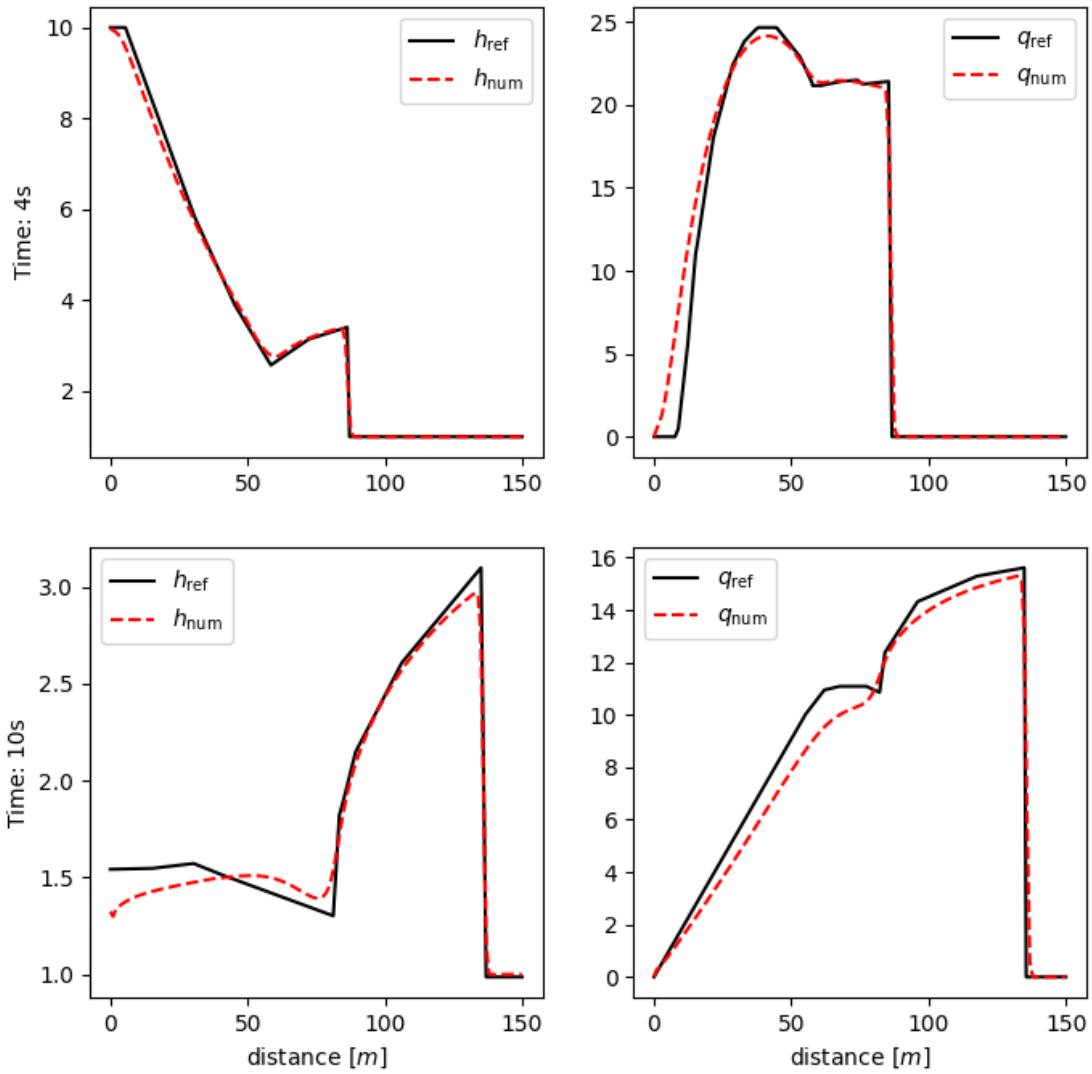

Figure 6: h and q at time $t = 4s$ and $t = 10s$, dam break with varying porosity

number of components of the gradient varies from 10^6 to 10^8, the computationally cheapest way of determining the gradient is through the adjoint model.

4.1. 3D/4D-Var formalism

In this section, we introduce the classical 3D-Var and 4D-Var formalisms [28–30].
Let $\boldsymbol{x} \in \mathbb{R}^n$ denote the vector representing the system state to be estimated, and let $\boldsymbol{y} \in \mathbb{R}^p$ represent the vector of observations. These observations are related to the model state through the observation operator $H : \mathbb{R}^n \to \mathbb{R}^p$, which maps the state into the observation space, such that $\boldsymbol{y} = H(\boldsymbol{x})$. The uncertainties associated with the observations are described by the observation error covariance matrix R. We also define $\boldsymbol{x}_b$ as the background state, which serves as a prior estimate of the true system state. In the context of numerical weather prediction (NWP), this background state is typically derived from a short-range model forecast. The uncertainty in $\boldsymbol{x}_b$ is quantified by the background error covariance matrix B.
The 3D variational data assimilation, or simply 3D-Var, aims to minimize the following cost function:

$$J(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_b)^T B^{-1}(\boldsymbol{x} - \boldsymbol{x}_b) + \frac{1}{2}(H(\boldsymbol{x}) - \boldsymbol{y})^T R^{-1}(H(\boldsymbol{x}) - \boldsymbol{y}) \tag{17}$$

which can be expressed in a more compact form as:

$$J = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_b\|_b^2 + \frac{1}{2}\|H(\boldsymbol{x}) - \boldsymbol{y}\|_o^2 = J_b(\boldsymbol{x}) + J_o(\boldsymbol{x}) \tag{18}$$

where J_b and J_o represent the background and observation components of the cost function, respectively. The term J_b serves as a regularization term for the minimization problem: indeed, suppose that H is linear. Then

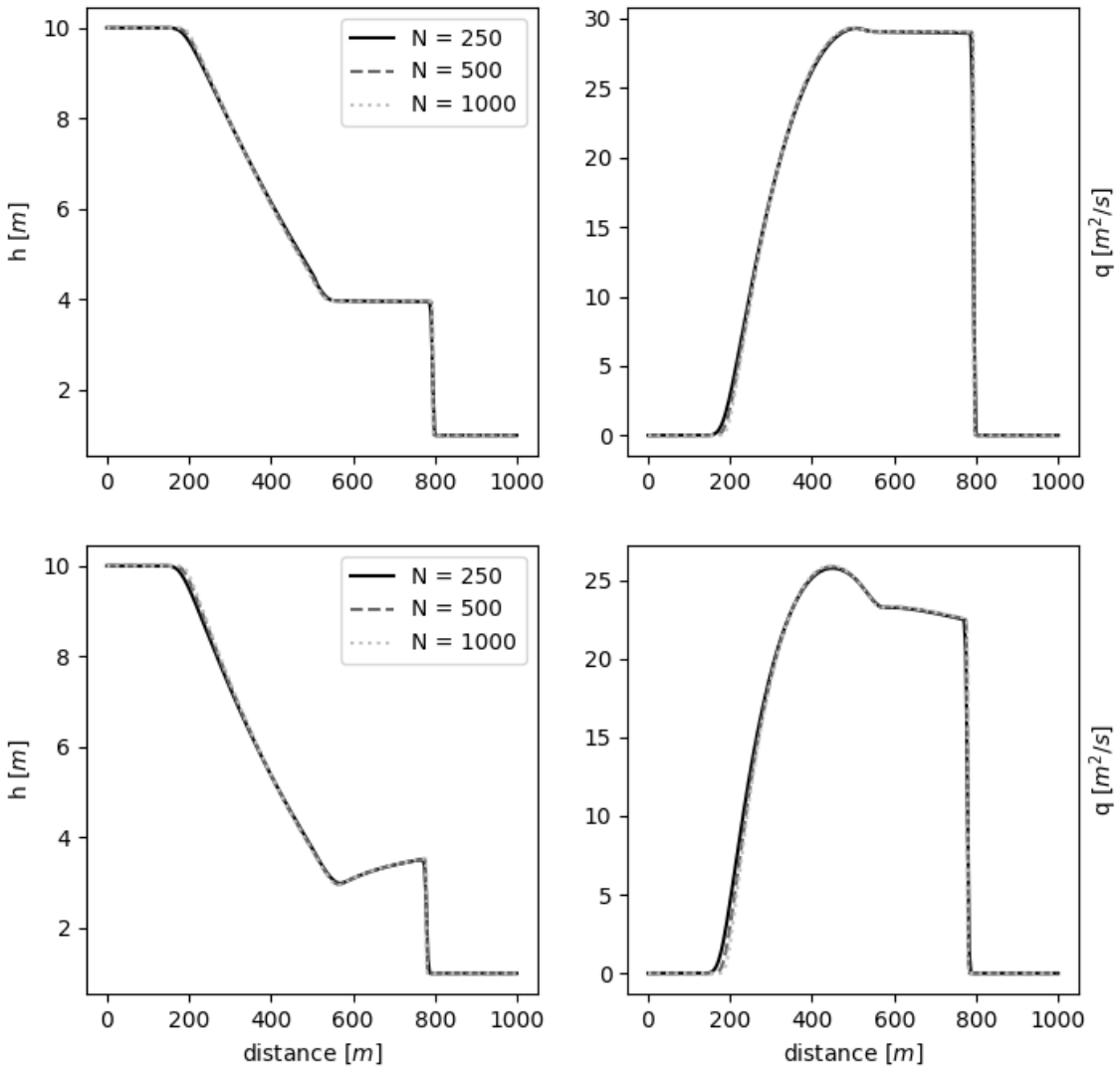

Figure 7: h and q at time $T = 30s$ using different meshes. Uniform porosity (top), varying porosity (bottom)

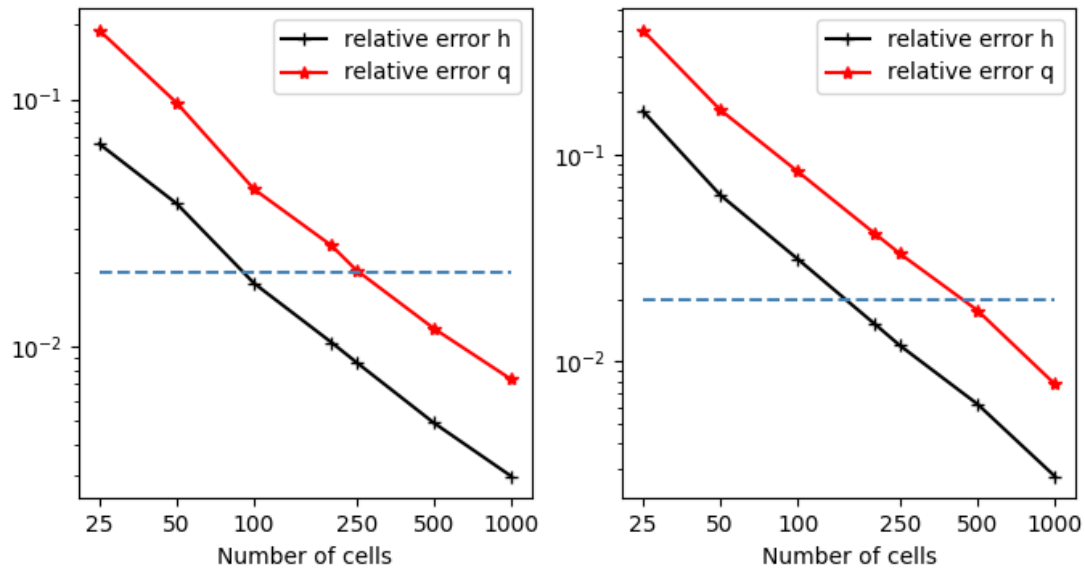

Figure 8: Mesh independence study in the case of uniform porosity (left) and varying porosity (right). The dotted blue line represents the 2% threshold

