



Degree Project in Technology

First Cycle, 15 credits

# **Simulating the Shallow Water Equations**

with an SBP-SAT scheme

**OLIVER BIELKE & NICLAS LI**

# Abstract

This thesis focuses on the derivation and numerical solution of the shallow water equations (SWE), which is a system of partial differential equations used to model fluid flow in situations where the horizontal length scales are much larger than the vertical depth. Since the SWE generally cannot be solved analytically, a high-order numerical method is introduced. Specifically, the Summation-By-Parts (SBP) framework combined with the Simultaneous Approximation Term (SAT) technique is presented and implemented to solve the equations with accuracy and stability. The scheme is implemented in Python. Different test cases were used to analyze the accuracy and stability of the implemented scheme. It was concluded that the scheme worked well for basic wave behaviors, while it struggled with more challenging scenarios involving complex water flow and shocks.

## Acknowledgments

We want to thank our supervisor Roman Stuhlmacher for suggesting an interesting and engaging topic for our bachelor thesis project. It has been a pleasure working with him, and we greatly appreciate his guidance, support, and valuable discussions throughout the project, especially when addressing the various challenges that arose along the way.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Background . . . . .	5
1.1.1	The Shallow Water Equations . . . . .	5
1.1.2	Numerical Method . . . . .	6
1.2	Purpose and Aim . . . . .	6
1.3	Outline . . . . .	6
<b>2</b>	<b>Numerical Scheme and Implementation</b>	<b>7</b>
2.1	System of equations . . . . .	7
2.1.1	Flux splitting . . . . .	8
2.2	Numerical method . . . . .	8
2.2.1	Finite Differences . . . . .	9
2.2.2	Summation By Parts . . . . .	9
2.2.3	Upwind SBP-operator . . . . .	10
2.2.4	Simultaneous Approximation Term . . . . .	12
2.3	Discretization . . . . .	12
2.3.1	Discretization of the Flux term . . . . .	13
2.3.2	Discretization of the Source Term . . . . .	14
2.3.3	Boundary treatment using SAT . . . . .	14
2.4	Time discretization and CFL condition . . . . .	15
2.5	Programming . . . . .	15
<b>3</b>	<b>Results</b>	<b>16</b>
3.1	Lake at rest . . . . .	16
3.2	Dam break . . . . .	18
3.3	Gaussian pulse . . . . .	19
3.4	Gaussian pulse over slope . . . . .	20
3.5	Subcritical flow over bump . . . . .	21
<b>4</b>	<b>Discussion</b>	<b>24</b>
4.1	Effects of Different Variables . . . . .	24
4.1.1	Effect of Initial Conditions . . . . .	24
4.1.2	Effect of Boundary Conditions . . . . .	25
4.1.3	Sensitivity to spatial resolution, $\Delta x$ . . . . .	25
4.1.4	Sensitivity to Time step, $\Delta t$ . . . . .	25
4.2	Error Analysis . . . . .	26
4.3	Convergence Behavior . . . . .	27
4.4	Future improvements . . . . .	27
<b>5</b>	<b>Conclusion</b>	<b>28</b>
<b>6</b>	<b>References</b>	<b>29</b>

# 1 Introduction

The shallow water equations (SWE), originally introduced by Saint-Venant in 1871, are a special case of the Navier-Stokes equations. They are commonly used to model various types of water flow, particularly when the wavelength of the phenomenon is significantly greater than its amplitude. Typical applications include river flow, flooding events, dam breaks, and tsunamis.[1]

The differential equations that make up the shallow water equations cannot be solved analytically. Therefore, the use of numerical methods is necessary in order to study fluid motion in accordance with shallow water theory. This project seeks to develop a simulation of water waves based, on shallow water theory, with a focus on investigating different physical phenomena by altering the boundary conditions.

## 1.1 Background

Shallow water flow describes fluid motion where two key conditions are met: the horizontal length scale is much larger than the vertical depth, and vertical velocities are sufficiently small. Together, these conditions justify the use of depth-averaged quantities to represent the flow. Under these assumptions, the full three-dimensional Navier-Stokes equations can be significantly simplified, reducing the problem to an effective one- or two-dimensional model.[6] In shallow water, wave speed becomes independent of wavelength and depends only on the local water depth. The characteristic wave speed is given by:  $c = u \pm \sqrt{gh}$  where  $u$  is the background flow velocity,  $g$  is the gravitational constant and  $h$  is the depth of the water. The term  $\sqrt{gh}$  represents the speed of small surface waves relative to the moving fluid. In this study, we focus on subcritical flow where  $|u| < \sqrt{gh}$ , allowing waves to propagate both upstream and downstream. This property plays a critical role in how disturbances and information travel through shallow water systems.[7]

### 1.1.1 The Shallow Water Equations

In the 1-D case, where propagation occurs in only one horizontal direction  $x$ -direction, the shallow water equations consists of two equations: one for mass conservation given by:

$$h_t + [hu]_x = 0 \tag{1}$$

and one for momentum conservation given by:

$$[hu]_t + [hu^2 + \frac{1}{2}gh^2]_x + ghb_x = 0, \tag{2}$$

where  $g$  is the gravitational constant.

The variables solved for are the water height  $h(x, t)$ , which is in relation to the bottom topography  $b(x)$  and the vertically averaged velocity of the water  $u(x, t)$  in the horizontal  $x$ -direction (see Figure 1).[1]

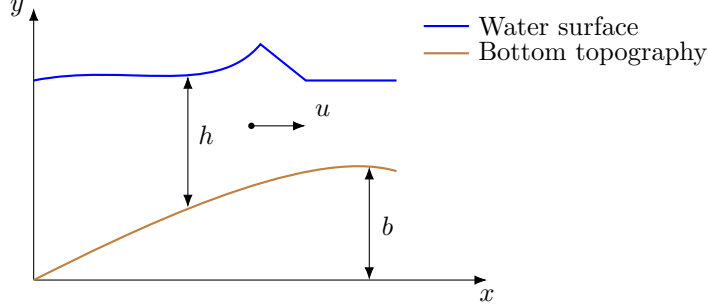


Figure 1: A one-dimensional diagram representing the shallow water model.

### 1.1.2 Numerical Method

Numerical methods are needed in order to solve the SWE, which consists of a system of non-linear partial differential equations. In this project a high order method called the Summation-By-Parts (SBP) Simultaneous Approximation Term (SAT) scheme is used.

## 1.2 Purpose and Aim

The purpose of this thesis is to study the SWE and investigate their numerical solution using the SBP-SAT scheme. Additionally, the thesis aims to implement a computational model capable of simulating a variety of test cases under different boundary conditions, enabling the analysis of different physical phenomena that can be modeled under shallow water theory. The simulations are designed to evaluate the accuracy, stability, and physical realism of the scheme across different scenarios, including both smooth and discontinuous solutions.

## 1.3 Outline

In 2 Numerical Scheme and Implementation, the numerical scheme used to solve the SWE is introduced, followed by an explanation of how it is applied for discretization. 3 Results presents results for both smooth wave propagation and shock-like behaviors to test the scheme's accuracy and robustness. 4 Discussion discusses the findings, highlights limitations, and suggests improvements of the used numerical scheme.

## 2 Numerical Scheme and Implementation

In this section the numerical scheme used to solve the SWE is introduced and implemented. First, the SWE (1) and (2) are reformulated into a set of equations that are required for the application of the numerical scheme. This reformulated system is then discretized using the numerical SBP-SAT scheme, which is also introduced and explained in this section.

### 2.1 System of equations

Before applying the numerical scheme, the shallow water equations (SWE) are first combined into a single system of equations. The procedure for this reformulation follows the approach described in [1]. Equations (1) and (2) are combined into a system of equations:

$$\begin{bmatrix} h \\ hu \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix}_x + gh \begin{bmatrix} 0 \\ b \end{bmatrix}_x = 0, \quad (3)$$

which can be rewritten into:

$$\mathbf{q}_t + \mathbf{F}(\mathbf{q})_x + \mathbf{G}(\mathbf{q}) = \mathbf{0}, \quad (4)$$

where  $\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} h \\ hu \end{bmatrix}$ ,  $\mathbf{F}(\mathbf{q}) = \begin{bmatrix} q_2 \\ \frac{q_2^2}{q_1} + \frac{1}{2}gq_1^2 \end{bmatrix}$  and  $\mathbf{G}(\mathbf{q}) = \begin{bmatrix} 0 \\ gq_1b_x \end{bmatrix}$

Here  $\mathbf{F}(\mathbf{q})$  is called the flux function and it describes how the quantity  $\mathbf{q}$  is transported through space, i.e. the transport of mass and momentum.  $\mathbf{G}(\mathbf{q})$  is called the source term and it accounts for the effect the bottom topology has on the fluid momentum.

