A novel entropy stable discontinuous Galerkin spectral element method for the shallow water equations on GPUs

Inaugural - Dissertation

zur

Erlangung des Doktorgrades

der Mathematisch-Naturwissenschaftlichen Fakultät

der Universität zu Köln

vorgelegt von

Niklas Wintermeyer

aus Düsseldorf

Köln, 2019

Berichterstatter:	Prof. Dr. Gregor J. Gassner
	Prof. Dr. Claus-Dieter Munz
Tag der Abschlussprüfung:	14.1.2019

Abstract

We present a novel arbitrary high-order accurate nodal discontinuous Galerkin spectral element approximation for the non-linear two dimensional shallow water equations on unstructured, possibly curved, quadrilateral meshes. The scheme is designed such that it is entropy stable and well-balanced for non-constant, possibly discontinuous, bathymetry. We call the resulting method the entropy stable discontinuous Galerkin spectral element method (ESDGSEM).

The scheme is derived from an equivalent flux differencing formulation of a specific split form of the equations. We prove that this discretization yields a conservative approximation. Combined with a special numerical interface flux function, the method exactly preserves the total energy as well, which is a mathematical entropy function for the shallow water equations. By adding interface dissipation in a controlled way to the baseline entropy conservative scheme we guarantee the entropy stability of the scheme. Finally, we prove that the particular choice of source term discretization leads to a well-balanced approximation for the presented split form discretization.

We extend the entropy stable scheme with a shock capturing technique and a positivity preservation capability to handle dry areas. For the shock capturing, we introduce an artificial viscosity to the equations and prove that the numerical scheme remains entropy stable. We add a positivity preserving limiter to guarantee non-negative water heights as long as the mean water height is non-negative. We prove that non-negative mean water heights are guaranteed under a certain additional time step restriction for the entropy stable numerical interface flux. All of these modifications preserve the entropy inequality, maintain the well-balanced property and work on unstructured, possibly curved, quadrilateral meshes.

Furthermore, we implement the method on GPU architectures using the abstract language OCCA, a unified approach to multi-threading languages. We thoroughly analyze the scheme and compare it to a standard DG implementation. Our findings show that the entropy stable scheme is well suited to GPUs as the necessary extra calculations do not negatively impact the runtime up to reasonably high polynomial degrees (around $N = 7$).

We provide numerical examples that verify the theoretical properties of the entropy stable method. Also, we demonstrate the increased robustness of the method with a series of challenging test cases that require both, the shock capturing and positivity limiter.

Finally, we validate the entropy stable method by simulation the real world example of the 2004 Indian Ocean tsunami. We obtain good approximations of the tsunami arrival times when comparing to tide measurement data on the Indian coast.

Kurzzusammenfassung

In dieser Arbeit präsentieren wir ein neues Discontinuous Galerkin Spektrale Elemente Verfahren hoher Ordnung zur Approximation der nicht-linearen Flachwassergleichungen auf unstrukturierten, möglicherweise gekrümmten Gittern. Das Verfahren ist so konstruiert, dass es Entropie-stabil ist und die well-balanced Eigenschaft hat. Wir bezeichnen dieses Verfahren mit ESDGSEM.

Das Verfahren wird anhand einer spezifischen alternativen Formulierung der Flachwassergleichungen konstruiert. Durch Umformulierung der Diskretisierung der alternativen Formulierung in ein Flux-Differencing Verfahren ist es uns möglich zu zeigen, dass das Verfahren konservativ ist, obwohl es nicht auf den Gleichungen in konservativer Form beruht. Durch Verwendung eines speziellen numerischen Flusses an den Elementkanten können wir außerdem zeigen, dass die Approximation die totale Energie als zusätzliche konservative Variable behandelt. Da die totale Energie eine Entropie-Funktion für die Flachwassergleichungen ist, ist das Verfahren somit Entropie-konservativ. Durch kontrolliertes Hinzufügen von numerischer Dissipation an den Elementkanten können wir garantieren, dass das Verfahren Entropie-stabil ist. Weiterhin zeigen wir, dass das Verfahren für eine spezielle Diskretisierung des Quellterms auch well-balanced ist.

Wir führen künstliche Viskosität in das System der Flachwassergleichungen ein, um die Oszillationen im Falle von Schocks zu dämpfen. Außerdem erweitern wir das Entropie-stabile Verfahren um einen Positivitätslimiter, der nicht-negative Wasserhöhen garantiert. Wir zeigen, dass beide Erweiterungen die Entropie-Stabilität des Verfahrens beibehalten. Ebenfalls beweisen wir die Funktionalität des Positivitäts-limiters. Für die in dieser Arbeiten verwendeten Entropie-stabilen numerischen Flussfunktionen war dies zuvor nicht bekannt.

Des Weiteren implementieren wir das ESDGSEM unter Verwendung von OCCA auf modernen Grafikkarten. Unsere Untersuchungen ergeben, dass das ESDGSEM hervorragend auf die Architektur moderner Grafikkarten passt. Trotz eines höheren erforderlichen Rechenaufwands im Vergleich zu herkömlichen DG-Verfahren ist für Polynomgrade von $N \leq 7$ kein Laufzeitunterschied festzustellen.

Mit einer Reihe von numerischen Beispielen verifizieren wir die theoretischen Ergebnisse bezüglich Konvergenz und Konservatität. Außerdem zeigen wir die well-balanced Eigenschaft numerisch und testen die Robustheit des Verfahrens anhand einiger numerisch herausfordernder Tests. Diese beinhalten sowohl starke Schocks als auch Trockenflächen und benötigen somit zwingend die Erweiterungen des Verfahrens.

Abschließend verwenden wir das entwickelte Verfahren, um den Tsunami im Indischen Ozean aus dem Jahr 2004 zu simulieren. Unsere Ergebnisse vergleichen wir dabei mit an der Küste Indiens aufgezeichneten Daten und beobachten gute Approximationen für die Ankunftszeiten des Tsunamis.