J_o is quadratic, however, it generally does not have a unique minimum. Adding J_o makes the problem of minimizing $J = J_o + J_b$ well posed. In the general case where H is not linear, we cannot guarantee the existence of a unique minimum of J, nonetheless, the term J_b makes the cost function "more quadratic".
The same approach can be naturally extended to the case where observations are available over a time window: let $\boldsymbol{y}(t_i)$ with $i = 1, ..., N$ denote the time-distributed observations. In this case, the observation cost function J_o becomes:

$$J_o = \frac{1}{2} \sum_{i=0}^{N} \|H_i(\boldsymbol{x}(t_i)) - \boldsymbol{y}(t_i)\|_o^2 \tag{19}$$

Suppose we have a model M, called the forward model, describing the evolution in time of $\boldsymbol{x}$. We write $\frac{d\boldsymbol{x}}{dt} = M(\boldsymbol{x})$ with $\boldsymbol{x}(t = 0) = \boldsymbol{x}_0$. Then J can be minimized with respect to $\boldsymbol{x}_0$ instead of $\boldsymbol{x}$. In fact, we can write $\boldsymbol{x}$ at time i as $\boldsymbol{x}(t_i) = M_{0 \to t_i}(\boldsymbol{x}_0)$. The cost function then becomes:

$$J(\boldsymbol{x}_0) = \frac{1}{2} \|\boldsymbol{x}_0 - \boldsymbol{x}_b\|_b^2 + \frac{1}{2} \sum_{i=0}^{N} \|H_i(M_{0 \to t_i}(\boldsymbol{x}_0)) - \boldsymbol{y}(t_i)\|_o^2 \tag{20}$$

We call this approach 4D variational data assimilation or simply 4D-Var. Notice that we assume that the model is perfect, meaning that $\boldsymbol{x}$ is fully determined by the initial condition $\boldsymbol{x}_0$ or, in other words, that the evolution of the system state is governed exactly by the model equations without any model error. The minimization of Eq. (20) requires its gradient with respect to $\boldsymbol{x}_0$ to vanish. If the operators H and M are linear, we can write:

$$\nabla_{\boldsymbol{x}_0} J = B^{-1}(\boldsymbol{x}_0 - \boldsymbol{x}_b) + \sum_{i=0}^{N} M_{0 \to t_i}^T H_i^T R^{-1} [H_i(M_{0 \to t_i}(\boldsymbol{x}_0)) - \boldsymbol{y}(t_i)] = 0 \tag{21}$$

where H^T is the adjoint observation operator, while M^T is called the adjoint model and consists in a backward integration from time t_i to time 0.
Notice that, in general, we do not know a closed-form for the operators H and M. Indeed, in practical applications, there are two primary strategies for implementing variational data assimilation: the first involves coding the tangent linear model (TLM) directly within the discrete forward model [31, 32], while the second, adopted in this work, consists in deriving and implementing the adjoint model through the adjoint equations by making use of the Lagrange multipliers.

4.2. Data assimilation for an ODE

In this section we present a simple example [33] to show how to derive the adjoint equations and exploit the adjoint variables to compute the gradient of the cost function with the aim of assimilating a model parameter. Let $c(x)$ be a given parameter. Consider the model described by the following ordinary differential equation (ODE):

$$\begin{cases} u''(x) + c(x)u'(x) = f(x), \ x \in (0, 1) \\ u(0) = u(1) = 0 \end{cases} \tag{22}$$

with $f(x) = 0.5, \ x \in (0, 1)$. This will be our forward model. Let us define the cost function as:

$$J(u, c) = \frac{1}{2} \int_0^1 (u(x) - u_{\text{obs}}(x))^2 dx \tag{23}$$

Following the 3D-Var notation presented in section 4.1, we are taking into account only the observation part J_o of the cost function J given by Eq. (18). Moreover, we are imposing the observation error covariance matrix R to be equal to the identity matrix, meaning that the observation error is assumed to be null. Clearly J depends on the parameter c, however, even in this simple case, *how* c affects J is not obvious. Our aim is to find $c(x)$ that minimizes J, i.e. we want to find $c(x)$ such that the distance between the observation u_{obs} and the prediction from the forward model u is minimal. This leads to a constrained optimization problem, which we formulate via a Lagrangian:

$$\mathcal{L}(u, c, \lambda) = J(u, c) + \int_0^1 \lambda(x)(u''(x) + c(x)u'(x) - f(x))dx \tag{24}$$

After integration by parts we get:

$$\mathcal{L}(u, c, \lambda) = J(u, c) + \int_0^1 [\lambda''(x)u(x) - (\lambda(x)c(x))'u(x) - \lambda(x)f(x)]dx \tag{25}$$

Here we have exploited the fact that $u(0) = u(1) = 0$. Moreover we impose $\lambda(0) = \lambda(1) = 0$. Taking the functional derivative of $\mathcal{L}$ with respect to u (see Appendix A for more details), and setting it to zero yields:

$$\frac{\delta \mathcal{L}}{\delta u} = u(x) - u_{obs}(x) + \lambda''(x) - (\lambda(x)c(x))' = 0 \tag{26}$$

In this way we obtain the adjoint (or backward) model:

$$\begin{cases} -\lambda''(x) + (c(x)\lambda(x))' = u(x) - u_{obs}(x),\ x \in (0,1) \\ \lambda(0) = \lambda(1) = 0 \end{cases} \tag{27}$$

The gradient of J with respect to the parameter c is then computed from (24) by setting $\nabla_c \mathcal{L} = 0$:

$$\nabla_c J = -u'(x)\lambda(x) \tag{28}$$

The forward model and its adjoint counterpart can be discretized using the finite difference method. A gradient descent algorithm is run in order to retrieve the optimal $c(x)$. Because we know that the gradient of a function is zero at its minimum, a natural method for stopping the iterations is to stop when the norm of the gradient of the cost function falls below a specified tolerance ϵ. In any case, we also set the maximum number of possible iterations N_{max}.

Algorithm 1 Data assimilation for c(x)

1: $k = 0$, $\|\nabla_c J^0\|_2 = 1 + \epsilon$
2: **while** $k < N_{\max}$ and $\|\nabla_c J^k\|_2 > \epsilon$ **do**
3: Compute u^k from the discrete forward model
4: Compute λ^k from the discrete adjoint model
5: Compute $\nabla_c J^k$ and $\|\nabla_c J^k\|_2$
6: $c^{k+1} = c^k - \delta \nabla_c J^k$
7: $k \to k + 1$
8: **end while**

Notice that the variational data assimilation procedure performed in operational settings (such as in NWP) is usually not run to complete convergence [34]: in fact the computational cost of the assimilation algorithm in operational settings is prohibitive and just few iterations can be performed. For this reason, in NWP applications, it is often preferred to choose the number of iterations to be performed according to the computational resources available and not to introduce a stopping criteria based on the norm of the gradient. Such fixed number of iterations usually ranges from 10 to 70 [35, 36].
For the numerical experiment the following data were used: the true parameter to be assimilated is given by $c_{\text{target}}(x) = 5.0\sin(4.0x)$, the observation u_{obs} is obtained by running the discrete model with $c_{\text{target}}(x)$ as given parameter. The first guess for $c(x)$ is given by $c_{\text{start}}(x) = 3.0$, and, as before, u_{start} is obtained by running the discrete model with c_{start} as given parameter. We set the tolerance to $\epsilon = 10^{-6}$, the algorithm stops after 30 iterations. In Figure 9 we can see that the assimilated parameter is converging to $c_{\text{target}}(x)$. As a consequence the assimilated u matches almost exactly u_{obs}.

4.3. The adjoint problem for the porous shallow water equations

The adjoint model for the 1D shallow water equations with porosity is here derived. A similar derivation was performed for the standard 1D shallow water equations by Brett F. Sanders and Nikolaos D. Katopodes [11]. We recall that the one dimensional shallow water equations with porosity without friction read:

$$\begin{cases} \dfrac{\partial(\phi h)}{\partial t} + \dfrac{\partial(\phi q)}{\partial x} = 0 & \text{(29a)} \\ \dfrac{\partial(\phi q)}{\partial t} + \dfrac{\partial}{\partial x}\left(\phi \dfrac{q^2}{h} + \phi g \dfrac{h^2}{2}\right) = S_{0,x} & \text{(29b)} \end{cases}$$

where q is given by $q = hu$. Let the cost function be:

$$J = \int_0^T \int_0^L r(h, q; x, t) dx dt \tag{30}$$

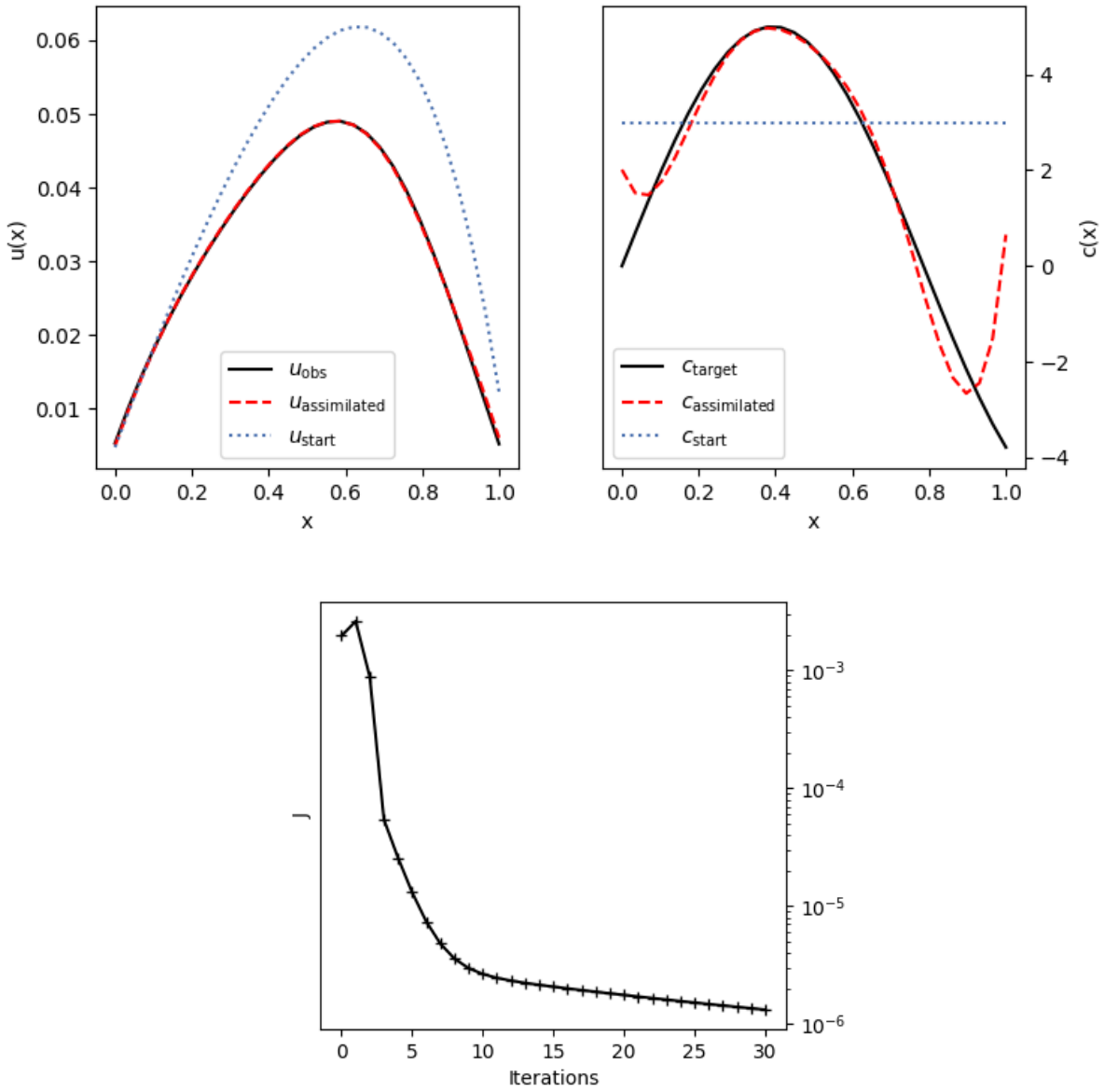

Figure 9: Data assimilation for an ODE, numerical experiment: evolution of solution u (top left) and parameter c (top right) during data assimilation, cost function J with respect to iterations of Algorithm 1 (bottom)

where r is a residual measuring the distance between observations and the computed solution. The Lagrangian then is:

$$\mathcal{L} = J + \int_0^T \int_0^L \lambda_h \left(\frac{\partial(\phi h)}{\partial t} + \frac{\partial(\phi q)}{\partial x} \right) + \lambda_q \left(\frac{\partial(\phi q)}{\partial t} + \frac{\partial}{\partial x}\left(\phi \frac{q^2}{h} + \phi g \frac{h^2}{2} \right) - S_{0,x} \right) dx dt \tag{31}$$

where λ_h and λ_q are two Lagrange multipliers differentiable in x and t. We substitute $S_{0,x}$ with its expression given by (4). After integration by parts the Lagrangian becomes:

$$\begin{aligned} \mathcal{L} = J + \int_0^T \int_0^L -\phi h \frac{\partial \lambda_h}{\partial t} - \phi q \frac{\partial \lambda_h}{\partial x} - \phi q \frac{\partial \lambda_q}{\partial t} - \left(\phi \frac{q^2}{h} + \phi g \frac{h^2}{2} \right) \frac{\partial \lambda_q}{\partial x} - \left(g \frac{h^2}{2} \frac{\partial \phi}{\partial x} - \phi g h \frac{\partial z_b}{\partial x} \right) \lambda_q \, dx dt \\ + \int_0^L \left[\phi h \lambda_h + \phi q \lambda_q \right]_0^T dx + \int_0^T \left[\phi q \lambda_h + \left(\phi \frac{q^2}{h} + \phi g \frac{h^2}{2} \right) \lambda_q \right]_0^L dt \end{aligned} \tag{32}$$

Taking the first variation with respect to h and q we have:

$$\delta\mathcal{L} = \int_0^T \int_0^L \Big[\Big(-\phi\frac{\partial\lambda_h}{\partial t} + \Big(\phi\frac{q^2}{h^2} - \phi g h\Big)\frac{\partial\lambda_q}{\partial x} - \Big(gh\frac{\partial\phi}{\partial x} - \phi g\frac{\partial z_b}{\partial x}\Big)\lambda_q + \frac{\partial r}{\partial h}\Big)\delta h \\ + \Big(-\phi\frac{\partial\lambda_q}{\partial t} - \phi\frac{\partial\lambda_h}{\partial x} - 2\phi\frac{q}{h}\frac{\partial\lambda_q}{\partial x} + \frac{\partial r}{\partial q}\Big)\delta q\Big] dxdt \\ + \int_0^L \Big[\phi\lambda_h\delta h + \phi\lambda_q\delta q\Big]_0^T dx + \int_0^T \Big[\phi\lambda_h\delta q + 2\phi\frac{q}{h}\lambda_q\delta q - \phi\frac{q^2}{h^2}\lambda_q\delta h + \phi g h\lambda_q\delta h\Big]_0^L dt \tag{33}$$

The adjoint model then reads:

$$\begin{cases} \phi\dfrac{\partial\lambda_h}{\partial\tau} + \Big(\phi\dfrac{q^2}{h^2} - \phi g h\Big)\dfrac{\partial\lambda_q}{\partial x} - \Big(gh\dfrac{\partial\phi}{\partial x} - \phi g\dfrac{\partial z_b}{\partial x}\Big)\lambda_q + \dfrac{\partial r}{\partial h} = 0 & \text{(34a)} \\ \phi\dfrac{\partial\lambda_q}{\partial\tau} - \dfrac{\phi\partial\lambda_h}{\partial x} - 2\phi\dfrac{q}{h}\dfrac{\partial\lambda_q}{\partial x} + \dfrac{\partial r}{\partial q} = 0 & \text{(34b)} \end{cases}$$

where $\tau = T - t$ is measured in the reverse time direction (hence the name *backward problem*). The above adjoint model can be written in compact form as follows:

$$\phi\frac{\partial\boldsymbol{\Lambda}}{\partial\tau} + \phi A\frac{\partial\boldsymbol{\Lambda}}{\partial x} + C\boldsymbol{\Lambda} + \boldsymbol{D} = 0 \tag{35}$$

where

$$\boldsymbol{\Lambda} = \begin{pmatrix}\lambda_h \\ \lambda_q\end{pmatrix}$$

$$A = \begin{pmatrix}0 & \frac{q^2}{h^2} - gh \\ -1 & -\frac{2q}{h}\end{pmatrix}$$

$$C = \begin{pmatrix}0 & -gh\frac{\partial\phi}{\partial x} + \phi g\frac{\partial z_b}{\partial x} \\ 0 & 0\end{pmatrix}$$

$$\boldsymbol{D} = \begin{pmatrix}\frac{\partial r}{\partial h} \\ \frac{\partial r}{\partial q}\end{pmatrix}$$

It should be noticed that, in contrast to the basic shallow-water equations, which have well-known conservation properties, the adjoint shallow-water equations cannot be recast in conservative form. For this reason the adjoint problem shall be solved by different numerical schemes than those of the classical shallow-water equations (see Section 4.4), which typically are solved in conservative form.
We take r as:

$$r(h,q;x,t) = \frac{1}{2}\Big(h(x,t) - h_{\text{obs}}(x,t)\Big)^2 + \frac{1}{2}\Big(q(x,t) - q_{\text{obs}}(x,t)\Big)^2 \tag{36}$$

With this choice for r, J becomes:

$$J = \frac{1}{2}\int_0^T \int_0^L \Big[\Big(h(x,t) - h_{\text{obs}}(x,t)\Big)^2 + \Big(q(x,t) - q_{\text{obs}}(x,t)\Big)^2\Big] dxdt \tag{37}$$

where h_{obs} and q_{obs} are the given observational data for h and q. The vector $\boldsymbol{D}$ then reads:

$$\boldsymbol{D} = \begin{pmatrix}h - h_{\text{obs}} \\ q - q_{\text{obs}}\end{pmatrix} \tag{38}$$

What remains to be done is to compute the gradient of J with respect to the parameter that needs to be assimilated. For instance, suppose we want to improve the initial conditions for h and q [37], so that the simulation better matches the observations. Taking in Eq. (32) the variation of $\mathcal{L}$ with respect to $h_0 = h(x,0)$ and $q_0 = q(x,0)$, and setting it to 0, we get:

$$\nabla_{h_0} J(x) = \phi(x)\lambda_h(x,0) \tag{39}$$

$$\nabla_{q_0} J(x) = \phi(x)\lambda_q(x,0) \tag{40}$$

The data assimilation algorithm to reconstruct the initial conditions in the SWE with porosity is summarized here:

Algorithm 2 Data assimilation for IC in SWE

1: $k = 0$, $\|\nabla_{h_0} J^0\|_2 = \|\nabla_{q_0} J^0\|_2 = 1 + \epsilon$
2: **while** $k < N_{\max}$ and $\|\nabla_{h_0} J^k\|_2 + \|\nabla_{q_0} J^k\|_2 > 2\epsilon$ **do**
3: Compute $h^k(x,t)$ and $q^k(x,t)$ from the discrete forward model using finite volume method
4: Integrate the discrete adjoint model backwards from $t = T$ to $t = 0$ to obtain $\lambda_h^k(x,t)$ and $\lambda_q^k(x,t)$
5: Compute $\nabla_{h_0} J^k$ and $\nabla_{q_0} J^k$
6: $h_0^{k+1}(x) = h_0^k(x) - \delta \nabla_{h_0} J(x)$, $q_0^{k+1}(x) = q_0^k(x) - \delta \nabla_{q_0} J(x)$
7: $k \to k + 1$
8: **end while**

4.4. Numerical solver for the adjoint shallow water equations with porosity

The adjoint problem is solved by implementing the Roe's one step fluctuation splitting method [38] with the addition of the porosity ϕ. This method was previously presented by Glaister [39] to solve conservation equations and later extended by LeVeque [40] to solve hyperbolic equations that do not appear in conservative form, by generalizing the classical flux difference splitting. This extension is best illustrated by a simple example. Consider the following general hyperbolic system:

$$\frac{\partial w}{\partial t} + A(x)\frac{\partial w}{\partial x} = 0 \tag{41}$$