The system (4) can be further rewritten into Quasi-linear form:

$$\mathbf{q}_t + \mathbf{A}(\mathbf{q})\mathbf{q}_x + \mathbf{C}\mathbf{q} = \mathbf{0}, \quad (5)$$

where  $\mathbf{A}(\mathbf{q})$  is the Jacobian matrix of  $\mathbf{F}(\mathbf{q})$ :

$$\mathbf{A}(\mathbf{q}) = \mathbf{F}'(\mathbf{q}) = \begin{bmatrix} \frac{\partial F_1}{\partial q_1} & \frac{\partial F_1}{\partial q_2} \\ \frac{\partial F_2}{\partial q_1} & \frac{\partial F_2}{\partial q_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ gq_1 - \frac{q_2^2}{q_1^2} & \frac{2q_2}{q_1} \end{bmatrix} \text{ and } \mathbf{C} = \begin{bmatrix} 0 & 0 \\ qb_x & 0 \end{bmatrix}$$

The eigenvalues of  $\mathbf{A}(\mathbf{q})$  describe the characteristic wave speeds of the system, determining how information propagates through the domain. In the context of the shallow water equations, small surface perturbations travel at speeds  $u \pm \sqrt{gh}$ . These speeds naturally emerge from the eigenvalues obtained by solving the characteristic equation for  $\mathbf{A}(\mathbf{q})$ :

$$\begin{aligned} \det(\mathbf{A} - \Lambda \mathbf{I}) &= 0 \\ \Rightarrow \Lambda_{\pm} &= \frac{q_2}{q_1} \pm \sqrt{gq_1} = u \pm \sqrt{gh} \end{aligned}$$

In the case of subcritical flow where,  $|u| < \sqrt{gh}$ , one eigenvalue will be positive and one will be negative, which confirms that the waves can travel both in the leftward and rightward direction (upstream and downstream) in the domain.

Furthermore, matrix  $\mathbf{A}$  can be diagonalized and expressed in terms of its eigenvalues and eigenvectors, which will be useful in later analysis:

$$\mathbf{A} = \mathbf{W} \mathbf{\Lambda} \mathbf{W}^{-1},$$

$$\text{where } \mathbf{\Lambda} = \begin{bmatrix} \Lambda_+ & 0 \\ 0 & \Lambda_- \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} 1 & 1 \\ \Lambda_+ & \Lambda_- \end{bmatrix}$$

### 2.1.1 Flux splitting

The Jacobian matrix  $\mathbf{A}$ , which governs the flux of the system can be split into two parts, one with positively propagating characteristics ( $\mathbf{A}_+$ ) and one with negatively propagating characteristics ( $\mathbf{A}_-$ ). This process, known as flux splitting, is introduced to separate left- and right-going wave components, which will later be useful for applying the numerical method. In this study, it is carried out using the *Lax-Friedrichs* flux-splitting method in the following way:

$\mathbf{A}_{\pm} = \frac{1}{2}(\mathbf{A} \pm \alpha \mathbf{I}_2)$ , where  $\alpha$  is absolute value of the largest-magnitude eigenvalue of  $\mathbf{A}$ .

Note that,  $\mathbf{A} = \mathbf{A}_+ + \mathbf{A}_-$ , and equation (5) can now be expressed as:

$$\mathbf{q}_t + \mathbf{A}(\mathbf{q})_+ \mathbf{q}_x + \mathbf{A}(\mathbf{q})_- \mathbf{q}_x + \mathbf{C}\mathbf{q} = 0 \quad (6)$$

## 2.2 Numerical method

The spatial discretization is done using the SBP-SAT technique. By combining summation-by-parts (SBP) operators with the simultaneous approximation term (SAT) method to enforce boundary conditions, a robust and high-order finite difference approach can be developed for solving well-posed initial boundary value problems efficiently.[1] The theoretical foundation of this method is presented below.

### 2.2.1 Finite Differences

A finite-difference method solves differential equations by replacing continuous derivatives with discrete approximations, using values of the function at a set of grid points. The spatial and temporal domains are discretized into a finite number of grid points, and the values at each point are estimated using information from neighboring points. This approach replaces continuous derivatives with algebraic approximations, allowing differential equations to be solved numerically.

As an example, consider a spatial discretization,  $\mathbf{u}$ , of a differentiable function  $\mathcal{U}(x)$  over the domain  $x_L \leq x \leq x_R$ . The domain is divided into  $N + 1$  equally spaced nodes indexed from 0 to  $N$ :  $[x_0, \dots, x_N]$ , and  $\mathbf{u} = [u_0, \dots, u_N]$ , where  $u_i = \mathcal{U}(x_i)$ . Using a second-order accurate central finite difference scheme, the derivative at an interior point  $x_i$  can be approximated as:

$$\mathcal{U}'(x_i) \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x},$$

where  $\Delta x = \frac{x_R - x_L}{N}$  is the uniform spacing between the nodes. At the boundaries, the central difference formula is not applicable. Therefore, a first-order accurate one-sided difference is used instead. Specifically, a forward difference is applied at the left boundary and a backward difference at the right boundary:

$$\mathcal{U}'(x_0) \approx \frac{u_1 - u_0}{\Delta x}, \quad \mathcal{U}'(x_N) \approx \frac{u_N - u_{N-1}}{\Delta x}.$$

The entire discrete approximation of the derivative can then be expressed as a matrix-vector product involving a  $(N + 1) \times (N + 1)$  finite difference matrix  $D$  acting on the discrete function  $\mathbf{u}$ :

$$\mathcal{U}'(x) \approx D\mathbf{u} = \frac{1}{2\Delta x} \begin{bmatrix} -2 & 2 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -2 & 2 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix},$$

where  $D$  has a second-order accurate central difference stencil in the interior.

### 2.2.2 Summation By Parts

A Summation by Parts operator (SBP) approximating the first derivative is a finite difference matrix defined in [3] as follows:

**Definition 2.1:** A difference operator  $D_1 = H^{-1}(Q + \frac{B}{2})$  approximating  $\frac{\partial}{\partial x}$ , using a  $p$ th-order accurate interior stencil, is said to be a  $p$ th order diagonal-norm first-derivative SBP operator if the diagonal matrix  $H$  defines a discrete norm, and  $Q + Q^T = 0$ .

Here, the matrix  $B = \text{diag}(-1, 0, \dots, 0, 1)$  and the matrix  $H$  defines a discrete inner product and norm in the following way:

$$(u, v)_H = \mathbf{u}^T H \mathbf{v}, \quad \|\mathbf{u}\|_H^2 = \mathbf{u}^T H \mathbf{u}$$

The different matrices used in constructing the discrete SBP operator  $D_1$  are defined in a way so that it mimics the results obtained in continuous case using the integration by parts method.

This is demonstrated in a way similar to what is done in [4]. Let us consider differentiable functions  $\mathcal{U}(x, t)$  and  $\mathcal{V}(x, t)$  defined in the domain:  $x_L \leq x \leq x_R$ , then applying integration by parts we find:

$$\int_{x_L}^{x_R} \frac{\partial \mathcal{U}}{\partial x} \mathcal{V} dx = [\mathcal{U} \mathcal{V}]_{x_L}^{x_R} - \int_{x_L}^{x_R} \mathcal{U} \frac{\partial \mathcal{V}}{\partial x} dx$$

To replicate this behavior in the discrete setting, instead consider a discretization in space (using the same domain as before) with  $N+1$  equally spaced nodes indexed from 0 to  $N$  such that:  $\mathbf{u} = [u_0, \dots, u_N]^T$  and  $\mathbf{v} = [v_0, \dots, v_N]^T$ .

The derivative and inner product are approximated in its discrete forms as:

$$\frac{\partial \mathcal{U}}{\partial x} \approx D_1 \mathbf{u}, \quad \int_{x_L}^{x_R} \mathcal{U} \mathcal{V} dx \approx \mathbf{u}^T H \mathbf{v} = (\mathbf{u}, \mathbf{v})_H$$

The discrete equivalent of integration by parts is then given by:

$$\begin{aligned} (D_1 \mathbf{u}, \mathbf{v})_H + (\mathbf{u}, D_1 \mathbf{v})_H &= (H^{-1}(Q + \frac{B}{2})\mathbf{u})^T H \mathbf{v} + \mathbf{u}^T H (H^{-1}(Q + \frac{B}{2})\mathbf{v}) \\ &= ((Q + \frac{B}{2})\mathbf{u})^T H H^{-1} \mathbf{v} + \mathbf{u}^T H H^{-1} (Q + \frac{B}{2})\mathbf{v} \\ &= \mathbf{u}^T (Q^T + \frac{B}{2})\mathbf{v} + \mathbf{u}^T (Q + \frac{B}{2})\mathbf{v} \\ &= \mathbf{u}^T (Q^T + Q)\mathbf{v} + \mathbf{u}^T B \mathbf{v} \end{aligned}$$

By definition  $Q^T + Q = 0$  and this reduces to:

$$\mathbf{u}^T B \mathbf{v} = u_N v_N - u_0 v_0$$

Thus, the discrete summation by parts formula becomes:

$$(D_1 \mathbf{u}, \mathbf{v})_H + (\mathbf{u}, D_1 \mathbf{v})_H = u_N v_N - u_0 v_0,$$

which exactly mimics the continuous integration by parts identity.