Danksagung

Ein besonderes Wort des Dankes möchte ich an meinen Betreuer Gregor Gassner richten, der mir die Möglichkeit zur Promotion gegeben hat und mit seinem Fachwissen und seinen Ideen immer eine sehr hilfreiche Unterstützung war. Dass auch die Themen abseits der Promotion nicht zu kurz gekommen sind hat sich immer positiv auf unsere Arbeitsgruppe ausgewirkt.

Die häufig zum Nachdenken anregenden Gespräche in der Arbeitsgruppe, sei es in der Mittagspause oder nach langen Arbeitstagen, waren stets eine willkommene Abwechslung. Hierbei gilt mein Dank insbesondere meinem Bürokollegen Andrew, der immer mit Rat und Tat zur Seite stand. Auch bei Lukas und Marvin und Toni, die sich mit aufmerksamen Kleinigkeiten wie etwa besonderen Traditionen zum Geburtstag und generellen Lebensweisheiten stets als große Inspiration erwiesen haben.

Ferner möchte ich mich bei Tim Warburton und seiner Arbeitsgruppe bedanken, die mich für einige Wochen sehr herzlich aufgenommen haben und wo ich viel lernen durfte. Die Zeit hat mich sehr geprägt und war eine tolle Erfahrung.

Mein ganz besonderer Dank geht an meine Eltern Kathrin und Rainer, die mir meinen bisherigen Lebensweg so ermöglicht haben und immer mit hilfreichen Ratschlägen zur Seite standen. Auch bei meinem Bruder Robert, meinen Freunden und den Jungs von Fussball möchte ich mich für die tolle Zeit bedanken, da sie immer für ausreichend Abwechslung abseits der Promotion gesorgt haben.

Außerdem möchte ich mich natürlich bei meiner Freundin Johanna bedanken, die auch in schwierigen Phasen immer eine kleine, aber großartige Unterstützung für mich war.

Contents

List of Symbols and Abbreviations

We denote continuous variables by lower case letters, e.g., w for a conservative variable. Discrete approximations to these quantities are denoted by upper case letters, e.g., W. We also indicate a nodal approximation by the indexing. For example, in the context of the shallow water equations h_i refers to the approximated value of quantity h at node i. We denote matrix operators by bold letters, e.g., $\mathbf{D}$ or $\mathbf{S}$.

Alphanumeric Symbols

b	Bottom topography in the shallow water equations
c	Wave celerity $c = \sqrt{gh}$
e	Entropy function
$\mathscr{E}$	Discrete entropy function
E_{kin}	Kinetic Energy
E_{pot}	Potential Energy
E_{total}	Total Energy
E_0	Reference element in one or two dimensions
$\mathcal{E}$	Discrete product rule error term
f_{cor}	Coriolis parameter
$\vec{f}$, $\vec{g}$	Physical fluxes
$\vec{f}^v$, $\vec{g}^v$	Viscous fluxes
$\bar{f}$, $\bar{h}$	Fluxes on the complimentary grid
$\vec{\tilde{f}}$, $\vec{\tilde{g}}$	Contravariant physical fluxes
$\vec{\tilde{f}}^v$, $\vec{\tilde{g}}^v$	Contravariant viscous fluxes
$\vec{F}^{\#}$, $\vec{G}^{\#}$	Numerical two-point fluxes
$\vec{\tilde{F}}^{\#}$, $\vec{\tilde{G}}^{\#}$	Curvilinear numerical two-point fluxes
$\bar{\tilde{F}}$, $\bar{\tilde{G}}$	Contravariant fluxes on the complimentary grid
$\mathcal{F}$, $\mathcal{G}$	Entropy fluxes

g	Gravitational constant
h	Water height in the shallow water equations
H	Total water height in the shallow water equations
J	Polynomial approximation of the Jacobian of the transformation
$\mathcal{J}$	Jacobian of the mapping
$\mathcal{J}^{\text{surf}}$	Surface Jacobian
$\mathcal{J}\vec{a}^1$, $\mathcal{J}\vec{a}^2$	Volume weighted contravariant basis vectors
$J\vec{a}^1$, $J\vec{a}^2$	Discrete volume weighted contravariant basis vectors
ℓ	Lagrange basis functions
$\vec{\mathcal{L}}_\xi$, $\vec{\mathcal{L}}_\eta$	Spatial operators in DG and flux differencing schemes
N	Polynomial degree
N_E	Number of elements per workgroup on GPUs
$\vec{n}$	Normal vector in physical space
$\hat{n}$	Normal vector in reference space
$\vec{q}$	Vector of entropy variables
$\mathcal{R}$	Spatial DG operator for Runge-Kutta updates
s	Shock speed
$\vec{s}$	Continuous source term vector
u, v	Fluid velocities in the shallow water equations
$\vec{\mathcal{U}}$, $\vec{\mathcal{V}}$	Gradient variables for artificial viscosity
$\vec{w}$	Vector of degrees of freedom for conservation or balance laws
x, y	Coordinates in physical space
x_ξ, y_ξ, x_η, y_η	Metric terms

$\mathbf{A}$, $\mathbf{A}_f$, $\mathbf{A}_g$	Jacobian of the fluxes $\vec{f}$ and $\vec{g}$
$\mathbf{B}$	Boundary matrix defined by $\mathbf{B} = \operatorname{diag}(-1, 0, \ldots, 0, +1)$
$\mathbf{D}$	Differentiation matrix
$\hat{\mathbf{D}}$	Scaled differentiation matrix of weak form DG
$\tilde{\mathbf{D}}$	Modified differentiation matrix
$\mathbf{H}$	Entropy Jacobian
$\mathbf{M}$	Mass matrix
n	Manning coefficient
$\mathcal{O}(\cdot)$	Landau notation for asymptotic behavior
$\mathbf{Q}$	Summation-by-parts matrix
$\mathbf{R}$, $\mathbf{R}_f$, $\mathbf{R}_g$	Matrices containing eigenvectors of the flux Jacobians
$\mathbf{S}$	Weighted surface matrix defined by $\mathbf{S} = \operatorname{diag}\left(\frac{1}{w_0}, 0, \ldots, 0, -\frac{1}{w_N}\right)$
$\mathbf{T}$	Diagonal scaling matrix used for interface dissipation
$\mathbf{V}$	Vandermonde matrix
$\{\{w\}\}$	Average of two states of w
$[\![w]\!]$	Jump between two states of w
∇	Nabla-operator in physical space
$\hat{\nabla}$	Nabla-operator in reference space