We wish to approximate the term $A(x)\frac{\partial w}{\partial x}$ by computing the net effect of all the waves, called fluctuations, arising at each cell edge from the Riemann problem. The fluctuations $\mathcal{A}\Delta w$ are split into a right and a left-going fluctuations, namely $\mathcal{A}^+\Delta w$ and $\mathcal{A}^-\Delta w$. Notice that, in this non-conservative problem, this does not correspond to the splitting of any flux difference. In particular:

$$\mathcal{A}^+\Delta w + \mathcal{A}^-\Delta w \neq A_i w_i - A_{i-1} w_{i-1} \tag{42}$$

Nonetheless, numerical results presented by LeVeque in [40] confirm the effectiveness of this method in solving non-conservative system of equations.
Here the 2D discretization scheme is presented for the sake of generality. The 1D version of such scheme is then implemented by dropping the irrelevant terms. The two-dimensional adjoint equations, i.e. the 2D counterpart of (35), are:

$$\phi\frac{\partial \mathbf{\Lambda}}{\partial \tau} + \phi A\frac{\partial \mathbf{\Lambda}}{\partial x} + \phi B\frac{\partial \mathbf{\Lambda}}{\partial y} + C\mathbf{\Lambda} + \boldsymbol{D} = 0 \tag{43}$$

The matrices A and B read:

$$A = \begin{pmatrix} 0 & u^2 - gh & uv \\ -1 & -2u & -v \\ 0 & 0 & -u \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & uv & v^2 - gh \\ 0 & -v & 0 \\ -1 & -u & -2v \end{pmatrix}$$

The matrix A can be decomposed as $A = R\Omega R^{-1}$, where

$$\Omega = \begin{pmatrix} -\sqrt{gh} - u & 0 & 0 \\ 0 & -u & 0 \\ 0 & 0 & \sqrt{gh} - u \end{pmatrix}$$

$$R = \frac{1}{2\sqrt{gh}} \begin{pmatrix} \sqrt{gh} - u & -v & \sqrt{gh} + u \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

Ω is the diagonal matrix containing the eigenvalues of A. The columns of R contain the corresponding right eigenvectors. We define $\Omega^{\pm}$ as the the diagonal matrices containing the positive and negative eigenvalues of A respectively. We split the fluctuations $\mathcal{A}\Delta\mathbf{\Lambda}$ into positive and negative moving waves as follows:

$$\mathcal{A}\Delta\mathbf{\Lambda} = \mathcal{A}^+\Delta\mathbf{\Lambda} + \mathcal{A}^-\Delta\mathbf{\Lambda} \tag{44}$$

where $\mathcal{A}^{\pm} = R\ \Omega^{\pm} R^{-1}$. The fluctuations relative to the term $B\frac{\partial \mathbf{\Lambda}}{\partial y}$ are split into $\mathcal{B}^{+}\Delta\mathbf{\Lambda}$ and $\mathcal{B}^{-}\Delta\mathbf{\Lambda}$ in an analogous way. The fluctuation splitting approach leads to the following discretization of Eq. (43):

$$\mathbf{\Lambda}_{j,k}^{n-1} = \mathbf{\Lambda}_{j,k}^{n} - \frac{\Delta t^n}{\phi_{j,k}}\left[\frac{(\phi\mathcal{A}^{+}\Delta\mathbf{\Lambda})^n_{j-\frac{1}{2},k} + (\phi\mathcal{A}^{-}\Delta\mathbf{\Lambda})^n_{j+\frac{1}{2},k}}{\Delta x} + \frac{(\phi\mathcal{B}^{+}\Delta\mathbf{\Lambda})^n_{j,k-\frac{1}{2}} + (\phi\mathcal{B}^{-}\Delta\mathbf{\Lambda})^n_{j,k+\frac{1}{2}}}{\Delta y} + (C\mathbf{\Lambda})^n_{j,k} + D^n_{j,k}\right] \tag{45}$$

where $\Delta\mathbf{\Lambda}^n_{j+\frac{1}{2},k} = \mathbf{\Lambda}^n_{j+1,k} - \mathbf{\Lambda}^n_{j,k}$. Notice that the adjoint equations are integrated backwards. Boundary conditions are implemented by linearly extrapolating the adjoint variables to the ghost cells while zero-order extrapolating the gradients [19, 38]. A null initial condition is set for $\mathbf{\Lambda}$, namely $\mathbf{\Lambda}(\boldsymbol{x}, T) = \mathbf{0}$. It should be noticed that initial conditions for the adjoint problem refers to $t = T$ or, in other words, $\tau = 0$.

4.5. Improved gradient descent method: backtracking line search

In the example in Section 4.2 we have retrieved the optimal parameter through a standard gradient descent algorithm with a fixed step length δ. More sophisticated methods can be implemented to obtain a faster convergence. Ideally, at each iteration k, given the gradient descent direction $-\nabla J^k(\boldsymbol{x}^k)$, we aim to determine the optimal step length δ^k. Here the backtracking line search algorithm is described.
In general, suppose we want to minimize the function $f : \mathbb{R}^n \to \mathbb{R}$. Gradient based optimization algorithms read:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \delta^k \boldsymbol{d}^k \tag{46}$$

where $\boldsymbol{d}^k$ is the descent direction at the k-th iteration. We choose $\boldsymbol{d}^k = -\nabla J(\boldsymbol{x}^k)$. Once we pick the search direction, line search algorithm seeks to minimize the function

$$h(\delta^k) = f(\boldsymbol{x}^k - \delta^k \nabla J(\boldsymbol{x}^k)) \tag{47}$$

with respect to δ^k. However, finding the true minima of this function would be unfeasible. The idea of backtracking line search is then to find an acceptable step length δ^k, rather than minimizing the function h over $\delta^k \in \mathbb{R}^+$ exactly. To do so, at each iteration backtracking line search starts with a large step length guess and checks whether such step satisfies the Armijo condition [41], namely:

$$f(\boldsymbol{x}^k + \delta^k \boldsymbol{d}^k) \leq f(\boldsymbol{x}^k) + c\delta^k (\boldsymbol{d}^k)^T \nabla f(\boldsymbol{x}^k) = f(\boldsymbol{x}^k) - c\delta^k \nabla f(\boldsymbol{x}^k)^T \nabla f(\boldsymbol{x}^k) \tag{48}$$

where $0 < c < 1$. Usually c is taken quite small (e.g. $c = 10^{-4}$ [42]). This inequality makes sure that the step length δ^k is such that f decreases sufficiently. If Eq. (48) is indeed satisfied, δ^k is picked as the right step length. Otherwise, if the Armijo condition is not satisfied, δ^k is shrunk by multiplying it by a factor $\alpha \in (0, 1)$ and the check by means of the Armijo condition is repeated. The price to pay for choosing a better step length is that at each gradient descent iteration we have to evaluate $f(\boldsymbol{x}^k + \delta^k \boldsymbol{d}^k)$ possibly multiple times. In our context, this translates to the need of running the forward model more than once at each gradient descent iteration. Despite that, we have found that the backtracking line search is slightly better performing than fine-tuned standard gradient descent, even when taking into account the multiple evaluations of f nested inside each gradient descent iteration (see Section 5.2).
Let γ be the parameter that we aim to assimilate. Here we report the modified assimilation algorithm with the adoption of backtracking line search:

Algorithm 3 Data assimilation with backtracking line search

$k = 0$, $\|\nabla_\gamma J^0\|_2 = 1 + \epsilon$
while $k < N_{\max}$ and $\|\nabla_\gamma J^k\|_2 > \epsilon$ **do**
  Compute $h^k(x,t)$ and $q^k(x,t)$ from the discrete forward model
  Integrate the discrete adjoint model backwards from $t = T$ to $t = 0$ to obtain $\lambda_h^k(x,t)$ and $\lambda_q^k(x,t)$
  Compute $\nabla_\gamma J^k$ and $\|\nabla_\gamma J^k\|_2$
  while $J(\gamma^k - \delta^k \nabla_\gamma J^k) > J(\gamma^k) - c\delta^k \nabla_\gamma J^{k^T} \nabla_\gamma J^k$ **do**
    $\delta^k \to \alpha \delta^k$
  end while
  $\gamma^{k+1} = \gamma^k - \delta^k \nabla_\gamma J^k$
  $k \to k + 1$
end while

5. Numerical Experiments

In this section we report the results of some numerical experiments performed using the numerical solvers previously presented.

5.1. Assimilation of initial conditions

In this subsection we perform data assimilation with the aim of understanding the initial conditions for h and q. The parameters reported in Table 3 are used for both the forward and backward problems. Synthetic data used as observations are generated by running the forward solver using the following initial conditions for h and q:

$$\begin{cases} h_{\text{obs}}(x, t = 0) = 0.05 \cdot e^{-0.1(x-50.0)^2} + 1.0 \\ q_{\text{obs}}(x, t = 0) = 0.0 \end{cases}$$

We will call them *observed initial conditions*.

Table 3: Data assimilation for initial conditions for h and q

Symbol	Meaning	Value
g	Gravitational acceleration	$9.81 m/s^2$
L	Length of the domain	100 m
Δx	Cell size	1.25 m
ϕ	Uniform porosity	1.0
z_b	Uniform bed elevation	0.0 m

The first guess is produced by running the forward solver with the following initial conditions for h and q:

$$\begin{cases} h_{\text{start}}(x, t = 0) = 0.02 \cdot e^{-0.1(x-50.0)^2} + 1.0 \\ q_{\text{start}}(x, t = 0) = 10^{-4} \cdot x(x - L) \end{cases}$$

We will call these *first guess initial conditions*. The choice of the first guess is arbitrary. Ideally the initial guess should be "close enough" to the real parameter to be assimilated, however the problem of choosing such first guess has not been examined in this work.
Algorithm 3 is run with an assimilation window of 8 seconds. The tolerance is set to $\epsilon = 10^{-6}$. Figure 10 shows that at the end of the assimilation process, the *assimilated initial conditions* match the observed initial conditions. Figure 11 shows the resulting h and q at time $t = 8s$ obtained by running the forward solver with the observed, assimilated, and first guess initial conditions. Figure 12 shows the behavior of the cost function J with respect to the iterations of the assimilation algorithm.
Different assimilation time windows are tested. Figure 13 shows how well the initial conditions are assimilated with different assimilation window lengths. In particular, the following quantity is considered at each iteration