### 2.2.3 Upwind SBP-operator

The SBP operator introduced in Definition 2.1 is constructed in such a way that a central-difference stencil is required in the interior points. In contrast, an

*upwind* SBP operator employs a non-central, directionally biased stencil. This is particularly useful in numerical schemes that utilize flux-splitting to separate the flow into components moving in opposite directions. The upwind bias is applied to each component, allowing the method to more accurately capture the correct propagation behavior. Additionally, the directional bias also introduces numerical dissipation, which helps stabilize the method by damping oscillations that could otherwise lead to instability. This concept will be further explored later during the discretization. Upwind SBP operators based on non-central stencils are formally defined in [3] as:

**Definition 2.2:** The difference operators  $D_+ = H^{-1}(Q_+ + \frac{B}{2})$  and  $D_- = H^{-1}(Q_- + \frac{B}{2})$  approximating  $\frac{\partial}{\partial x}$ , using a  $p$ th-order accurate interior stencil, are said to be a  $p$ th order diagonal-norm upwind SBP operators if the diagonal matrix  $H$  defines a discrete norm,  $Q_+ + Q_-^T = 0$ , and  $\frac{Q_+ + Q_+^T}{2} = S$  is a negative semi-definite.

The stencil for  $Q_+$  is skewed from the central operator  $Q$  in the positive direction, while  $Q_-$  is skewed in the negative direction, in a symmetric manner such that:  $\frac{Q_+ + Q_-}{2} = Q$ . This leads to the following useful relations:

$$\begin{aligned}\frac{D_+ + D_-}{2} &= H^{-1}\left(\frac{Q_+ + Q_-}{2} + \frac{B}{2}\right) = H^{-1}\left(Q + \frac{B}{2}\right) = D_1 \\ \frac{D_+ - D_-}{2} &= H^{-1}\left(\frac{Q_+ - Q_-}{2}\right) = H^{-1}\left(\frac{Q_+ + Q_+^T}{2}\right) = H^{-1}S\end{aligned}$$

How the operators explicitly looks like is also described in [3]. For instance, the matrices  $Q_+$  and  $H$  defining  $D_+$  for the third-order accurate case is presented as (with  $m = 7$  nodes):

$$H = \Delta x \begin{bmatrix} \frac{5}{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{13}{12} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{13}{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{5}{12} \end{bmatrix}$$

$$Q_+ = \begin{bmatrix} -\frac{1}{12} & \frac{3}{4} & -\frac{1}{6} & 0 & 0 & 0 & 0 \\ -\frac{5}{12} & -\frac{5}{12} & 1 & -\frac{1}{6} & 0 & 0 & 0 \\ 0 & -\frac{1}{3} & -\frac{1}{2} & 1 & -\frac{1}{6} & 0 & 0 \\ 0 & 0 & -\frac{1}{3} & -\frac{1}{2} & 1 & -\frac{1}{6} & 0 \\ 0 & 0 & 0 & -\frac{1}{3} & -\frac{1}{2} & 1 & -\frac{1}{6} \\ 0 & 0 & 0 & 0 & -\frac{5}{12} & -\frac{5}{12} & \frac{3}{4} \\ 0 & 0 & 0 & 0 & 0 & -\frac{5}{12} & -\frac{1}{12} \end{bmatrix}$$

### 2.2.4 Simultaneous Approximation Term

If boundary conditions are enforced exactly within a numerical scheme, it becomes challenging to preserve both accuracy and stability. To address this, the *Simultaneous Approximation Term* (SAT) method is introduced. SAT incorporates a small penalty term into the numerical equations, enabling boundary conditions to be enforced weakly. In this approach, the solution is not forced to match the boundary values point-wise, but is instead drawn smoothly towards them. This is done in a way that preserves energetic stability, ensuring the discrete system mirrors the energy behavior of the continuous problem. By treating boundary conditions in this way, the SAT method achieves high-order accuracy while maintaining provable stability. [4]

### 2.3 Discretization

Finally, the discretization using our scheme is introduced. First, the spatial domain is discretized using  $m$  grid points between the boundaries  $x_l$  and  $x_r$ . Using this the  $i$ :th grid point and the spatial step size is given by:

$$x_i = x_l + (i - 1)\Delta x, \quad i = 1, \dots, m, \quad \Delta x = \frac{x_r - x_l}{m-1}$$

The discrete form of the solution vector  $q$  and the bottom topography  $\underline{b}$  are defined as  $2m \times 1$  and  $m \times 1$  vectors and are given by:

$$q = [q^{(1)} \quad q^{(2)}]^T = \begin{bmatrix} q_1^{(1)} & q_2^{(1)} & \dots & q_m^{(1)} & q_1^{(2)} & \dots & q_m^{(2)} \end{bmatrix}^T,$$

$$\underline{b} = [b_1 \quad b_2 \quad \dots \quad b_m]^T$$

where  $b_i = b(x_i)$ .

Note: In this report, discrete representations of vectors and matrices are written in non-bold font, in contrast to their continuous counterparts. This convention is consistently followed throughout the report to distinguish between continuous and discretized quantities.

Furthermore the flux Jacobian matrix  $A$  is given by the  $2m \times 2m$  matrix:

$$A = \begin{bmatrix} 0 & I_m \\ N & M \end{bmatrix},$$

where  $N = \text{diag}(gq_1^{(1)} - \frac{q_1^{(2)2}}{q_1^{(1)2}}, \dots, gq_m^{(1)} - \frac{q_m^{(2)2}}{q_m^{(1)2}})$  and  $M = \text{diag}(\frac{q_1^{(2)}}{q_1^{(1)}}, \dots, \frac{2q_m^{(2)}}{q_m^{(1)}})$

The Lax-Friedrichs flux-splitting version of  $A$  is then given by:

$A_{\pm} = A \pm \frac{R}{2}$ , where  $R = (I_m \otimes \alpha I_2) = \alpha I_{2m}$  and  $\alpha$  is the locally largest in magnitude eigenvalue of  $A$ .

For the discretizations the use of Kronecker products will be utilized in the coming analysis, which is defined as:

**Definition 2.3 (Kronecker product)** If  $\mathbf{A}$  is an  $m \times n$  matrix and  $\mathbf{B}$  is a  $p \times q$  matrix, the the Kronecker product  $\mathbf{A} \otimes \mathbf{B}$  is the  $pm \times qn$  block matrix:

$$\begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix}$$

Two useful rules for the Kronecker product are  $(A \otimes B)(C \otimes D) = (AC \otimes BD)$  and  $(A \otimes B)^T = A^T \otimes B^T$ .

### 2.3.1 Discretization of the Flux term

The discretization of the flux term is mainly based on the approach presented in [1], with some minor modifications introduced. To simplify the notation, the source term will be dropped in the following analysis and reintroduced later. Equations (5) and (6) are then expressed as:

$$\mathbf{q}_t + \mathbf{A}(\mathbf{q})\mathbf{q}_x = 0 \quad (7)$$

$$\mathbf{q}_t + \mathbf{A}(\mathbf{q})_+ \mathbf{q}_x + \mathbf{A}(\mathbf{q})_- \mathbf{q}_x = 0 \quad (8)$$

A semi-discrete approximation, using the standard SBP operator (defined in Definition 2.1) on the non-split form equation (7) is then given by:

$$q_t + A(I_2 \otimes D_1)q = 0 \quad (9)$$

In the upwind discretization of hyperbolic systems, the backward difference operator  $D_-$  is applied to the flux Jacobian associated with positive characteristic speeds. This ensures that the numerical flux at a given point is influenced by information propagating from the left—the upwind direction for right-moving waves. Similarly, for left-moving waves, the forward difference operator  $D_+$  is used in conjunction with the negative eigenvalue component  $A_-$ , ensuring proper upwinding from the right. This structure aligns the numerical scheme with the physical characteristic directions of the system, enhancing both stability and accuracy. A semi-discrete approximation of equation (8) using upwind SBP operators (defined in Definition 2.2) is given by:

$$\begin{aligned} & q_t + A_+(I_2 \otimes D_-)q + A_-(I_2 \otimes D_+)q &= 0 \\ \Rightarrow & q_t + \frac{A+R}{2}(I_2 \otimes D_-)q + \frac{A-R}{2}(I_2 \otimes D_+)q &= 0 \\ \Rightarrow & q_t + A(I_2 \otimes \frac{D_+ + D_-}{2})q - R(I_2 \otimes \frac{D_+ - D_-}{2})q &= 0 \\ \Rightarrow & q_t + A(I_2 \otimes D_1)q - R(I_2 \otimes H^{-1}S)q &= 0 \end{aligned} \quad (10)$$