α	Maximum wave speed
$\vec{\Gamma}$	Boundary curves for curved elements
Δ	Denotes differences between values
Δt^a and Δt^v	Advective and diffusive time steps
$\boldsymbol{\Delta}$	Differencing matrix
ϵ_0	Base viscosity parameter
ϵ	Dynamically computed viscosity parameter
θ	Parameter in Positivity Limiter
λ	Eigenvalue of flux Jacobian
λ^v	Eigenvalue of diffusive flux Jacobian
ξ, η	Coordinates in reference space
$\bar{\xi}$, $\bar{\eta}$	Coordinates on the complimentary grid
σ_{dof}	Smoothness indicator
σ_{min},σ_{max}	Bounding parameters for shock detection
τ_{bx}, τ_{by}	Manning friction source terms
ϕ	Test function in the variational formulation
Φ	Entropy potential
Ψ, Ψ_f and Ψ_g	Entropy flux potentials
Ω	Physical domain
ω	Earth's angular velocity

Abbreviations

AV	Artificial viscosity
BR1	Discretization of viscous terms by Bassi and Rebay
BW	Bandwidth
CFL	Courant-Friedrichs-Lewy number
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
DFL	Parameter for diffusive time step
DG	Discontinuous Galerkin
DGSEM	Discontinuous Galerkin Spectral Element Method
DOF	Degree of freedom
DNS	Direct Numerical Simulation
EC	Entropy conservative
EOC	Experimental order of convergence
ENO	Essentially non-oscillatory
ES	Entropy stable
ESDGSEM	Entropy stable discontinuous Galerkin spectral element method
FD	Finite difference
FEM	Finite element method
FV	Finite volume
GPU	Graphics Processing Unit
GP-GPU	General-purpose GPU
HLL	Harten-Lax-Leer
HPC	High Performance Computing
LES	Large Eddy Simulation
LGL	Legendre-Gauss-Lobatto
MHD	Magnetohydrodynamics
MPI	Message Passing Interface
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
RK	Runge-Kutta
SAT	Simultaneous Approximation Terms
SBP	Summation-by-parts
SIMD	Single instruction, multiple data
SIMT	Single-Instruction, Multiple-Thread
SPD	Symmetric positive definite
SPH	Smoothed particle hydrodynamics
SM	Shared Memory
SSPRK	Strong stability preserving Runge-Kutta
SW	Shallow water
TVD	Total variation diminishing
VRAM	Video Random Access Memory
WENO	Weighted essentially non-oscillatory

1. Introduction

In the world of numerical simulation there exist a vast variety of methods to approximate fluid behavior each exhibiting unique properties that make them more or less suitable toward individual application. In this work, we consider the approximation of hyperbolic balance laws. There are three key characteristics that define the applicability of a numerical method. First, there is the solution quality. Highly resolved solutions are necessary to accurately reflect finer physical phenomena, for example in the field of turbulence modeling. Second, there is the robustness of a scheme. It describes the capability of reliably producing a meaningful solution, even for challenging problems. Lastly, there is the computational efficiency. The resources available for a numerical simulation are generally limited and must be spent reasonably. Balancing these criteria is very important when designing a numerical method. Many available schemes shine in one or two of these categories, at the cost of the other. Traditional low-order finite volume methods, for example, are among the most reliable schemes available. However, the solution quality is lacking as they struggle to resolve finer features. This behaviour can be observed at shocks, which are treated robustly, but smeared out. In long time simulations these inaccuracies may deteriorate and lead to unacceptable solutions [217].

In this thesis, we are interested in spectrally accurate schemes. Discontinuous Galerkin (DG) spectral element methods separate the domain into elements and approximate the local solutions by polynomials. To resolve the discontinuities at interfaces elements are coupled by a numerical flux, a concept borrowed from the finite volume community. Within the DG framework, high-order accuracy is achieved by simply increasing the order of the polynomial approximations. Generally, high-order methods are not only capable of accurately resolving finer physical features, they are also known to be very effective in terms computational cost, requiring less work at a given accuracy than low-order schemes [107, 54]. Due to the local ansatz of the approximation, DG methods are well-suited for parallelization. Neighbouring elements are weakly coupled and only need to exchange interface data and a bulk of the computational work is performed locally. Furthermore, these type of schemes are very flexible in handling complex geometries as the elements can have a variety of shapes with possibly curved sides.

Solutions of non-linear systems of hyperbolic conservation laws can develop shocks in finite time, even if the initial data is smooth [178, 174]. To allow non-smooth solutions, weak formulations of the conservation laws are typically considered. Then, the Lax-Wendroff theorem guarantees that if a conservative scheme converges, it converges to a weak solution [147]. Unfortunately, weak solutions are not unique, posing the non-trivial task of finding the physically relevant solution out of the infinite family of weak solutions [234]. Entropy stable methods address part of this difficulty by requiring the solution to satisfy an additional entropy inequality [149, 223, 107]. In the context of gas dynamics, this can be interpreted as consistency to the second law of thermodynamics. Also, since the entropy is bounded in L_2, they demonstrate increased robustness when compared to their non entropy stable counterparts [64, 253]. This increased robustness is important, especially in the context of high-order methods. High-order approximations tend to introduce oscillations at shock fronts, a phenomenon first observed by Gibbs [92]. If these spurious oscillations are too severe, they may lead to unphysical values and cause the simulation to crash. Other strategies to address this problem include slope limiters or added artificial viscosity, but this may come with a loss of accuracy when applied to high-order methods [32]. However, until recently there have not been many high-order entropy stable methods available. The E-schemes proposed by Osher [192] are entropy stable for all entropy pairs but limited to first order accuracy. Tadmor presented a family of second order entropy stable finite volume schemes, where the evaluation of the entropy conservative flux is based on a path integral in phase space [235, 234]. LeFloch and Rohde built on Tadmor's foundation to find fully discrete entropy stable finite volume schemes of third order accuracy [152]. Finally, a framework for entropy stable finite volume schemes of arbitrary order was proposed by Fjordholm, Mishra and Tadmor in 2012 [69].