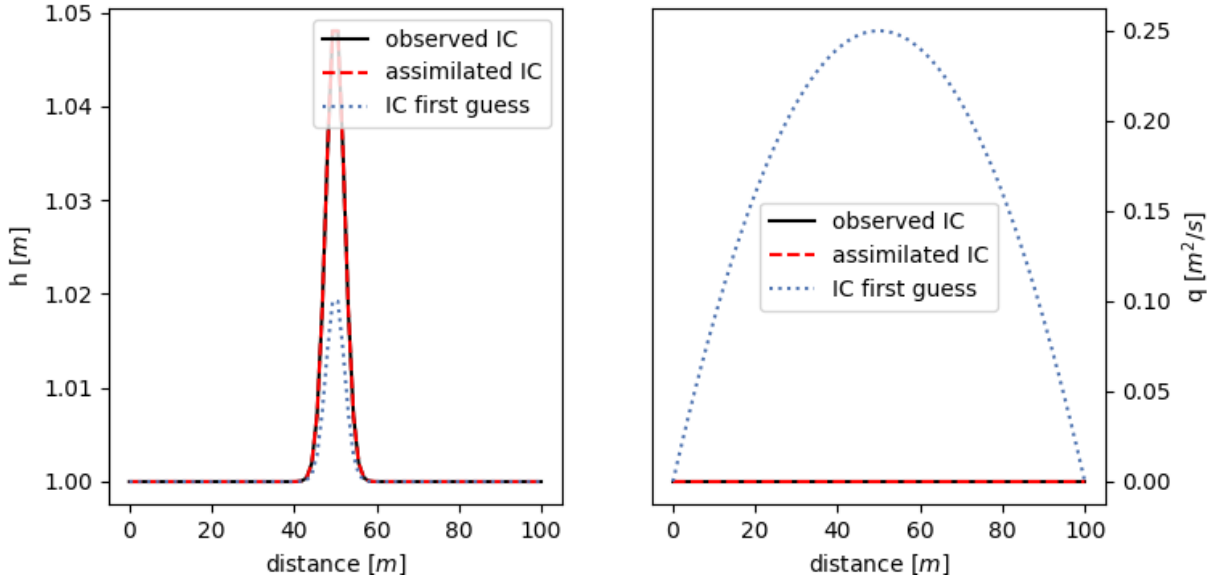

Figure 10: Assimilation of initial conditions: initial conditions for h (left) and q (right)

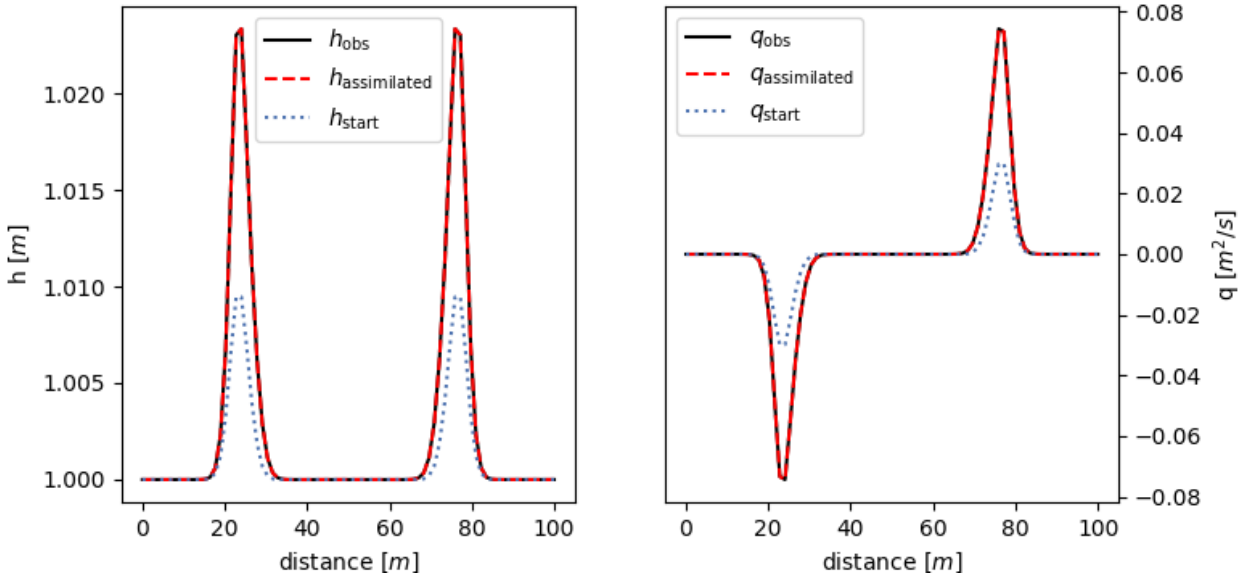

Figure 11: Assimilation of initial conditions: h (left) and q (right) at time $t = 8s$ using different initial conditions in the case of uniform porosity

k:

$$e_A^k = \int_0^L |h^k(x,0) - h_{\text{obs}}(x,0)| + |q^k(x,0) - q_{\text{obs}}(x,0)|dx \tag{49}$$

We call this quantity *assimilation error*. As expected longer assimilation windows result in smaller values of the assimilation error given the same number of iterations of the assimilation algorithm. Of course, the longer the assimilation window, the more computationally expensive the forwards and the backward solvers will be.
The same data assimilation case is then run keeping all the same data except for the porosity ϕ that now varies linearly from 0 to 1 in the interval $[0, L]$. Figure 14 reports h and q at time $t = 8s$, while Figure 15 reports the cost function with respect to the k-th iteration. These two experiments show that the data assimilation algorithm is able to retrieve the observed initial conditions both in the case of uniform and varying porosity.

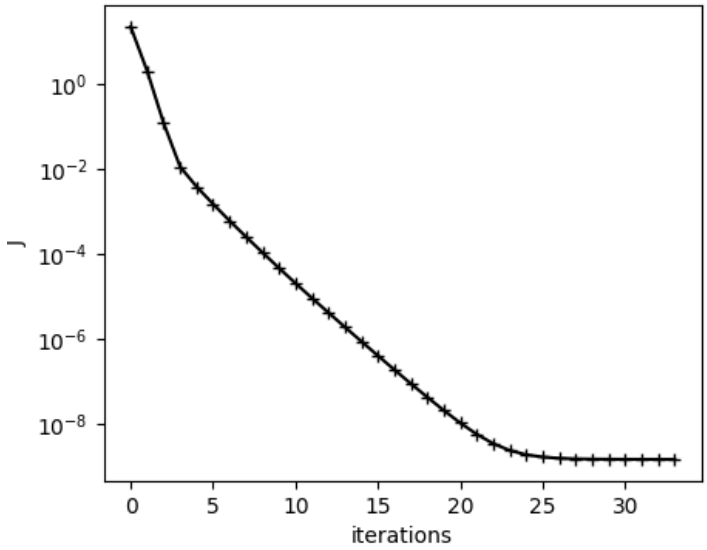

Figure 12: Assimilation of initial conditions: cost function J in the case of uniform porosity

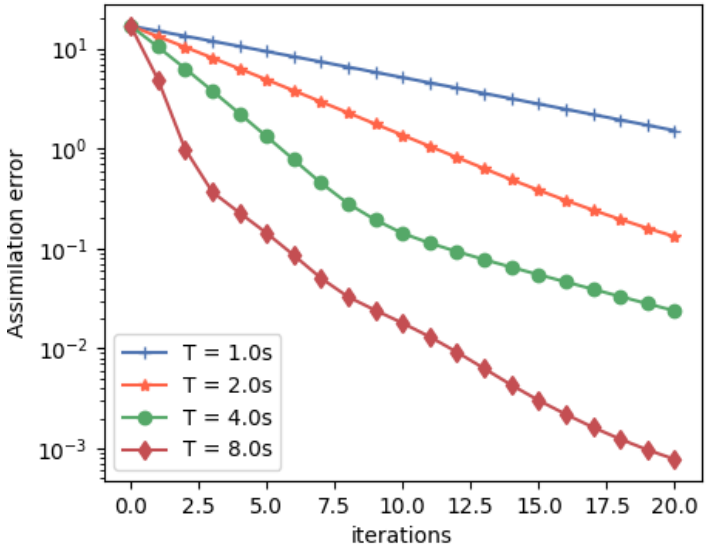

Figure 13: Assimilation of initial conditions: assimilation error with different time windows in the case of uniform porosity (fixed $\delta = 0.1$)

5.2. Assimilation of porosity

The problem of the assimilation of porosity is more interesting in practice than the assimilation of initial conditions: in fact, knowing porosity ϕ a priori would require to estimate the effects of obstacles in the floodplain for each computational cell of our domain. Instead, we aim to use observations of the water depth h and of the unit discharge q to retrieve the value of the porosity ϕ in space a posteriori. The retrieved porosity map can then be used for future simulations. All the methods illustrated before are still valid for the assimilation of porosity, except for the computation of the gradient of the cost function. In theory, to compute the gradient of the cost function J with respect to ϕ starting from Eq. (32), as done in Eq. (39) and in Eq. (40), we should compute an integral in time depending on h, q, λ_h, and λ_q. This would require our adjoint solver to store the solution $\mathbf{\Lambda}$ at all the time steps. In real world applications, such as NWP, this is not feasible due to the number of the degrees of freedom and time steps needed. What we can do instead is to mimic what we have done for the assimilation of the initial conditions: suppose in fact that the porosity ϕ also depends on time, so that we have $\phi = \phi(x, t)$. Now our goal becomes to assimilate the initial conditions for the parameter ϕ, namely $\phi_0 = \phi(x, 0)$.

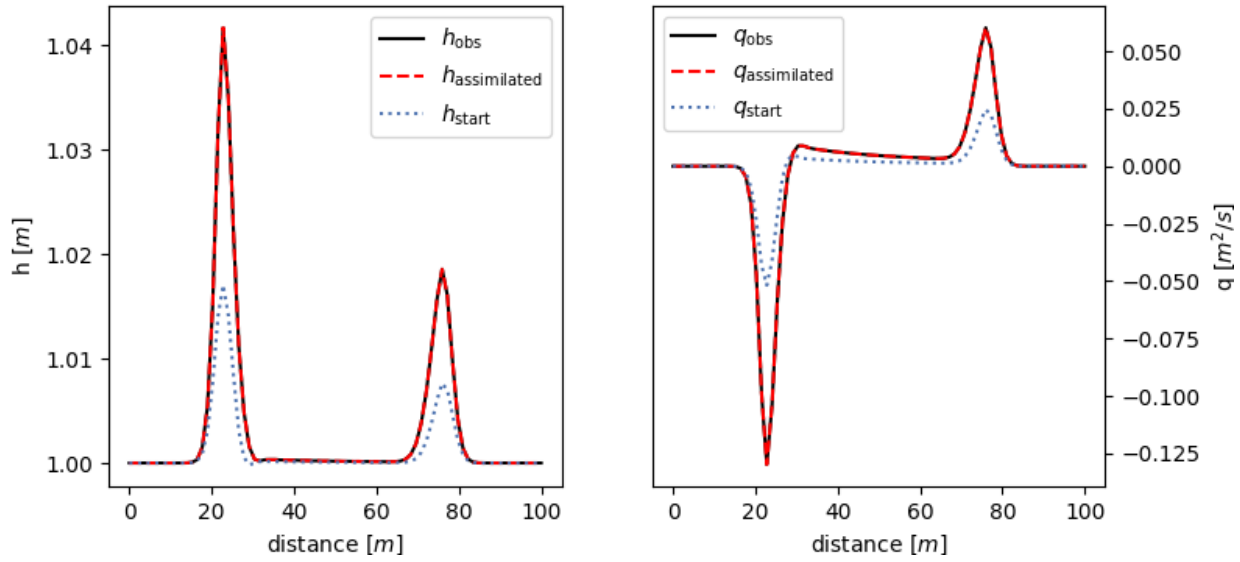

Figure 14: Assimilation of initial conditions: h (left) and q (right) at time $t = 8s$ using different initial conditions in the case of varying porosity