A comparison between equation (9) and (10) leads to the conclusion that the upwind SBP operator combined with the flux splitting technique generates an extra term  $R(I_2 \otimes H^{-1}S)q$ . This term represents artificial dissipation (AD), which is a numerical mechanism that introduces additional damping into the system. While it does not correspond to any physical phenomenon in the original PDEs, it is intentionally included in the numerical scheme to enhance stability. Specifically, AD helps suppress spurious high-frequency oscillations that can arise due to discretization errors, especially near discontinuities or steep gradients in the solution (often from shocks). Thus, artificial dissipation is not only a byproduct of flux splitting with upwind SBP operators but also a beneficial feature that improves the robustness of high-order methods for solving hyperbolic systems. However, a downside is that it can overly smooth steep gradients or discontinuities in the solution, reducing the accuracy where those details matter. [1]

Since  $A$  is the Jacobian of  $F$ , the term  $A(I_2 \otimes D_1)q$  can be discretized as  $(I_2 \otimes D_1)F(q)$ , allowing the flux to be computed directly. A slight modification to the artificial dissipation term is also made in order to preserve the well-balanced property. This modification follows the approach presented in [5]. With these changes equation (10) becomes:

$$q_t + (I_2 \otimes D_1)F(q) - (R \otimes H^{-1}S)(q + \bar{b}) = 0. \quad (11)$$

where  $\bar{b}$  is the  $2m \times 1$  matrix  $\bar{b} = \begin{bmatrix} \underline{b} \\ 0 \end{bmatrix}$

### 2.3.2 Discretization of the Source Term

The source term  $\mathbf{C}q$  is discretized following the approach shown in [5] as this has been shown to lead to a well-balanced scheme. First, the source term is rewritten in the following way:

$$\mathbf{C}q = \begin{bmatrix} \mathbf{0} \\ gq_1 b_x \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ gq_1 b_x + gbb_x - gbb_x \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ g(q_1 + b)b_x - (\frac{1}{2}gb^2)_x \end{bmatrix}$$

Using SBP operators the discretization of the source term is then given by:

$$G(q) = \begin{bmatrix} \mathbf{0} \\ g(\underline{q}^{(1)} + \underline{b})D_1\underline{b} - D_1(\frac{1}{2}g\underline{b}\underline{b}) \end{bmatrix}, \quad (12)$$

where  $\underline{q}^{(1)} = \text{diag}(q^{(1)})$  and  $\underline{b} = \text{diag}(b)$ .

### 2.3.3 Boundary treatment using SAT

The boundary condition (BC) for equation (5) are defined as:

$$L_l \mathbf{q} = g_l(t) \quad x = x_l, t > 0$$

$$L_r \mathbf{q} = g_r(t) \quad x = x_r, t > 0,$$

where  $L_{l,r}$  is a  $1 \times 2$  boundary operator and  $g_{l,r}(t)$  are scalars containing the boundary data. These BC can be imposed weakly using a SAT penalty approach. Following the notation used in [1], the SAT term for subcritical flow with one boundary condition at each boundary specified, is constructed as follows:

$$\begin{aligned} SAT = & -\bar{H}^{-1} e_L W_+ \Lambda_+ (L_l W_+)^{-1} (L_l e_L^T q - g_l(t)) \\ & + \bar{H}^{-1} e_R W_- \Lambda_- (L_r W_-)^{-1} (L_r e_R^T q - g_r(t)), \end{aligned} \quad (13)$$

where  $\Lambda_{\pm}$  and  $W_{\pm}$  are the eigenvalues and eigenvectors defined in 2.1 System of equations and  $\bar{H} = I_2 \otimes H$ ,  $e_L = I_2 \otimes e_1$  and  $e_R = I_2 \otimes e_m$ , with  $e_1, e_m \in \mathbb{R}^m$  denoting the standard unit vectors.

Combining equations (11),(12) and (13) the full SBP-SAT approximation of equation (5), using flux-splitting combined with upwind SBP-operators, which is the same scheme proposed by Lundgren and Mattson in [1], is given by:

$$q_t + (I_2 \otimes D_1)F(q) - R(I_2 \otimes H^{-1}S)(q + \bar{b}) + G(q) = SAT \quad (14)$$

## 2.4 Time discretization and CFL condition

To obtain the fully discrete form of Equation (5), Equation (14) is discretized in time using the classical fifth-order Runge–Kutta method, with initial conditions selected based on the specific test case being considered. The time step  $\Delta t$  is chosen to satisfy the Courant–Friedrichs–Lewy condition  $\Delta x = C\Delta t$ , where  $C < |\Lambda_{max}|$ , and  $|\Lambda_{max}|$  is the largest in magnitude eigenvalue computed in section 2.1. This ensures that numerical information does not propagate faster than physical information, thereby maintaining stability of the scheme. [8]

## 2.5 Programming

The simulations were implemented in Python, utilizing the NumPy library to efficiently handle the vector and matrix operations required by the numerical scheme. For a detailed overview of the code structure and implementation, see the appendix.

The time integration of equation (14) was performed using the `scipy.integrate.RK45` function, which implements an explicit Runge-Kutta method of order 5(4).[2] Additionally, upwind SBP operators of second, third, and fifth order accuracy were implemented to explore the effect of spatial discretization on the solution. A time step of  $\Delta t = 0.01 \Delta x \frac{s}{m}$  was used for all the simulations to ensure that the CFL condition is satisfied throughout.

### 3 Results

In this section, we present the results of our simulations using the developed numerical scheme. To verify the accuracy of the method, three benchmark cases, originally introduced in [1], were first implemented and analyzed. These three cases include 3.1 Lake at rest, 3.2 Dam break and 3.3 Gaussian pulse. To further assess the performance of the scheme, two additional test case simulating 3.4 Gaussian pulse over slope and 3.5 Subcritical flow over bump is implemented. For the cases where the  $L_2$  error per grid point was computed, the following formula was used:

$$L_2 = \frac{\sqrt{\sum_{i=1}^{2m} (q_i - q_i^{exact})^2}}{2m}$$

#### 3.1 Lake at rest

The Lake at rest problem is a well known benchmark case for the 1D SWE. The simulation domain is set to be  $L = 25m$ , with initial conditions defined as  $h(x) + b(x) = 0.5m$  and  $hu = 0m^2/s^2$  (see Figure 2). Here,  $b(x)$  is defined as:

$$b(x) = \begin{cases} 0.2 - 0.05(x - 10)^2, & \text{if } 8 < x < 12. \\ 0, & \text{else.} \end{cases} \quad (15)$$

The boundary conditions were chosen as:

$$L_l = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad L_r = \begin{bmatrix} 1 & 0 \end{bmatrix} \\ g_l(t) = 0.5, \quad g_r(t) = 0.5,$$

which imposes a constant water depth of 0.5m at both boundaries throughout the simulation. The simulation was then run using different numbers of spatial grid points,  $m$  and varying orders of accuracy for the numerical scheme. Since this is a steady-state test, the numerical scheme is expected to preserve the initial conditions exactly. Any deviation from the initial state indicates numerical error, which is summarized in Table 1. The error was very small for all of the cases, with only a slight increase in error as the number of spatial grid points increases.

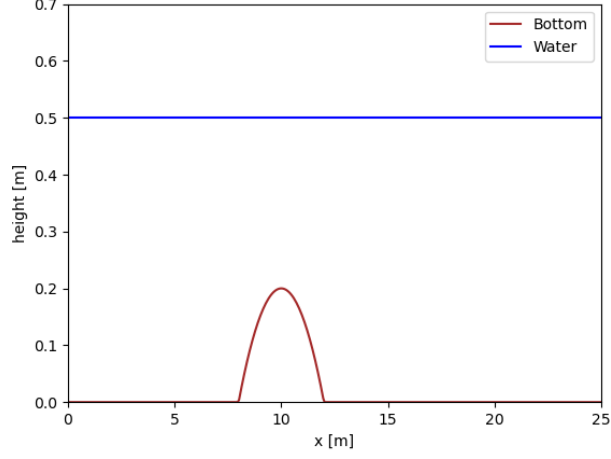


Figure 2: Reference solution for the "Lake at rest" problem

$m$	$\log_{10}(L_{(2nd)}^2/2m)$	$\log_{10}(L_{(3rd)}^2/2m)$	$\log_{10}(L_{(5th)}^2/2m)$
50	-16.571	-16.640	-16.491
100	-16.579	-16.627	-16.356
200	-16.448	-16.652	-16.326
400	-16.460	-16.609	-16.376

Table 1:  $L_2$  errors per point for the "Lake at rest" problem at  $t = 10$  for different spatial grid points ( $m$ ) and orders of accuracy.