In the context of high-order finite difference and DG methods, Carpenter and Fisher have established a framework for the construction of entropy stable schemes in a remarkable series of papers [32, 64, 66]. Their proposed systematic approach towards deriving entropy stable schemes is based on summation-by-parts (SBP) operators. By using flux difference formulations on a complimentary grid, the authors are able to construct entropy conservative high-order finite difference and DG approximations from entropy conservative numerical fluxes frequently used in the low-order finite volume framework. These entropy conservative *flux differencing* schemes are extended by adding dissipation to the numerical interface fluxes in a specific way, such that the entropy inequality holds globally. Gassner et al. [82, 62] have found an equivalency between schemes in the framework of Carpenter and Fisher and certain DG discretizations of split formulations of systems of conservation laws. The DG method must use derivative operators with the SBP property. This remarkable result from Fisher paves the way to combine the beneficial robustness of split formulations with the flux differencing framework and its systematic approach towards entropy stability.

The aim of this thesis is to build on the framework by Carpenter and Fisher to derive a novel entropy stable discontinuous Galerkin method for the shallow water (SW) equations. The SW equations are a set of non-linear hyperbolic balance laws widely used in oceanic and atmospheric modeling as well as hydraulic engineering [15, 7, 250, 167, 50, 85, 242]. Specific applications include the simulation of tsunamis [20, 75] and storm surges [113, 12, 11]. For the SW equations it is critical that the numerical method maintains a balance between the source term which includes the gradient of the bottom topography and the hydrostatic pressure. Only for such *well-balanced* methods, will the important steady state solution called "lake-at-rest" be preserved for all time. Many real world applications, such as tsunamis, are modeled as only slight perturbations of this steady state. A numerical method that is not well-balanced may introduce unphysical numerical waves, polluting the solution.

Thus, we require the new method to not only be entropy stable but well-balanced as well. To this end, we present a specific split formulation of the shallow water equations as well as a special variant of DG methods called the discontinuous Galerkin spectral element method (DGSEM). The DGSEM uses Legendre-Gauss-Lobatto nodes such that its derivative operators have the SBP property. We show that if the split form of the shallow water equations is discretized by the DGSEM and specific entropy conservative interface fluxes are used, the resulting method is globally entropy conservative. Also, we provide an equivalent reformulation into Carpenter and Fisher's flux differencing framework, immediately proving that the scheme is conservative - a property easily lost when discretizing systems in split form. This reformulation also offers an efficient implementation as it replaces the computation of many volume integrals for the individual split form terms by a single volume integral with a special, albeit more expensive, numerical flux. We call the resulting method the entropy stable DGSEM (ESDGSEM).

In the modern high performance computing (HPC) environment, the trend has shifted from developing faster central processing units (CPUs) towards increasing the number of CPUs used in the computation. Numerical methods, such as DG, where the work can be easily divided into separate pieces that can be treated individually, are a perfect fit for the parallel environment of modern supercomputers. Recently, this has led to the use of graphics processing units (GPUs) for scientific computing purposes. Modern GPUs are many-core architectures that deliver extremely fast single thread performance executed on thousands of cores in parallel. Consequently, the world's most powerful supercomputers now usually feature combinations of many CPU cores and GPUs on each node. The calculation of floating point operations on modern GPUs is oftentimes so fast, that the actual computations are no longer the limiting factor, but the time spent on data transfer. This makes methods such as the high-order DGSEM, that put a high computational load on each node but require little data exchange, very well suited for GPU implementations.

We will show that the reformulation of the split form DGSEM in the flux differencing framework by Carpenter and Fisher is particularly well suited to implementation on GPU architectures. The additional cost of evaluating the new numerical volume fluxes is handled so well, that it does not negatively affect the runtime of the simulation for frequently used polynomial orders $N \leq 7$. Additionally, computations for these widely used polynomial degrees remain memory bound and the ESDGSEM is as fast as a standard DG implementation.

The outline of this thesis is as follows. In Chapter 2, we start with an introduction into the theory of conservation laws and the shallow water equations in particular. Here, we will formally explain the concept of entropy stability and develop suitable entropy functions for the shallow water equations. We follow this physical discussion by an overview of numerical methods in Chapter 3. We also derive the DGSEM in one and two dimensions and outline the development of entropy stable methods. We discuss the SBP framework by Carpenter and Fisher and the relationship to split formulations in Chapter 4. Finally, we derive the entropy stable DG scheme in Chapter 5. First, we derive purely entropy conservative methods for the one and two dimensional shallow water equations. We start with the simpler one dimensional case to highlight the important ideas before we move to two dimensions and possibly curved geometries. Furthermore, we explain how to add dissipation to the entropy conservative interface fluxes to guarantee entropy stability. This allows us to formally present the full ESDGSEM. We proceed by presenting two extensions to the method. First, we discuss shock capturing by artificial viscosity. This modification helps to reduce any oscillations near shocks and greatly stabilizes the scheme. The second modification is a positivity limiter. Practical examples of the shallow water equations may include the dynamic wetting and drying of the domain, for example if a dry area is flooded by a breaking dam. Such a limiter is necessary to prevent unphysical negative water heights that may crash the simulation. Next, we present an efficient implementation of the ESDGSEM on modern GPU architectures in Chapter 6. We discuss several optimization techniques specific to the architecture that significantly enhance the performance. We thoroughly analyze the performance and compare it to a standard DG implementation. A numerical validation of the theoretical properties of the scheme and its implementation can be found in chapter 7. We also use several challenging examples from the literature to fully stress test the shock capturing capabilities and the positivity limiter. In chapter 8, we finally use the ESDGSEM in a real world application. We simulate the 2004 Indian Ocean tsunami, one of the biggest natural catastrophes of the modern age. We approximate the initial state by the Okada model and compare our simulation results with actual measurements from several stations on the Indian coast. The comparison shows that the ESDGSEM gives good approximations of the tsunami arrival times and reinforce that numerical simulations could help in studying such natural phenomenons. Lastly, a summary of this work along with an outlook on possible further research is provided in chapter 9.