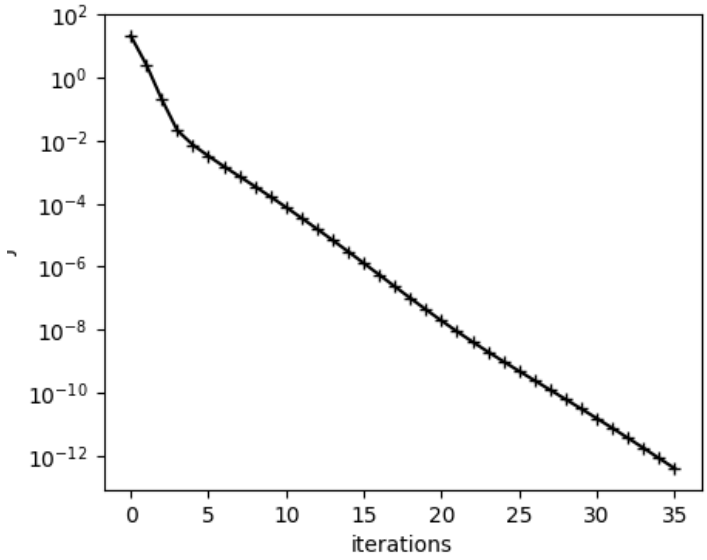

Figure 15: Assimilation of initial conditions: cost function J in the case of varying porosity

In this way we just need to compute the gradient of the cost function J with respect to ϕ_0 and we simply get:

$$\nabla_{\phi_0} J(x) = h(x,0)\lambda_h(x,0) + q(x,0)\lambda_q(x,0) \tag{50}$$

Once again we notice that the instant $t = 0$ corresponds to the instant $\tau = T$ in the reversed time of the adjoint equations.
Since in our case we assume a constant porosity, i.e. $\phi(x,t) = \phi(x,0)\ \forall t$, assimilating the initial conditions ϕ_0 is sufficient. The algorithm for the assimilation of porosity is summarized here:

Algorithm 4 Data assimilation for ϕ in SWE

while $k < N_{\max}$ and $\|\nabla_{\phi_0} J^k\| > \epsilon$ **do**
  Compute $h^k(x,t)$ and $q^k(x,t)$ from the discrete forward model
  Integrate the discrete adjoint model backwards from $t = T$ to $t = 0$ to obtain $\lambda_h^k(x,t)$ and $\lambda_q^k(x,t)$
  Compute $\nabla_{\phi_0} J^k(x) = h^k(x,0)\lambda_h^k(x,0) + q^k(x,0)\lambda_q^k(x,0)$ and $\|\nabla_{\phi_0} J^k\|_2$
  Optional: compute δ^k using the backtracking line search algorithm
  $\phi_0^{k+1}(x) = \phi_0^k(x) - \delta^k \nabla_{\phi_0} J^k(x)$
  $k \to k+1$
end while

A numerical experiment is performed to test the assimilation of porosity. Synthetic data used as observations are generated by running the forward solver with the following values for porosity and initial conditions and the parameters reported in Table 4 :

$$\begin{cases} \phi_{\text{target}}(x) = 1.0 - 0.3e^{-0.2(x-75.0)^2} \\ h(x,0) = 1.0 + 0.42e^{-0.1(x-50.0)^2} \\ q(x,0) = 0.0 \end{cases}$$

The initial guess is generated by using the same parameters and initial conditions except for the porosity ϕ which is set to $\phi_{\text{start}}(x) = 1$ in the whole domain. We perform the assimilation of porosity using both the fine-tuned fixed-step gradient descent algorithm and the backtracking line search algorithm adopting in both cases an assimilation window of 8 seconds. For the fixed-step algorithm, we take $\delta = 0.1$, while for the backtracking line search algorithm, we adopt the following hyper parameters: the large step length guess is set to 0.65, the shrinking factor α is set to 0.5, the constant c appearing in the Armijo condition is set to 0.1.

Table 4: Parameters for porosity assimilation experiment

Symbol	Meaning	Value
g	Gravitational acceleration	$9.81 m/s^2$
L	Length of the domain	100 m
Δx	Cell size	1.25 m
z_b	Uniform bed elevation	0.0 m

Figure 16 reports the results of the experiment. In particular we notice that the algorithm is able to retrieve the target porosity distribution. The trend of the cost function J with respect to the iterations confirms that the assimilated solution is indeed converging towards the observed solution. The backtracking line search outperforms the fixed fine-tuned δ gradient descent algorithm, that is, it returns a smaller value of the cost function J given the same run time.
As for the assimilation of the initial conditions, we test the effect of different assimilation window lengths. In particular, in Figure 17 we consider the following quantity:

$$e_\phi^k = \int_0^L |\phi^k(x) - \phi_{\text{target}}(x)| dx \tag{51}$$

Also in this case we can conclude that the longer the assimilation window, the more precise the assimilation of the parameter will be (given a fixed number of iterations).
We then repeat the same numerical experiment only changing the target porosity that now reads:
$\phi_{\text{target}} = 0.8 - 0.3e^{-0.2(x-75.0)^2}$.
We run the data assimilation algorithm with backtracking line search until convergence (i.e. until $\|\nabla_{\phi_0} J^k\|_2 < \epsilon = 10^{-6}$). The results are reported in Figure 18. We notice that, in this case, we are not able to retrieve the target porosity, however the trend of the cost function suggests that the assimilated solution is converging towards the observed solution nonetheless. The reason is that, with the data we have used, the shallow water model is "blind" to different constant value of ϕ. In fact, consider Eq. (29): suppose that ϕ doesn't depend on x and t. Moreover suppose that also the bed elevation z_b doesn't depend on x. Then $S_{0,x} = 0$. We can then bring ϕ out of the partial derivatives and divide both sides of the two equations by ϕ. We get:

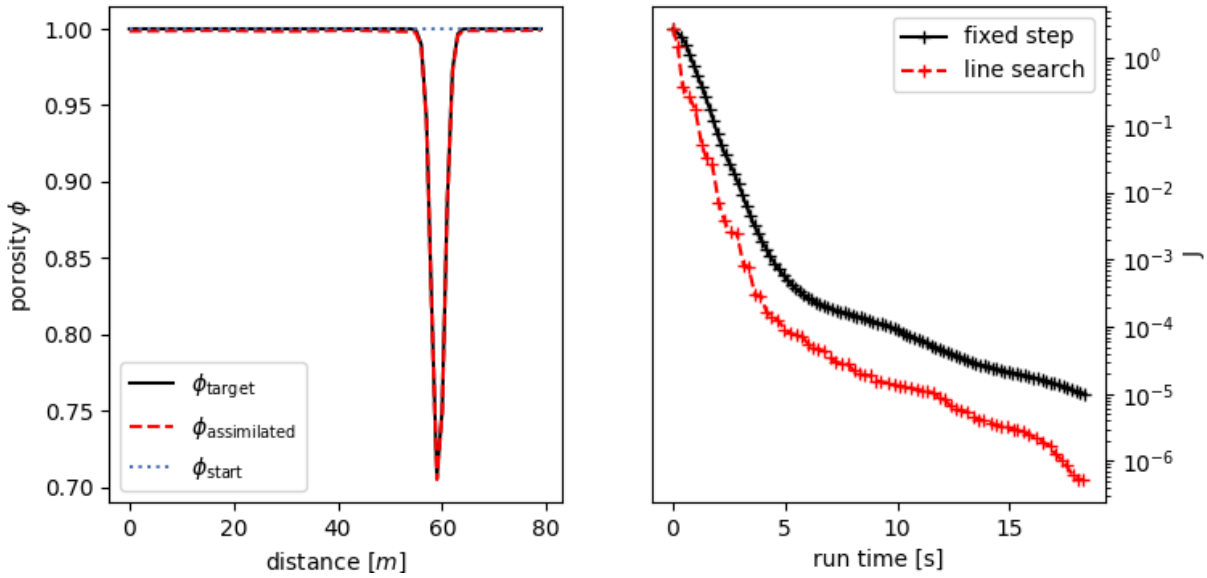

Figure 16: Assimilation of porosity ϕ: porosity distribution (left), cost function J (right)

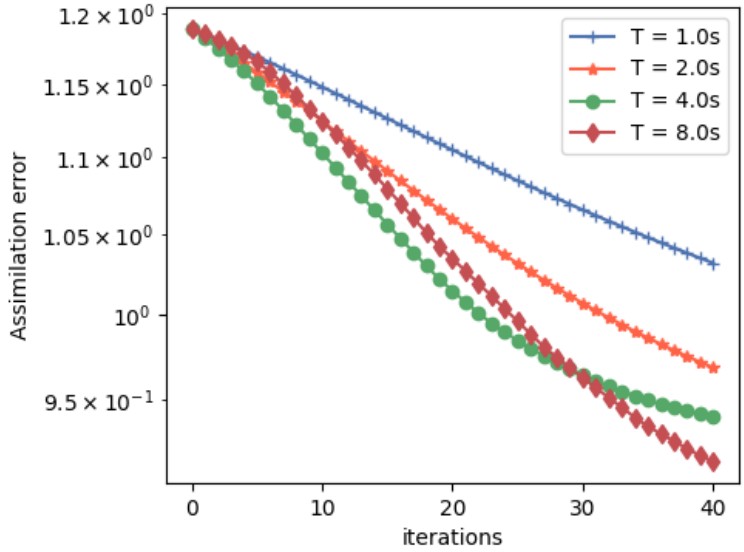

Figure 17: Assimilation of porosity ϕ: assimilation error with different time windows

$$\begin{cases} \dfrac{\partial h}{\partial t} + \dfrac{\partial q}{\partial x} = 0 & \text{(52a)} \\ \dfrac{\partial q}{\partial t} + \dfrac{\partial}{\partial x}\left(\dfrac{q^2}{h} + g\dfrac{h^2}{2}\right) = 0 & \text{(52b)} \end{cases}$$

which is equivalent to a uniform porosity $\phi = 1$. In other words, ϕ becomes just a rescaling factor of the whole equations. For this reason we can't expect to assimilate the right level of porosity in the portion of the domain where it is constant. Nonetheless, the target and the assimilated porosity have the same effect on the behavior of the solution. This is the reason why we see the cost function converging to zero. In a way, in this case the assimilation problem is ill-posed: there exist more than one porosity ϕ that minimize the cost function J. We can solve this issue by modifying the cost function J. Following the terminology introduced in Section 4.1, we introduce the background state for porosity ϕ_b. This background state can be estimated by the standard computation of the area available for the flow thanks to satellite data. In order to penalize the distance between the assimilated porosity and the background state, we add the regularizing term J_b to the cost