### 3.2 Dam break

The Dam break problem is a benchmark case meant to test if the numerical scheme properly captures shock behavior. The simulation domain is set to be  $L = 1\text{m}$  and the initial conditions are defined as  $hu = 0$ ,  $b(x) = 0$  and:

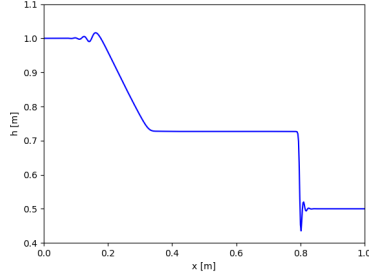
$$h = \begin{cases} 1, & \text{if } 0 \leq x \leq 0.5. \\ 0.5, & \text{if } 0.5 < x \leq 1. \end{cases}$$

Boundary conditions are defined as:

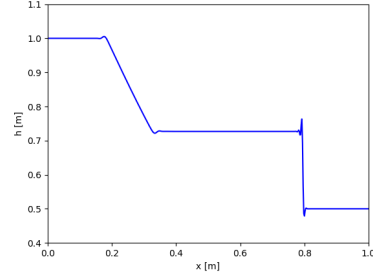
$$L_l = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad L_r = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$g_l = 1, \quad g_r = 0.5,$$

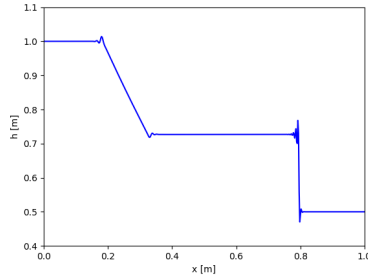
which imposes a constant water depth of  $0.5\text{m}$  for the right BC and  $1\text{m}$  for the left BC throughout the simulation. Using 501 grid points, the solutions at  $t = 0.1\text{s}$  is shown in Figure 3. As can be observed, spurious oscillations appear near the sharp discontinuities. No error analysis was performed, as the goal is to illustrate the scheme's shock-capturing behavior and its limitations.



(a) second order



(b) third order



(c) fifth order

Figure 3: Dam break,  $m = 501$ ,  $t = 0.1\text{s}$

### 3.3 Gaussian pulse

In this test case, a Gaussian pulse originating at the center of the domain is simulated, with an initial velocity of zero. The domain is defined as  $L = 1\text{m}$  and the initial conditions are given by  $hu = 0$ ,  $b(x) = 0$  and:

$$h = 1 + 0.1 \exp \left( - \left( \frac{x - 0.5}{0.1} \right)^2 \right)$$

Wall boundary conditions are applied as the following:

$$L_l = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad L_r = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

$$g_l = 0, \quad g_r = 0.$$

These conditions enforce a zero-velocity constraint at both boundaries throughout the simulation. The evolution of the Gaussian pulse over time is illustrated in Figure 4.

In addition to the  $L_2$  errors, the convergence rate was also calculated for the different orders of accuracy for this test case. This is done using the same formula described in Lundgren and Mattsson's study [1], with the results are summarized in Table 2:

$$c = \log_{10} \left( \frac{\|\mathbf{q} - \mathbf{q}^{(m_1)}\|_H}{\|\mathbf{q} - \mathbf{q}^{(m_2)}\|_H} \right) / \log_{10} \left( \frac{m_2}{m_1} \right)$$

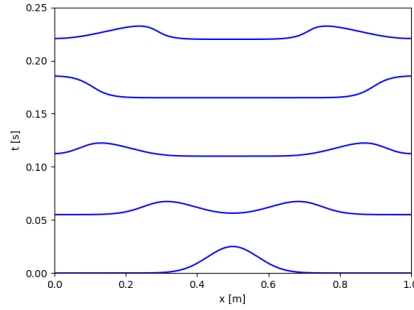


Figure 4: Gaussian pulse over time, 5th order accuracy,  $m = 801$

The simulation of the Gaussian pulse appears to follow realistic wave behavior, closely resembling the expected physical dynamics of wave propagation. It is also practically identical to the simulation made in Lundgrens and Mattsons study[1].

$m$	$\log_{10}(L_{(2nd)}^2/2m)$	c	$\log_{10}(L_{(3rd)}^2/2m)$	c	$\log_{10}(L_{(5th)}^2/2m)$	c
51	-2.635	-	-3.086	-	-3.012	-
101	-3.130	1.168	-3.513	0.937	-3.466	1.030
201	-3.704	1.422	-3.980	1.063	-3.951	1.122
401	-4.317	1.544	-4.498	1.228	-4.472	1.238
801	-4.996	1.759	-5.138	1.629	-5.101	1.594

Table 2:  $L_2$  errors per point and convergence rate (c) for the Gaussian pulse problem at  $t = 0.22$  for different spatial grid points ( $m$ ) and orders of accuracy. Fifth order with  $m = 1601$  was used as reference.

As shown in Table 2, the results of the convergence rates of the Gaussian pulse appear not to follow expected convergence behavior. The expected convergence rates were 1, 2 and 3 for the 2nd, 3rd and 5th order schemes. This expectation comes from that the even orders should have an convergence rate of  $(p + 1)/2$  and the odd orders a convergence rate of  $p/2$ , were  $p$  is the order of accuracy.[3]

### 3.4 Gaussian pulse over slope

To study the effects of water height on waves in the scheme, a Gaussian pulse was simulated with a slanted bottom topography. The domain length was set to  $L = 50\text{m}$  and the bottom topography was defiened as:

$$b(x) = 0.45 - 0.009x$$

The initial conditions were chosen to be similar to those in the 3.3 Gaussian Pulse case with  $hu = 0\text{m}^2/\text{s}$  and the initial water height defiened as:

$$h(x) = 0.5 + 0.05\exp\left(-(x - 25)^2\right)$$

Boundary conditions were defined as:

$$L_l = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad L_r = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$g_l = 0.05, \quad g_r = 0.5,$$

which imposes a constant water height of 0.05 m at the left boundary and 0.5 m at the right boundary. This setup ensures that the total water surface level  $h + b$  is the same on both boundaries. Figure 5 shows how the wave propagates over time using the fifth order accuracy scheme with 400 grid points.

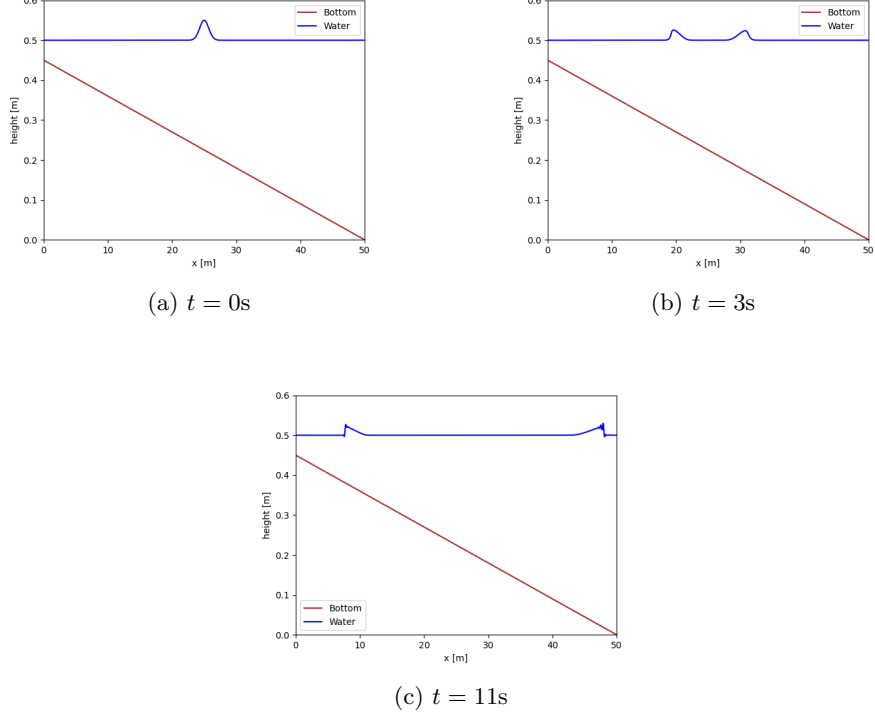


Figure 5: Gaussian pulse over a slope, 5th order of accuracy,  $m = 800$ .

As expected, the left-moving wave propagates more slowly compared to the right-moving wave. This is due to the shallower water depth on the left side of the domain, caused by the sloping bottom topology. According to the shallow water wave speed relation  $\sqrt{gh}$ , waves travel slower in shallower regions and faster in deeper water.