2. Conservation Laws

2.1. Concepts for conservation laws

In this work, we are interested in the numerical approximation of solutions to the shallow water (SW) equations. The SW equations are a non-linear hyperbolic system of PDEs available in one and two spatial dimensions. They consist of a conservation law for the water height and a balance law in each spatial direction for the momentum equations. The balance laws feature a source term proportional to the gradient of the bottom topology. While the focus of this work lies on the approximation of the two dimensional system, we will introduce many concepts for the simpler one dimensional case first. Thus, we will establish the basics for both sets of equations.

The mandatory first step in the development of any discrete method is the investigation of the equations itself. From this continuous analysis we can develop important concepts for the numerical solutions of the system. These concepts include upwinding [46] and conservation [149] but also a principle called entropy stability [233]. Entropy stability plays a pivotal role in this work and will be thoroughly introduced in the following sections. We note that this overview on the solutions of hyperbolic conservation laws should only serve as a motivation towards the development of the numerical scheme and is far from complete. For a more thorough overview we refer to the works of Lax [148, 147] or teaching literature such as [107, 106, 240] and others.

2.1.1. Systems of conservation laws in one dimension

We now explore solution concepts regarding systems of conservation laws in one spatial dimension on the domain $\Omega \subset \mathbb{R}$ of the form

$$\vec{w}_t + \vec{f}(\vec{w})_x = 0. \tag{2.1}$$

The system is completed with suitable initial conditions $\vec{w}_0(x) = \vec{w}(x,0)$ and appropriate boundary conditions. Using the flux Jacobian $\mathbf{A}(\vec{w}) = \vec{f}_{\vec{w}}(\vec{w})$, the system can be rewritten in quasi-linear form

$$\vec{w}_t + \mathbf{A}(\vec{w})(\vec{w})_x = 0, \tag{2.2}$$

and is (strictly) hyperbolic, if $\mathbf{A}$ has a full set of (distinct) real eigenvalues.

A fundamental difficulty is the fact that discontinuities in the solution can develop in finite time, even for smooth initial conditions [178, 174]. Once discontinuities, or shocks, are present in the solution, the classical derivatives in system (2.1) are not valid anymore. To allow for non-smooth solutions, we consider a weak formulation of the conservation law. The weak form is obtained by first multiplying each equation in (2.1) by a smooth test function $\phi \in C^\infty$ with compact support $\Omega \times [0, T)$. Then, the equations are integrated in space and time. Finally, the temporal and spatial derivatives are moved onto the test function by integration by parts. This leads to the conservation law in weak form

$$\int_{t_0}^{T} \int_{\Omega} \left(\vec{w}(x,t)\phi_t + \vec{f}(\vec{w})\phi_x \right) \, dx \, dt = \int_{\Omega} \left(\vec{w}_0(x)\phi(x,0) \right) \, dx. \tag{2.3}$$

A function $\vec{w}(x,t)$ is a weak solution of the original system (2.1) if it satisfies the conservation law in weak form (2.3) for all feasible test functions ϕ. A smooth solution to the conservation law in differential form (2.1) is respectively called a strong solution.

The development of shocks and its behaviour can be further understood by analyzing the characteristics of the solution. The characteristics, or characteristic curves, are curves $x(t)$ in the $x-t$ plane along which the PDE becomes an ordinary differential equation (ODE). The solution has a constant speed along its characteristics, which is also called the characteristic speed. An intersection of two characteristics is equivalent to the formation of a shock in the solution [240]. It is possible to derive the velocity of the propagation of the discontinuity, or shock speed, s [148, 240] as

$$s[\![\vec{w}]\!] = [\![\vec{f}]\!], \tag{2.4}$$

where we have introduced the jump operator $[\![\cdot]\!]$ defined by

$$[\![a]\!] := a_R - a_L, \tag{2.5}$$

which denotes the difference between the states to the left, $\vec{w}_L$ and $\vec{f}_L$, and the right, $\vec{w}_R$ and $\vec{f}_R$, of the shock. The equations (2.4) are called the Rankine-Hugoniot conditions and set the shock speed in relation to the jumps in fluxes and conservative variables across the shock. A piecewise differentiable solution $\vec{w}$ that contains discontinuities is a weak solution exactly if it fulfills the conservation law in differential form in the smooth region and the Rankine-Hugoniot condition (2.4) holds across the discontinuities [70].

The downside of considering weak solutions is that uniqueness is generally lost [149, 174]. This raises the question of how to select a *correct*, or physically relevant, solution out of the family of weak solutions. For **scalar** conservation laws with convex physical flux f, an entropy condition can be imposed onto the solution,

$$f_x(w_L) \geq s \geq f_x(w_R). \tag{2.6}$$

The condition (2.6) follows from the Rankine-Hugoniot condition and is called the Lax entropy condition. It is derived by adhering to the idea that characteristics must travel into shocks and not originate from them [240]. Any solution that does not satisfy the entropy condition (2.6) contains an entropy-violating shock and can be deemed physically incorrect [240].