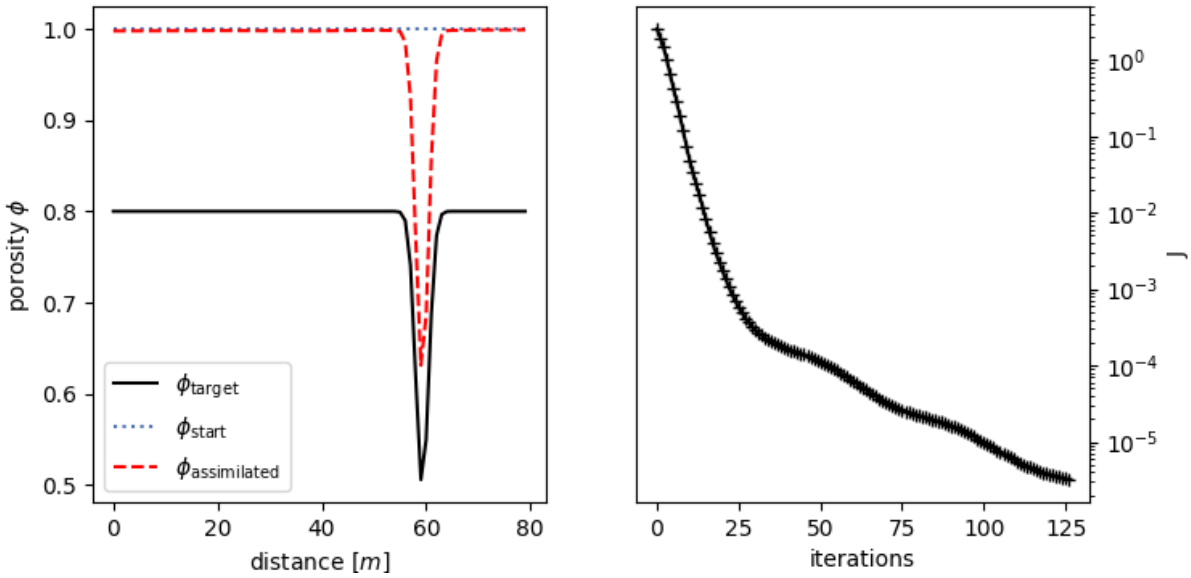

Figure 18: Porosity distributions before and after assimilation (left), cost function J (right)

function. Moreover we can modify the weights associated to each term in the cost function. Let w_1, w_2, and $w_3 \in \mathbb{R}^+$ be such weights. We impose their sum to be equal to 1. While this constraint is not strictly necessary, it is imposed for clarity: it allows to immediately understand the relative contribution of each term of the cost function. The cost function now reads:

$$J = w_1 \int_0^T \int_0^L \Big(h(x,t) - h_{\text{obs}}(x,t)\Big)^2 dxdt + w_2 \int_0^T \int_0^L \Big(q(x,t) - q_{\text{obs}}(x,t)\Big)^2 dxdt + \\ + w_3 \int_0^T \int_0^L \Big(\phi(x) - \phi_b(x)\Big)^2 dxdt \tag{53}$$

with

$$\sum_{i=1}^{3} w_i = 1 \tag{54}$$

As a consequence, the gradient of the cost function now reads:

$$\nabla_{\phi_0} J(x) = 2w_1 \cdot h(x,0)\lambda(x,0) + 2w_2 \cdot q(x,0)\lambda_q(x,0) + 2w_3 \cdot (\phi(x) - \phi_b(x)) \tag{55}$$

We repeat the previous numerical experiment four times using the modified cost function given by (53) and the weights reported in Table 5. In particular, the porosity background state is set to $\phi_b(x) = 0.8$.

Table 5: Weight choices

Test name	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$
A	0.0	0.5	0.5
B	0.5	0.5	0.0
C	0.33	0.33	0.33
D	0.5	0.2	0.3

The results of the experiment are presented in Figure 19. The choice of the weights significantly influences the assimilation process. As expected, if we remove the penalty on the deviation from the background state (i.e. we set $w_3 = 0$) we are not able to retrieve the right level of the porosity, on the other hand, if we do not penalize the error on the water level h (i.e. we set $w_1 = 0$), we retrieve the background state and not the target porosity. This experiment highlights the necessity of including all terms in the cost function, however the problem of optimally tuning the weights w_i has not been examined in this thesis and could be subject of future developments.

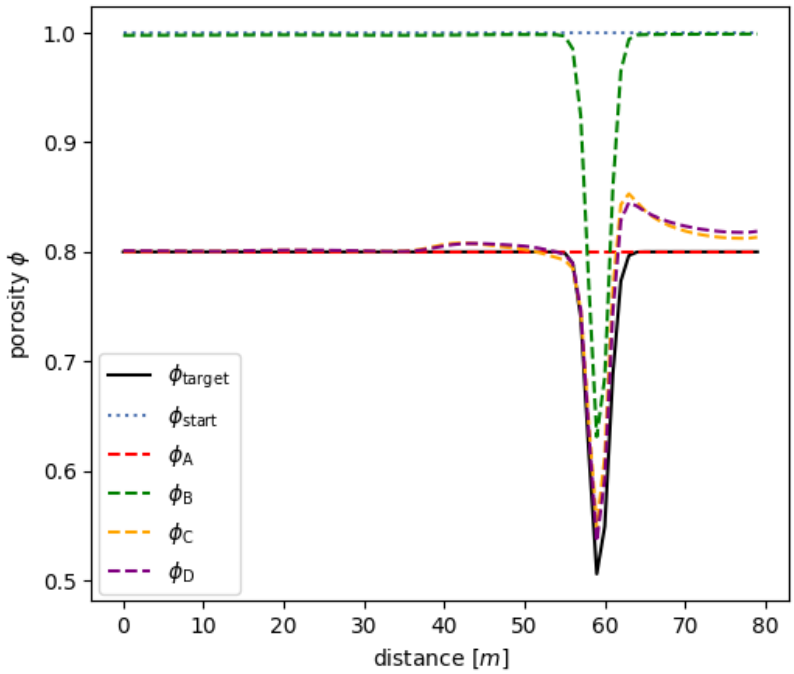

Figure 19: Assimilation of porosity ϕ: assimilation with different weight choices

6. Conclusions

In this thesis, we addressed the problem of variational data assimilation applied to the porous shallow water equations, with the aim of improving the spatial estimation of porosity by integrating observational data into the numerical model. The assimilation process was formulated as the minimization of a cost function that quantifies the discrepancy between numerical predictions and observations, with the porous shallow water equations serving as a constraint for the optimization problem. The minimization process is based on a gradient descent algorithm. In order to compute the gradient in a computationally efficient way, we derived the adjoint equations corresponding to the forward model and developed two dedicated solvers for the one-dimensional forward and adjoint problems, respectively. Indeed, the adjoint shallow-water equations cannot be recast in conservative form, thus they shall be solved using different numerical schemes than those used for the classical shallow water equations. The numerical scheme for the adjoint problem was based on Roe's one-step fluctuation splitting method that generalizes the classical flux difference splitting for a general non-conservative hyperbolic system of equations. The assimilation procedure was implemented through an iterative algorithm that alternates between the forward and adjoint solvers in order to compute the gradient of the cost function with respect to the parameter to be assimilated. The proposed method was successfully tested for assimilating both the initial conditions and the spatial distribution of porosity, demonstrating the feasibility and effectiveness of the approach. Notice that, although the numerical experiments were all performed in a one-dimensional setting, all the procedures presented can naturally be extended to the two-dimensional case with just an increased implementation complexity. Nonetheless, this study does have some limitations. In particular, we assumed that observational data are available at every point in space and at each time step, which is rarely the case in real-world applications. While the cost function can easily be modified to accommodate sparse data (for instance by replacing the computation of the integral with a sum over available data points), the quality of the data in terms of their size, location in space and frequency in time plays a major role in the well-posedness of the assimilation problem [43, 44]. Moreover, we assumed observational data to be free of noise and error. Lastly, this approach to variational data assimilation, although computationally efficient, is problem specific [30, 45]. In particular, it requires to derive and numerically solve the corresponding adjoint equations, which vary with each different forward model. This manual process could be avoided by exploiting automatic differentiation techniques to compute the gradient of the cost function from the discrete forward solver, as demonstrated, for example, in the development of DassFlow [32]. Another promising option to overcome the need of the derivation of the adjoint equations is to train differentiable surrogate models (e.g. neural networks) to approximate the forward model, as done, for example, by Chennault et al. [46].

References

[1] LEMON team presentation. https://team.inria.fr/lemon/.

[2] Geir Evensen, Femke C Vossepoel, and Peter Jan van Leeuwen. *Data assimilation fundamentals*. Springer Textbooks in Earth Sciences, Geography and Environment. Springer Nature, Cham, Switzerland, 2022 edition, April 2022.

[3] P. del Moral. Nonlinear filtering using random particles. *Theory of Probability & Its Applications*, 40(4):690–701, 1996.

[4] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[5] G. Kontarev. *The adjoint equation technique applied to meteorological problems*. Shinfield Park, Reading, 09/1980 1980.

[6] Dishant Pandya, Bhasha Vachharajani, and Rohit Srivastava. A review of data assimilation techniques: Applications in engineering and agriculture. *Materials Today: Proceedings*, 62:7048–7052, 2022. International Conference on Additive Manufacturing and Advanced Materials (AM2).

[7] Dan Givoli. A tutorial on the adjoint method for inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 380:113810, 2021.

[8] R. Fablet, B. Chapron, L. Drumetz, E. Mémin, O. Pannekoucke, and F. Rousseau. Learning variational data assimilation models and solvers. *Journal of Advances in Modeling Earth Systems*, 13(10), 2021.

[9] Philippe Courtier and Olivier Talagrand. Variational assimilation of meteorological observations with the direct and adjoint shallow water equations. *European Center for Medium-Range Weather Forecasts*, 1988.

[10] Mike Fisher, Jorge Nocedal, Yannick Trémolet, and Stephen J. Wright. Data assimilation in weather forecasting: a case study in pde-constrained optimization. *Optim Eng*, page 10: 409–426, 2009.

[11] Brett F. Sanders and Nikolaos D. Katopodes. Adjoint sensitivity analysis for shallow-water wave control. *Journal of Engineering Mechanics*, 2000.