### 3.5 Subcritical flow over bump

A benchmark test case from SWASHES [9] was used to evaluate how the scheme handles constant subcritical flow over a bump, a case that is analytically known to admit a steady-state solution. This type of flow requires both the water height and the discharge ( $h$  and  $hu$ ) to be specified at the outflow boundary to fully define the physical state. However, the implemented SAT formulation in this study only allows for one boundary condition to be imposed per boundary, making it impossible to specify both water depth and discharge at the outflow. As a workaround, the simulated domain was extended to  $L = 250\text{m}$ , while the region of interest was limited to  $0\text{m} \leq x \leq 25\text{m}$ . This pushed the boundary effects far upstream, effectively simulating an open outflow without needing to explicitly define both conditions. The bottom topography was defined the same as in

3.1 Lake at rest. The initial conditions were set to:  $h = 1.147\text{m}$  and  $hu = 0\text{m}^2/\text{s}$ . The initial water height  $h$  was determined through trial and error by repeatedly simulating the test case until the steady state water height would approximately be  $h \approx 2\text{m}$ . The boundary conditions were chosen to be:

$$L_l = \begin{bmatrix} 0 & 1 \end{bmatrix}, L_r = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$g_l = 4.42\text{m}^2/\text{s}, g_r = 1.147\text{m},$$

which imposes steady inflow of water at the left boundary and a constant water height at right boundary. Note here that the right boundary does not significantly influence the solution, since the region of interest lies far downstream. The initial height and right boundary were different from the test case in SWASHES.[9] There, the initial condition had a height of 2m and the outflow boundary was set to be a constant height of 2m. The reason for the difference is that the SAT term changes  $hu$  and  $h$  simultaneously and proportionally, so when the inflow suddenly starts at  $t = 0\text{s}$ , the water height will also rise (see 2.3.3 Boundary treatment using SAT). Thus, in order to get the same steady state water height, a lower initial height was needed. Figure 6 shows how the wave propagates through the domain and gradually settles into a steady-state solution.

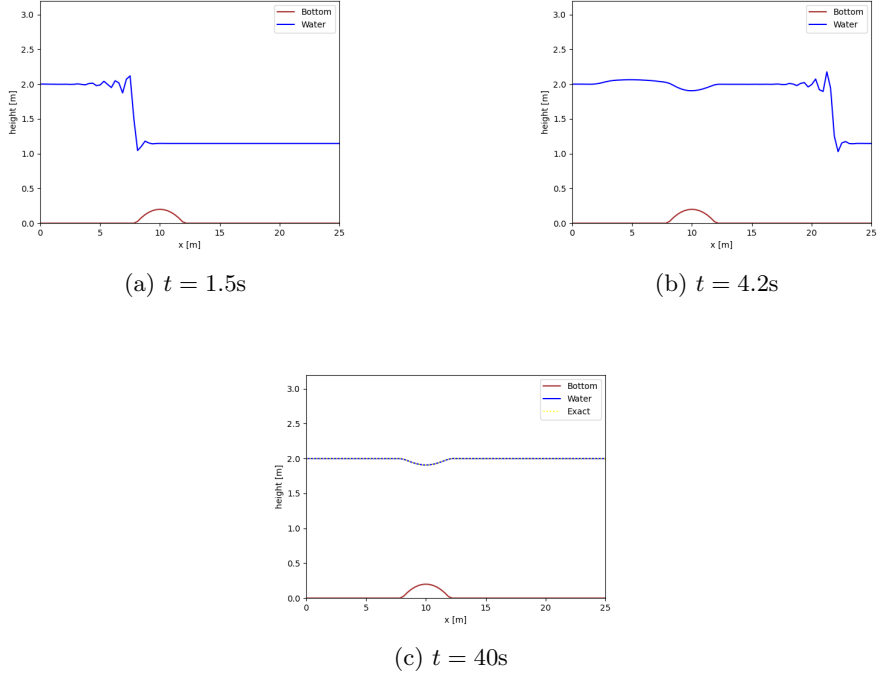


Figure 6: Subcritical flow over a bump, 5th order of accuracy,  $m = 800$

The  $L_2$  errors in Table 3 were calculated using the analytical solution from SWASHES in [9]. The analytical solution found there is:

$$h(x)^3 + \left( b(x) - \frac{(hu)_0^2}{2gh_L^2} - h_L \right) + \frac{(hu)_0^2}{2g} = 0$$

Note that the number of spatial grid points,  $m$ , in Table 3 was the amount of points inside the observed area. Thus, about 10 times as many points were needed in order to run these simulations.

$m$	$\log_{10}(L_{(2nd)}^2/m)$	$\log_{10}(L_{(3rd)}^2/m)$	$\log_{10}(L_{(5th)}^2/m)$
6	-1.593	-1.561	-1.541
11	-1.873	-1.842	-1.814
21	-1.942	-1.921	-1.907
41	-2.057	-2.050	-2.047
81	-2.192	-2.191	-2.190

Table 3:  $L_2$  errors per point for the Subcritical flow problem at  $t = 40$ s for different spatial grid points ( $m$ ) and orders of accuracy. Analytical solution was used as reference.

The errors per point in Table 3 were relatively small and did not deviate much with respect to order of accuracy. However, there is a clear decrease in the error when the amount of spatial grid points increases.

## 4 Discussion

The simulations were designed primarily to assess the numerical properties and stability of the SBP-SAT scheme when applied to the shallow water equations, rather than to precisely model real-world fluid dynamics. Within this context, all test cases were implemented successfully, with results either matching with their known benchmarks or produced behavior that were consistent with the expected outcomes of the mathematical model. However, the scheme also had its limitations, especially in scenarios involving discontinuities or non-smooth behavior.

This is particularly evident in the 3.2 Dam break simulation, where the simulation showed unnatural, rapid oscillations near sharp discontinuities. These arise as a result of the SBP-SAT discretization, which despite being stable and high-order accurate in smooth regions tends to introduce non-physical oscillations near shocks.

Additionally, the SWE cannot capture wave breaking. This limitation becomes evident in the 3.5 Subcritical flow over bump case. In the final frame of the simulation, the wave front becomes very steep, and noticeable numerical noise appears near the crest. This behavior likely results from the wave approaching a breaking state, which the SWE are not capable of representing physically.

Despite this, the SBP-SAT scheme performed well across the other test cases, demonstrating robust and realistic behavior under more smooth conditions.

### 4.1 Effects of Different Variables

This section investigates how changing various parameters in the numerical setup affects the performance and results of the SBP-SAT scheme. Specifically, the effects of altering the initial conditions, boundary conditions, spatial resolution, and time step size are explored.

#### 4.1.1 Effect of Initial Conditions

The test cases with smooth initial conditions for the water height (3.1 Lake at rest, 3.2 Dam break, 3.4 Gaussian pulse over slope and 3.5 Subcritical flow over bump) produced the most stable results with smooth and physically realistic simulations. In contrast, the 3.2 Dam break case, where the initial condition were discontinuous, the results appeared to be less natural with noticeable spurious oscillations appearing near the discontinuities. One possible explanation of this is that the SBP-SAT scheme can interpret local errors as small waves and as they propagate through the domain without naturally dissipating, it can lead to the accumulation of spurious oscillations.

This behavior highlights the scheme’s sensitivity to non-smooth initial states and suggests that the implemented SBP-SAT scheme performs best when applied to smooth and continuous initial conditions.

#### 4.1.2 Effect of Boundary Conditions

Boundary conditions in this study were enforced weakly using the SAT technique, which generally worked well, as all test cases produced stable simulations.

In the 3.1 Lake at Rest, 3.2 Dam break, and 3.4 Gaussian Pulse Over Slope test cases, the boundary conditions were held constant throughout the simulations. In these cases, the simulated waves did not reach or significantly interact with the boundaries within the simulated time frames. As a result, the specific boundary condition choice had minimal influence on the results.

In the 3.3 Gaussian Pulse test case, the wave did interact with the boundaries, where a wall boundary condition was applied. This interaction appeared physically reasonable, and the wave remained smooth and stable throughout the simulation. However, to fully evaluate the long-term stability of these boundary interactions, a longer simulation with multiple boundary reflections would be needed to assess whether the solution eventually diverges or remains stable.

In the 3.5 Subcritical flow over bump test case, a steady inflow was imposed at the left boundary, which worked well with the SAT formulation. However, the SAT approach only allows one condition per boundary, which made it impossible to enforce a steady outflow at the right boundary. To address this limitation, the domain was extended to simulate open outflow conditions, while only a portion of the domain was analyzed as the test region.