To extend this idea to systems of conservation laws and to establish a formal framework,we introduce the notion of an entropy function $e(\vec{w})$ and the corresponding entropy flux $\mathcal{F}(\vec{w})$, often denoted as an entropy-entropy flux pair $(e, \mathcal{F})$. The entropy function must be convex and the entropy-entropy flux pair must satisfy the following contractability condition

$$\langle \vec{q}, \vec{f}_{\vec{w}} \rangle = \mathcal{F}_{\vec{w}}, \tag{2.7}$$

where we have introduced the new set of variables $\vec{q} = e_{\vec{w}}$, called the entropy variables. The entropy variables have the interesting property that they symmetrize the system of conservation laws. This can be seen by treating the entropy variables as independent variables and writing

$$\vec{w}_t + \vec{f}_x = \vec{w}_{\vec{q}}\,\vec{q}_t + \vec{f}_{\vec{q}}\,\vec{q}_x = 0, \tag{2.8}$$

with symmetric matrices $\vec{w}_{\vec{q}} = [\vec{w}_{\vec{q}}]^T$ and $\vec{f}_{\vec{q}} = [\vec{f}_{\vec{q}}]^T$.

Since the entropy function is convex, the Hessian $e_{\vec{w}\vec{w}} = \vec{q}_{\vec{w}}$ is symmetric positive definite (SPD),

$$\langle\, \zeta\,,\, e_{\vec{w}\vec{w}}\, \vec{\zeta}\,\rangle > 0, \quad \forall \vec{\zeta} \neq 0, \tag{2.9}$$

and provides a one-to-one mapping from conservative to entropy variables, $\vec{w} \to \vec{q}$. SPD matrices are invertible and the inverse $(\vec{q}_{\vec{w}})^{-1}$ is SPD as well [103].

To further develop the entropy framework, the concept of the *entropy solution* is introduced. We derive a regularization of the original conservation law (2.1) by introducing a viscous term on the right hand side

$$\vec{w}^{\epsilon}_t + \vec{f}(\vec{w}^{\epsilon})_x = \epsilon\, \vec{w}^{\epsilon}_{xx}. \tag{2.10}$$

The viscous term makes all discontinuities theoretically resolvable, thus guaranteeing the existence of a strong solution for all time [150, 107, 63]. The entropy solution is then defined as the limit of the viscous solution, $\vec{w}^{\epsilon}$, for $\epsilon \to 0$,

$$\vec{w} = \lim_{\epsilon \to 0} \vec{w}^{\epsilon} \qquad \text{in } L^1_{\text{loc}}, \tag{2.11}$$

the space of locally integrable functions. The regularized conservation law (2.10) can be used to derive an entropy inequality condition. Contracting (2.10) with the entropy variables $\vec{q}$ leads to a regularized entropy equation

$$e^{\epsilon}_t + \mathcal{F}^{\epsilon}_x = \epsilon \langle\, \vec{q}^{\,\epsilon}, \vec{w}^{\epsilon}_{xx}\,\rangle. \tag{2.12}$$

The product rule can be used to rewrite the viscous terms as

$$\epsilon \langle\, \vec{q}^{\,\epsilon}, \vec{w}_{xx}\,\rangle = \epsilon \langle\, \vec{q}^{\,\epsilon}, \vec{w}_x\,\rangle_x - \epsilon \langle\, \vec{q}^{\,\epsilon}_x, \vec{w}_x\,\rangle. \tag{2.13}$$

For the last term in (2.13), we see that

$$\langle\, \vec{q}^{\,\epsilon}_x, \vec{w}_x\,\rangle = \langle\, e_{\vec{w}\vec{w}} \vec{w}_x, \vec{w}_x\,\rangle \geq 0, \tag{2.14}$$

since the Hessian of the entropy function is SPD. Thus, from inserting equations (2.13) and (2.14) into (2.12), we find the following entropy inequality for the regularized conservation law (2.10)

$$e^{\epsilon}_t + \mathcal{F}^{\epsilon}_x \leq \epsilon \langle\, \vec{q}^{\,\epsilon}, \vec{w}_x\,\rangle_x. \tag{2.15}$$

Integrating over the domain and taking the limit $\epsilon \to 0$ leads to an entropy inequality for the original conservation law (2.1) [150]

$$\int_\Omega e_t + \mathcal{F}_x \, dx \leq 0. \tag{2.16}$$

In the context of gas dynamics, the (mathematical) entropy inequality is closely related to the second law of thermodynamics, which dictates that the entropy in a closed system can only increase. As can be seen from the entropy inequality (2.16), this concept is applied in mathematics with a reversed sign convention, requiring that the entropy must decrease monotonically over time. Thus, a weak solution that satisfies an entropy inequality is in this case consistent to the second law of thermodynamics and more physically relevant than solutions that do not. This is especially true for systems such as the Euler equations where the mathematical entropy density is a physical entropy as well [173].

In addition to the entropy-entropy flux pair, there is also the entropy potential-entropy flux potential pair denoted by (Φ, Ψ) and defined by

$$\begin{aligned} \Phi(\vec{w}) &= \langle \vec{w}, \vec{q} \rangle - e(\vec{w}) \\ \Psi(\vec{w}) &= \langle \vec{f}, \vec{q} \rangle - \mathcal{F}(\vec{w}), \end{aligned} \tag{2.17}$$

that fulfill the following relation,

$$\Phi_{\vec{q}} = \vec{w}, \qquad \Psi_{\vec{q}} = \vec{f}. \tag{2.18}$$

The entropy potential and entropy flux potential can be used to reformulate the symmetrization (2.8) in terms of the Hessians of the entropy potential Φ and the entropy flux potential Ψ [89, 174],

$$\vec{w}_t + \vec{f}_x = \Phi_{\vec{q}\vec{q}} \vec{q}_t + (\Psi_f)_{\vec{q}\vec{q}} \, \vec{q}_x = 0. \tag{2.19}$$

Also, if $\vec{w}$ is a solution of the conservation law (2.1), then $\vec{q}(\vec{w})$ is a solution of

$$\left(\Phi_{\vec{q}}(\vec{w})\right)_t + \left(\Psi_{\vec{q}}(\vec{w})\right)_x = 0. \tag{2.20}$$