[12] Byunghyun Kim, Brett F. Sanders, Famiglietti James, and Vincent Guinot. Urban flood modeling with porous shallow-water equations: a case study of model errors in the presence of anisotropic porosity. *Journal of Environmental Hydrology*, pages 523, pp.680–692, 2015.

[13] H. Ozdemir, C.C. Sampson, G.A.M. de Almeida, and P.D. Bates. Evaluating scale and roughness effects in urban flood modelling using terrestrial lidar data. *Hydrol. Earth Syst. Sci.*, pages 17, 4015–4030, 2013.

[14] Vincent Guinot and Sandra Soares-Frazao. Flux and source term discretization in two-dimensional shallow water models with porosity on unstructured grids. *Int. J. Numer. Meth. Fluids*, pages 309–345, 2006.

[15] V. Guinot, C. Delenne, A. Rousseau, and O. Boutron. Flux closures and source term models for shallow water models with depth-dependent integral porosity. *Advances in Water Resources*, 122:1–26, 2018.

[16] Vincent Guinot, Brett F. Sanders, and Jochen E. Schubert. Dual integral porosity shallow water model for urban flood modelling. *Advances in Water Resources*, 103:16–31, 2017.

[17] S. Soares Frazao and G. Testa. The toce river test case: Numerical results analysis. *Proceedings of the 3rd CADAM Workshop, Milan, Italy*, 1999.

[18] Joao Guilherme Caldas Steinstraesser, Carole Delenne, Vincent Guinot, Pascal Finaud-Guyot, Joseph Luis Kahn Casapia, et al. Sw2d-lemon: A new software for upscaled shallow water modeling. *SimHydro 2021: Models for complex and global water issues – Practices and expectations*, Jun 2021.

[19] Benjamin Seibold. Numerical methods for partial differential equations. Graduate level course at MIT, 2009. https://ocw.mit.edu/courses/18-336-numerical-methods-for-partial-differential-equations-spring-2009/.

[20] Alfio Quarteroni. *Modellistica numerica per problemi differenziali*. Springer, 2016.

[21] Sarmad Ghader and Jan Nordström. Revisiting well-posed boundary conditions for the shallow water equations. *Dynamics of Atmospheres and Oceans*, 66:1–9, 2014.

[22] Marie-Odile Bristeau and Benoit Coussin. Boundary conditions for the shallow water equations solved by kinetic schemes. *Inria Research Report-4282*, 2001.

[23] A. McDonald. Transparent boundary conditions for the shallow-water equations: Testing in a nested environment. *Monthly Weather Review*, 131(4):698 – 705, 2003.

[24] E.F. Toro. Chapter 2 - the riemann problem: Solvers and numerical fluxes. In Rémi Abgrall and Chi-Wang Shu, editors, *Handbook of Numerical Methods for Hyperbolic Problems*, volume 17 of *Handbook of Numerical Analysis*, pages 19–54. Elsevier, 2016.

[25] E.F. Toro, M. Spruce, and W. Speares. Restoration of the contact surface in the hll-riemann solver. *Shock Waves-Springer*, pages 4:25–34, 1994.

[26] SF David. Simplified second-order godunov-type methods. *SIAM J Sci and Star Comput*, page 9:445, 1988.

[27] B Einfeldt. On godunov-type methods for the euler equations with general equation of state. *Proc Second Internat Conference on Hyperbolic Problems*, 1988.

[28] Erik Andersson and Jean-Noël Thépaut. Ecmwf's 4d-var data assimilation system – the genesis and ten years in operations. *ECMWF Newsletter No. 115*, pages 8–12, 2008.

[29] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data Assimilation: Methods, Algorithms, and Applications*. SIAM, 2017.

[30] Florence Rabier and Zhiquan Liu. Variational data assimilation: Theory and overview. *European Center for Medium-Range Weather Forecasts*, 2003.

[31] J. Monnier. Data assimilation. Course at Toulouse University, 2013.

[32] Frédéric Couderc, Ronan Madec, Jerome Monnier, and Jean-Paul Vila. DassFlow-Shallow, Variational Data Assimilation for Shallow-Water Models: Numerical Schemes, User and Developer Guides. Research report, University of Toulouse, CNRS, IMT, INSA, ANR, 2013.

[33] Eric Blayo, Emmanuel Cosme, and Arthur Vidard. Introduction to data assimilation. Course at Grenoble University, 7 - 11 January 2019. https://dataassim.sciencesconf.org/.

[34] A. S. Lawless and N. K. Nichols. Inner-loop stopping criteria for incremental four-dimensional variational data assimilation. *Monthly Weather Review*, 134(11):3425 – 3435, 2006.

[35] Florence Rabier, Jean-Noel Thépaut, and Philippe Courtier. Extended assimilation and forecast experiments with a four-dimensional variational assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 124(550):1861–1887, 1998.

[36] A. T. Weaver, J. Vialard, and D. L. T. Anderson. Three- and four-dimensional variational assimilation with a general circulation model of the tropical pacific ocean. part i: Formulation, internal diagnostics, and consistency checks. *Monthly Weather Review*, 131(7):1360 – 1378, 2003.

[37] Marc Honnorat, Joel Marin, Jerome Monnier, and Xijun Lai. Dassflow v1.0: a variational data assimilation software for 2d river flows. *Inria research report*, 2007.

[38] Brett F. Sanders and Scott F. Bradford. High-resolution, monotone solution of the adjoint shallow-water equations. *Int. J. Numer. Meth. Fluids*, pages 38:139–161, 2002.

[39] P. Glaister. Approximate riemann solution of the two-dimensional shallow-water equations. *Journal of Engineering Mathematics*, page 24:45–53, 1990.

[40] RJ. LeVeque. Wave propagation algorithms for multidimensional hyperbolic systems. *Journal of Computational Physics*, page 131:327–353, 1997.

[41] Philip Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

[42] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Science Business Media, 2000.

[43] Marta D'Elia, Lucia Mirabella, Tiziano Passerini, Mauro Perego, Marina Piccinelli, Christian Vergara, and Alessandro Veneziani. *Applications of variational data assimilation in computational hemodynamics*, pages 363–394. Springer Milan, Milano, 2012.

[44] J. L. Lions. *Exact Controllability for Distributed Systems. Some Trends and Some Problems*, pages 59–84. Springer Netherlands, Dordrecht, 1991.

[45] Oliver Brenner, Justin Plogmann, Pasha Piroozmand, and Patrick Jenny. A variational data assimilation approach for sparse velocity reference data in coarse rans simulations through a corrective forcing term. *Computer Methods in Applied Mechanics and Engineering*, 427:117026, July 2024.

[46] Austin Chennault, Andrey A. Popov, Amit N. Subrahmanya, Rachel Cooper, Ali Haisam Muhammad Rafid, Anuj Karpatne, and Adrian Sandu. Adjoint-matching neural network surrogates for fast 4d-var data assimilation, 2022.

A. Appendix A

Computation of functional derivatives
Just as standard derivatives relate a change in the function to a change in its argument, functional derivative relates a change in a functional to a change in a function on which the functional depends. Consider the function $u = u(x)$ and the functional $J(u) = \int_0^1 u^2(x)dx$. We want to compute the functional derivative of J with respect to u, namely $\frac{\delta J}{\delta u}$.

Let B be a Banach space, F be a functional defined on B. The differential of F at point $u \in B$ is defined $\forall \eta \in B$ as:

$$\delta F(u, \eta) = \lim_{\varepsilon \to 0} \frac{F(u + \varepsilon\eta) - F(u)}{\varepsilon} \tag{56}$$

Suppose that the domain of F is the space of differentiable functions u defined on some domain Ω. If $\delta F(u, \eta)$ can be written as

$$\delta F(u, \eta) = \int_\Omega \frac{\delta F}{\delta u}(x)\eta(x)dx \tag{57}$$

we say that $\frac{\delta F}{\delta u}$ is the functional derivative of F at u.
Applying this definition to the functional J we get:

$$\delta J = \lim_{\varepsilon \to 0} \frac{J(u + \varepsilon\eta) - J(u)}{\varepsilon} = \lim_{\varepsilon \to 0} \int_0^1 [\varepsilon\eta^2(x) + 2u(x)\eta(x)]dx = \int_0^1 2u(x)\eta(x)dx \tag{58}$$

$$\delta J = \int_0^1 2u(x)\eta(x)dx = \int_0^1 \frac{\delta F}{\delta u}\eta(x)dx \tag{59}$$

In the end we get:

$$\frac{\delta J}{\delta u} = 2u(x) \tag{60}$$

Abstract in lingua italiana

Questa tesi è stata svolta presso il centro di ricerca Inria di Montpellier all'interno del team LEMON, un team interdisciplinare che che si occupa di sviluppare modelli accurati e computazionalmente poco costosi per i processi naturali che si verificano nell'area litoranea [1]. I modelli di alluvione su scala urbana sono sempre più studiati nel contesto della prevenzione del rischio, a tal fine, da diversi anni, il team LEMON sviluppa i cosiddetti modelli di porosità per le equazioni delle acque poco profonde, per fornire simulazioni rapide e su larga scala di questi fenomeni. Le equazioni porose per le acque poco profonde estendono il modello classico introducendo la porosità per rappresentare gli effetti a scala sotto-griglia causati da ostacoli come edifici e vegetazione nelle pianure alluvionali. Questo lavoro esplora l'uso dell'assimilazione variazionale dei dati per migliorare la stima della distribuzione spaziale della porosità e, di conseguenza, migliorare l'accuratezza nella previsione del flusso di piena. L'assimilazione variazionale dei dati viene implementata utilizzando un algoritmo di ottimizzazione basato sulla discesa del gradiente. Tale gradiente viene calcolato in modo efficiente attraverso il problema aggiunto. Presentiamo sia la formulazione diretta che quella aggiunta, insieme ai rispettivi risolutori numerici unidimensionali. Gli esperimenti numerici dimostrano l'efficacia di questo approccio per migliorare la stima della porosità. Sottolineiamo l'importanza di affrontare la buona posizione del problema di assimilazione per garantire risultati affidabili.

Parole chiave: assimilazione dei dati variazionale, equazioni delle acque poco profonde, porosità, problema aggiunto

Acknowledgements

Desidero esprimere la mia più sincera gratitudine alla Professoressa Scotti, relatrice di questa tesi, per la sua disponibilità e per il supporto fornitomi durante questi mesi.

Je tiens à remercier tous les membres de l'équipe LEMON de l'antenne Inria de l'Université de Montpellier pour leur accueil et pour m'avoir permis de participer à leurs activités de recherche. Je remercie également Pascal et Lilas pour leur disponibilité. Je remercie particulièrement à Antoine pour sa gentillesse et sa disponibilité à répondre à toutes mes questions.