#### 4.1.3 Sensitivity to spatial resolution, $\Delta x$

As expected, increasing the number of grid points improved the spatial accuracy of the solution by better resolving smooth features. However, in the case of 3.2 Dam break, it did not eliminate the spurious oscillations near discontinuities. This suggests that while higher resolution benefits smooth problems, it may not fully resolve issues around shocks without additional shock-capturing techniques. Additionally, a problem that appeared while using a finer spatial resolution was that the scheme often became unstable. These issues were resolved by reducing the time step, which is discussed in more detail in following section.

#### 4.1.4 Sensitivity to Time step, $\Delta t$

The implemented scheme was found to be sensitive to the choice of time step size. Specifically, using a smaller step size of  $\Delta t = 0.01\Delta x \frac{s}{m}$  provided more stable results, especially for high spatial resolutions and higher-order schemes.

A larger time step of  $\Delta t = 0.1\Delta x \frac{s}{m}$ , which was used by Lundgren and Mattsson in their work [1], often led to instabilities in some of the test cases when  $\Delta x$  was small. This is likely due to the CFL-condition being breached. While the smaller time step increased computational cost, it ensured stability across all of test cases considered in this study.

## 4.2 Error Analysis

The 3.1 Lake at rest test case resulted in a very small error per point ( $\sim 10^{-16.5}\text{m}$ ) (see Table 1). This was expected since the simulation should only remain constant during the simulation. The errors were also very similar for the different order schemes and spatial resolution, which indicates that the source for this is likely due to rounding errors in Python, rather than the numerical scheme itself.

The 3.3 Gaussian pulse test case resulted in much larger errors per point ( $10^{-5}\text{m} \sim 10^{-2.5}\text{m}$ ) (see Table 2). These errors were, of course, not noticeable by just observing the simulation. However, they are not entirely negligible, especially if the simulation were to be run over longer time spans, where error accumulation could eventually affect the solution. This behavior is not unexpected, since the reference solution itself was not exact, and the simulated flow involved significantly more dynamic behavior compared to the 3.1 Lake at Rest test case. As expected, the errors also decreased as the number of spatial grid points increased. The order of the scheme, however, did not appear to have a significant impact on the observed error, which is contrary to expectations based on the findings of Lundgren and Mattsson’s study [1]. This is further discussed in the next section. (see 4.1.4 Convergence rate)

The 3.5 Subcritical Flow Over Bump test case produced the largest error per point among all the evaluated error test cases, with values ranging from approximately  $10^{-1.6}\text{m} \sim 10^{-2.2}\text{m}$  (see Table 3). One possible explanation for the relatively larger error is that the error analysis was limited to coarser space resolutions, with grid sized ranging from  $m = 6$  to  $m = 81$ . In comparison a finer space resolution of  $m = 400$  was used for the 3.1 Lake at rest test case and  $m = 801$  for the 3.3 Gaussian pulse test case. This reduced spatial resolution likely limited the accuracy of the simulation. Another possible explanation is that this test case required the simulation to reach a steady state, unlike the other cases which primarily focused on short-term wave propagation. Extending the simulation time (from 40s) could therefore allow the solution to converge more accurately, potentially reducing the error. The order of accuracy of the scheme did not have a significant impact of the error, which is consistent with the other test cases where error analysis was performed. This is unexpected since, based on theoretical expectations, a higher-order scheme should produce a smaller error due to its improved approximation properties.

### 4.3 Convergence Behavior

The convergence rates of 3.3 Gaussian pulse did not match expected convergence behavior, with the different order schemes having similar convergence rates. (see Table 2) In fact, the results showed a decreasing trend in convergence rate with increasing scheme order, which is opposite to the expected behavior. The expected convergence rates were that even order schemes should have a convergence rate of  $(p + 1)/2$ , while odd order schemes should have a convergence rate of  $p/2$ , where  $p$  is the order of accuracy.[3] One possible explanation is the use of a fifth-order Runge-Kutta time integration method (RK5(4)) combined with a fifth-order spatial scheme as the reference solution. In contrast, Lundgren and Mattsson’s study [1], which conducted a similar convergence analysis, used a sixth-order Runge-Kutta method along with a ninth-order spatial scheme as their reference. Further investigation is needed to determine what source is responsible for the deviation from the expected convergence rates.

### 4.4 Future improvements

To further evaluate the effectiveness of the SBP-SAT numerical scheme for solving the shallow water equations, additional test cases could be introduced to assess whether the simulated physics remain consistent under a wider range of conditions. Moreover, to more accurately capture real-world phenomena, the numerical scheme could also be extended to two spatial dimensions, allowing flow to occur in two perpendicular directions within the horizontal plane.

To further improve the solution quality, various shock-damping techniques could be incorporated to reduce spurious oscillations and produce smoother transitions near discontinuities. Additionally, the SAT formulation could be extended to handle multiple boundary conditions simultaneously at a single boundary, which would allow for more realistic inflow and outflow control.

The current implementation is also limited to subcritical flow, which reduces the variety of flow behaviors that can be captured. Future work could explore the inclusion of supercritical flows, though this may introduce challenges such as unstable oscillations that are more difficult to handle numerically.

Finally, higher-order schemes for both spatial and temporal discretizations could be introduced to improve the overall accuracy and stability of the solution.

## 5 Conclusion

In this thesis, the shallow water equations were solved using a high-order SBP-SAT scheme. The numerical model was applied to several test cases to assess its ability to capture various flow behaviors. The results demonstrated that the SBP-SAT implementation accurately simulates basic water behaviors, such as steady-state conditions and smooth wave propagation. However, the method struggled with more challenging scenarios involving shocks and complex water flow, where the accuracy was notably reduced. Future work could focus on addressing these issues by implementing shock-capturing techniques and enhanced boundary treatments. Additionally, applying the model to a wider range of test cases would help to further evaluate and validate the method's robustness and applicability.

## 6 References

- [1] Lukas Lundgren, Ken Mattsson. An efficient finite difference method for the shallow water equations. *Journal of Computational Physics*, Volume 422, Article 109784, 2020. DOI: 10.1016/j.jcp.2020.109784. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0021999120305581?via%3Dihub>.
- [2] SciPy Community. `scipy.integrate.RK45` — Explicit Runge-Kutta method of order 5(4). *SciPy Documentation*, version 1.15.2. Accessed 2025-04-23. URL: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.RK45.html>.
- [3] Ken Mattsson. Diagonal-norm Upwind SBP operators *Journal of Computational Physics*, Volume 335, Article 283310, 2017. DOI: 10.1016/j.jcp.2017.01.042. URL: <https://www.sciencedirect.com/science/article/pii/S002199911730058X?via%3Dihub>.
- [4] David C. Del Rey Fernández, Jason E. Hicken, David W. Zingg. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations *Computers & fluids*, Volume 95, Article 171196, 2014. DOI: 10.1016/j.compfluid.2014.02.016. URL: <https://www.sciencedirect.com/science/article/pii/S0045793014000796>.
- [5] Yulong Xing, Chi-Wang Shu. High order finite difference WENO schemes with the exact conservation property for the shallow water equations *Journal of Computational Physics*, Volume 208, Article 206227, 2005. DOI: 10.1016/j.jcp.2005.02.006. URL: <https://www.sciencedirect.com/science/article/pii/S002199910500094X?via%3Dihub>.
- [6] Yulong Xing. Chapter 13 - Numerical Methods for the Nonlinear Shallow Water Equations *Handbook of Numerical Analysis*, Volume 18, Pages 361-384, 2017. DOI: 10.1016/bs.hna.2016.09.003. URL: <https://www.sciencedirect.com/science/article/pii/S1570865916300126?ref=pdf.download&fr=RR-2&rr=939799b02c13f8ae%3Dihub>.
- [7] Achim Feldmeier. Shallow Water Waves *Theoretical Fluid Dynamics*, Pages 395-366, 2020. DOI: 10.1007/978-3-030-31022-6\_8. URL: [https://link.springer.com/chapter/10.1007/978-3-030-31022-6\\_8?via%3Dihub](https://link.springer.com/chapter/10.1007/978-3-030-31022-6_8?via%3Dihub).
- [8] F.X. Trias, X. Álvarez-Farré, A. Alsalti-Baldellou, A. Gorobets, A. Oliva. An efficient eigenvalue bounding method: CFL condition revisited. *Computer Physics Communications*, Volume 305, Article 109351, 2024. DOI: 10.1016/j.cpc.2024.109351. URL: <https://www.sciencedirect.com/science/article/pii/S0010485124001093?via%3Dihub>.

<https://www.sciencedirect.com/science/article/pii/S0010465524002741%3Dihub>.

- [9] Olivier Delestre, Carine Lucas, Pierre-Antoine Ksinant, Frédéric Darboux, Christian Laguerre, et al.. SWASHES: a compilation of Shallow Water Analytic Solutions for Hydraulic and Environmental Studies. *International Journal for Numerical Methods in Fluids*, 2013, 72 (3), pp.269-300. DOI: 10.1002/flid.3741. URL: <https://hal.science/hal-00628246v6/document>.