2.1.2. Systems of conservation laws in two dimensions with possible source terms

We follow similar steps as in the one dimensional case to establish the entropy framework for systems of conservation laws in two dimensions. Specifically, we need to extend the definition of the entropy-flux pair $(e, \mathcal{F})$ to an entropy-entropy flux triplet, $(e, \mathcal{F}, \mathcal{G})$ in order to apply the concepts to systems of conservation laws

$$\vec{w}_t + \vec{f}(\vec{w})_x + \vec{g}(\vec{w})_y = 0, \tag{2.21}$$

and balance laws

$$\vec{w}_t + \vec{f}(\vec{w})_x + \vec{g}(\vec{w})_y = \vec{s}, \tag{2.22}$$

in two dimensions. For systems of balance laws, we split the source term into parts that contain spatial derivatives in x and y directions, $\vec{s}^x$ and $\vec{s}^y$, and in terms that do not, $\vec{s}^0$, such that

$$\vec{s} = \vec{s}^x + \vec{s}^y + \vec{s}^0. \tag{2.23}$$

This is important, because the parts that contain spatial derivatives must be taken into the account for the contractability condition imposed on the entropy variables and entropy fluxes. We define the mathematical entropy functions for inviscid PDEs similar to [65] with added source term compatibility as in [68]. The first requirement is that the convex entropy function $e = e(\vec{w})$ contracts the fluxes and source term components in the following way

$$\begin{aligned} \langle\, \vec{q},\, \vec{f}_x - \vec{s}^x \,\rangle &= \mathcal{F}_x, \\ \langle\, \vec{q},\, \vec{g}_y - \vec{s}^y \,\rangle &= \mathcal{G}_y, \end{aligned} \tag{2.24}$$

where the components of the contracting vector, $\vec{q} = e_{\vec{w}}$, are the entropy variables and $\mathcal{F}$ and $\mathcal{G}$ are the entropy fluxes. Furthermore, the entropy variables must symmetrize the source term free system of PDEs if they assume the role of independent variables (i.e., $\vec{w} = \vec{w}(\vec{q})$). Then, the system of PDEs in terms of $\vec{q}$ can be written as

$$\vec{w}_t + \vec{f}_x + \vec{g}_y = \vec{w}_{\vec{q}}\,\vec{q}_t + \vec{f}_{\vec{q}}\,\vec{q}_x + \vec{g}_{\vec{q}}\,\vec{q}_y = 0, \tag{2.25}$$

with symmetry conditions: $\vec{w}_{\vec{q}} = [\vec{w}_{\vec{q}}]^T$, $\vec{f}_{\vec{q}} = (\vec{f}_{\vec{q}})^T$ and $\vec{g}_{\vec{q}} = (\vec{g}_{\vec{q}})^T$.

The entropy potential and the corresponding entropy flux potentials in two dimensions are defined by

$$\Phi := \langle\, \vec{q}, \vec{w}\,\rangle - e, \qquad \Psi_f := \langle\, \vec{q}, \vec{f}\,\rangle - \mathcal{F}, \qquad \Psi_g := \langle\, \vec{q}, \vec{g}\,\rangle - \mathcal{G}. \tag{2.26}$$

From the contraction condition (2.24) it follows that

$$\Phi_{\vec{q}} = \vec{w}, \qquad (\Psi_f)_{\vec{q}} = \vec{f}, \qquad (\Psi_g)_{\vec{q}} = \vec{g}. \tag{2.27}$$

The symmetrization (2.25) can then be formulated in terms of the *symmetric* Hessians of the entropy potential and the entropy flux potentials [234],

$$\vec{w}_t + \vec{f}_x + \vec{g}_y = \Phi_{\vec{q}\vec{q}}\,\vec{q}_t + (\Phi_f)_{\vec{q}\vec{q}}\,\vec{q}_x + (\Phi_g)_{\vec{q}\vec{q}}\,\vec{q}_y = 0. \tag{2.28}$$

2.2. The shallow water equations

The shallow water equations are a set of partial differential equations used to model a flow under a pressure surface that were developed by Saint-Venant in 1871 [51]. The equations are derived by depth-averaging the three dimensional Navier-Stokes equations under the following assumptions. First, horizontal length scales are assumed to be much larger than vertical ones. Then, vertical velocities are seen as negligible and treated as constant. In this case, the hydrostatic pressure (pressure exerted by a fluid at equilibrium due to gravity) depends linearly on depth. The density of the fluid is considered constant.

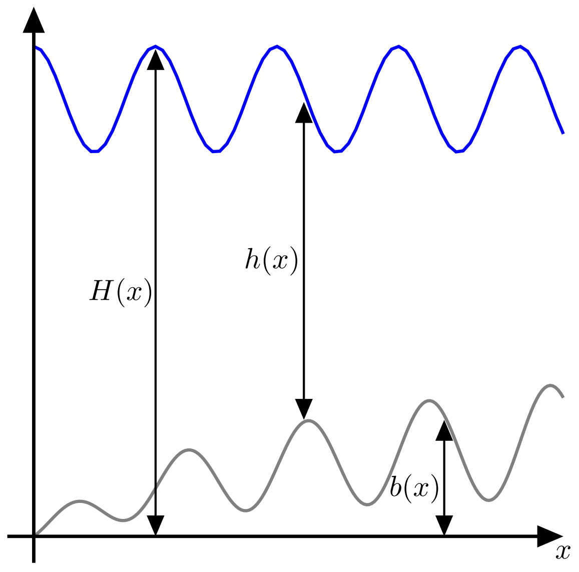


Figure 2.1.: Water height h, total water height H and bottom topography b for the one dimensional shallow water equations at a fixed time t.

We calculate the depth-integrated density which is also the fluid height and denote it by h. The horizontal velocities u and v are also understood as depth-averaged.

These assumptions are highly idealized but reasonably applicable, e.g., for flows such as oceans and coastal regions or rivers and channels [10]. Since a two dimensional system is far more cost effective to solve numerically, the shallow water equations are widely used in oceanic and atmospheric modeling. Applications include predicting tides as well as storm surge levels and coastline changes due to unusual events such as hurricanes or tsunamis, e.g. [20, 250, 167].