# Appendix

## Equations and Notation for Vectors and Matrices:

Most of the important equations, matrices and vectors for implementing the scheme are collected here.

In this section,  $m$  is the number of spatial points used to describe the domain.

**The full equation for the scheme:**

$$q_t + (I_2 \otimes D_1)F(q) - R(t) \otimes H^{-1}S(q + \bar{b}) + G(q) = SAT$$

**q<sub>t</sub> :**

$$q_t = \frac{\partial q}{\partial t}$$

**q :**

Here,  $q_i^{(1)} = h$  and  $q_i^{(2)} = hu$  for a given spatial point  $i$ .

$$q = \begin{bmatrix} q_1^{(1)} \\ q_2^{(1)} \\ \vdots \\ q_m^{(1)} \\ q_1^{(2)} \\ \vdots \\ q_m^{(2)} \end{bmatrix}$$

**I<sub>2</sub> :**

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$\otimes :$

The Kronecker product is defined as:

$$C \otimes D = \begin{bmatrix} c_{1,1}D & \cdots & c_{1,q}D \\ \vdots & & \vdots \\ c_{p,1}D & \cdots & c_{p,q}D \end{bmatrix}$$

$\mathbf{D}_1 :$

$D_1$  is the SBP operator approximating the first order derivative  $\frac{\partial}{\partial x}$ .

$$D_1 = H^{-1}(Q + \frac{B}{2})$$

For further reading about  $Q$ , see Definition 2.1.

$\mathbf{H} :$

$H$  is the matrix that approximates the inner product:

$$(u, v)_H = u^T H v, \quad \|u\|_H^2 = u^T H u$$

$H$  is a diagonal matrix with the inner points being  $\frac{1}{m}$ , and the edges depend on the order of accuracy. For further reading, see Definition 2.1.

$\mathbf{B} :$

$B$  is an  $m \times m$  matrix. Each element is zero except the top left and bottom right, being -1 and 1 respectively.

$$B = \text{diag}(-1, 0, \dots, 0, 1)$$

$\mathbf{F}(\mathbf{q}) :$

$F(q)$  is a  $2m \times 1$  vector.  $q_2$  and  $\frac{q_2^2}{q_1} + \frac{1}{2}gq_1^2$  each represent a  $m \times 1$  vector, where  $q_1 = h$  and  $q_2 = hu$  for each spatial point.

$$F(q) = \begin{bmatrix} q_2 \\ \frac{q_2^2}{q_1} + \frac{1}{2}gq_1^2 \end{bmatrix}$$

**R(t)** :

Here,  $\alpha(t)$  is the globally largest of  $|u \pm \sqrt{gh}|$  (for a given time,  $t$ ) .

$$R(t) = \alpha(t)I_2$$

**S** :

$$S = \frac{Q_+ + Q_+^T}{2} = \frac{Q_+ - Q_-}{2}$$

For further reading about  $Q_+$  and  $Q_-$ , see Definition 2.2.

$\overline{\mathbf{b}}$  :

$$\overline{b} = \left[ \begin{array}{c} \frac{b}{\mathbf{0}} \end{array} \right]$$

$\underline{\mathbf{b}}$  :

$$\underline{b} = \left[ \begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_m \end{array} \right], \text{ where } b_i = b(x_i) \text{ is the height of the bottom}$$

$\mathbf{0}$  :

$$\mathbf{0} = \left[ \begin{array}{c} 0 \\ \vdots \\ 0 \\ 0 \end{array} \right], \text{ with a length of } m$$

**G(q) :**

$$G(q) = \begin{bmatrix} \mathbf{0} \\ g(\underline{\underline{q}}^{(1)} + \underline{\underline{b}})D_1\underline{\underline{b}} - D_1(\frac{1}{2}g\underline{\underline{b}}\underline{\underline{b}}) \end{bmatrix}$$

**$\underline{\underline{q}}^{(1)}$  :**

$$\underline{\underline{q}}^{(1)} = diag(q^{(1)})$$

**$\underline{\underline{b}}$  :**

$$\underline{\underline{b}} = diag(\underline{\underline{b}})$$

**SAT :**

$$\begin{aligned} SAT = & -\bar{H}^{-1}e_L W_+ \Lambda_+ (L_l W_+)^{-1} (L_l e_L^T q - g_l(t)) \\ & + \bar{H}^{-1}e_R W_- \Lambda_- (L_r W_-)^{-1} (L_r e_R^T q - g_r(t)) \end{aligned}$$

**$\bar{\mathbf{H}}$ :**

$$\bar{H} = I_2 \otimes H$$

**$\mathbf{e}_L, \mathbf{e}_R$  :**

$e_L = I_2 \otimes e_1$ , where:

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \text{ with a length of } m$$

and  $e_R = I_2 \otimes e_m$ , where:

$$e_m = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \text{ with a length of } m$$

$\mathbf{W}_+, \mathbf{W}_-$  and  $\Lambda_+, \Lambda_-$

$$\Lambda_{\pm} = \frac{q_2}{q_1} \pm \sqrt{gq_1} = u \pm \sqrt{gh}$$

$$\mathbf{W}_{\pm} = \begin{bmatrix} 1 \\ \Lambda_{\pm} \end{bmatrix}$$

$L_l, L_r$  and  $g_l(t), g_r(t)$

$L_l, L_r, g_l(t)$  and  $g_r(t)$  define the boundary conditions of the simulation.  $L_l$  and  $g_l(t)$  define the left boundary, and  $L_r$  and  $g_r(t)$  define the right boundary.

$$\begin{array}{ll} L_l \mathbf{q} = g_l(t) & x = x_l, t > 0 \\ L_r \mathbf{q} = g_r(t) & x = x_r, t > 0 \end{array}$$

## Code appendix:

The code in the appendix has been simplified and altered to make the important parts easier to understand.

Any functions or constants that are not explained in the code have self-explanatory names.

### Time integration

```

1 from scipy.integrate import RK45
2
3 #Initialize the Runge-Kutta integrator
4 rk = RK45(fun=q_t, t0=0, y0=q0, t_bound=t_max, max_step=t_step,
5         first_step=t_step)
6
7 #Iterate over all time steps
8 while rk.t < rk.t_bound:
9     #One step
10    rk.step()
```

Listing 1: Integration

$q_t$ , equation (14)

```

1 import numpy as np
2 from constants import Constants as c #Self made class with
   constants for each simulation
3
4 def q_t(self, t=None, y=None) -> np.ndarray:
5     """Returns the time derivative of q for the current time step.
6         """
7
8     SAT = self.SAT() #self.SAT returns the SAT for the current time
9     #Calculating the equation except SAT and q_t
10
11     equation1 = np.dot(c.I2_D1, self.F()) #self.F() returns F(q)
12         for the current time
13
14     equation2 = np.kron(self.R(), c.H_inv_dot_S) #self.R() returns
15         R(t) for the current time
16     equation2 = np.dot(equation2, (q + c.b_overline))
17
18     equation = equation1 - equation2 + self.G() #self.G() returns G
19         (q) for the current time
20
21     return SAT - equation

```

Listing 2:  $q_t$

## SAT

```

1 import numpy as np
2 from constants import Constants as c #Self made class with
   constants for each simulation
3
4 def SAT(self) -> np.ndarray:
5     """Returns the SAT. Output of size 2m x 1. """
6
7     #A_eigen returns the eigenvalues of A for a given point
8     Lambda_plus, _ = A_eigen(i=0)
9     _, Lambda_minus = A_eigen(i=c.m-1)
10
11     W_plus = np.array([[1], [Lambda_plus]])
12     W_minus = np.array([[1], [Lambda_minus]])
13
14     #Left part
15     left_part = np.dot(c.e_LT, q)
16     left_part = np.dot(self.L_l, left_part)
17     left_part -= g_L() #Everything inside left parenthesis
18     inside_par = np.dot(L_l, W_plus)
19     left_part *= 1 / inside_par.item()
20     left_part *= Lambda_plus
21     left_part = W_plus * left_part
22     left_part = np.dot(c.e_L, left_part)
23     left_part = - np.dot(c.H_bar_inv, left_part)
24
25     #Right part
26     right_part = np.dot(c.e_RT, q)
27     right_part = np.dot(L_r, right_part)
28     right_part -= g_R() #Everything inside right parenthesis
29     inside_par = np.dot(self.L_r, W_minus)
30     right_part *= 1 / inside_par.item()
31     right_part *= Lambda_minus
32     right_part = W_minus * right_part
33     right_part = np.dot(c.e_R, right_part)
34     right_part = np.dot(c.H_bar_inv, right_part)
35
36
37     return left_part + right_part

```

Listing 3: SAT