There are shallow water equations for one and two spatial dimensions. The one dimensional shallow water equations are also called the Saint-Venant equations and given by

$$\begin{aligned} h_t + (hu)_x &= 0, \\ (hu)_t + \left(hu^2 + \frac{1}{2} g\, h^2\right)_x &= -ghb_x, \end{aligned} \tag{2.29}$$

where h denotes the water height measured from the bottom topography b. Furthermore, u is the fluid velocity and g is the gravitational constant. The total water height is denoted by $H = h + b$, as illustrated in Figure 2.1.

The system (2.29) can be written in compact balance law form as

$$\vec{w}_t + \vec{f}_x = \vec{s}, \tag{2.30}$$

with

$$\vec{w} = \begin{pmatrix} h \\ hu \end{pmatrix}, \quad \vec{f} = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{pmatrix}, \quad \vec{s} = \begin{pmatrix} 0 \\ -ghb_x \end{pmatrix}. \tag{2.31}$$

The shallow water equations are a (strictly) hyperbolic system, as can be seen from the eigenvalues of the flux Jacobian $\mathbf{A} = \vec{f}_{\vec{u}}$, which are found to be

$$\lambda_1 = u + c, \qquad \lambda_2 = u - c, \tag{2.32}$$

with wave celerity $c = \sqrt{gh}$. These eigenvalues are all real and also distinct for $h > 0$.

The shallow water equations in two dimensions are also a system of hyperbolic balance laws given by

$$\begin{aligned} h_t + (hu)_x + (hv)_y &= 0, \\ (hu)_t + \left(hu^2 + \frac{1}{2} g\, h^2\right)_x + (huv)_y &= -ghb_x, \\ (hv)_t + (huv)_x + \left(hv^2 + \frac{1}{2} g\, h^2\right)_y &= -ghb_y, \end{aligned} \tag{2.33}$$

and share the nomenclature of its one dimensional version described above, e.g. the water height $h = h(x, y, t)$ measured from the bottom topography $b = b(x, y)$. We compactly write the system (2.33) as

$$\vec{w}_t + \nabla \cdot (\vec{f}, \vec{g})^T = \vec{s}, \tag{2.34}$$

with

$$\vec{w} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \quad \vec{f} = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}, \quad \vec{g} = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{pmatrix}, \quad \vec{s} = \begin{pmatrix} 0 \\ -ghb_x \\ -ghb_y \end{pmatrix}. \tag{2.35}$$

The hyperbolic nature of the system can, again, be observed in the eigenvalues of the flux Jacobians, which are

$$\begin{aligned} &\lambda_1^f = u + c, \qquad \lambda_2^f = u, \qquad \lambda_3^f = u - c, \qquad \text{for } \mathbf{A}_f = \vec{f}_{\vec{u}}, \\ &\lambda_1^g = v + c, \qquad \lambda_2^g = v, \qquad \lambda_3^g = v - c, \qquad \text{for } \mathbf{A}_g = \vec{g}_{\vec{u}}, \end{aligned} \tag{2.36}$$

which are all real for $h \geq 0$ and also distinct for each flux Jacobian, if $h > 0$.

The shallow water equations come with several unique challenges. One of these is the correct handling of certain steady state solutions. The most prominent example is the "lake-at-rest" state given by

$$\begin{aligned} &h + b = \text{const}, \\ &u = v = 0. \end{aligned} \tag{2.37}$$

In these steady state cases, there is a balance between the derivative of the flux pressure term $\frac{1}{2}gh^2$ and the slope of the bottom topography. The preservation of the states (2.37) is a highly desirable property for any numerical solver. If the balance between flux divergence and source term is not maintained discretely, numerical waves are introduced to the solution and can lead to numerical storms [180]. This is problematic, especially for test cases that are modeled as just small perturbations of such steady states [156]. A numerical solver that exactly (to machine precision) preserves the "lake-at-rest" state is called *well-balanced* [83, 26, 180]. There exist more general moving steady state solutions given by

$$\begin{aligned} hu &= \text{constant}, \\ \frac{1}{2}u^2 + gH &= \text{constant}. \end{aligned} \tag{2.38}$$

While these generalized steady states are not considered for the development of the numerical method in this work, we point out that such numerical schemes exist, e.g. for high-order finite volume methods [181] or flux-vector splitting methods [91].

Another natural phenomenon is the flooding of dry areas such as islands or beaches, especially in the prediction of tidal waves. These dry areas provide a difficult numerical challenge. Even small oscillations in the solution may render a numerical scheme unstable. Strategies to address this issue include positivity limiters that remove problematic oscillations, as we will discuss in Section 5.5.

2.3. Entropy analysis for the shallow water equations

In the shallow water framework, we distinguish between kinetic energy

$$E_{\text{kin}} := \frac{1}{2}h\left\|\vec{u}\right\|^2, \tag{2.39}$$

and potential energy,

$$E_{\text{pot}} := \frac{1}{2}gh^2 + ghb. \tag{2.40}$$

The total energy is the sum of kinetic and potential energy

$$E_{\text{total}} := E_{\text{kin}} + E_{\text{pot}} = \frac{1}{2}h\left\|\vec{u}\right\|^2 + \frac{1}{2}gh^2 + ghb. \tag{2.41}$$

We will show that the total energy serves as an entropy function for the shallow water equations. We prove this for the one dimensional and two dimensional shallow water equations separately in the following Lemmas, starting with the one dimensional case.

Lemma 1 (Entropy function for one dimensional shallow water equations)**.** *The total energy serves as a mathematical entropy function for the one dimensional shallow water equations* (2.29)

$$e := E_{total} = \frac{1}{2}hu^2 + \frac{1}{2}gh^2 + ghb. \tag{2.42}$$

The entropy variables are given by

$$q_1 = g(h+b) - \frac{1}{2}u^2, \qquad q_2 = u, \tag{2.43}$$

and the entropy flux is

$$\mathcal{F} = \frac{1}{2}hu^3 + ghu(h+b). \tag{2.44}$$

The entropy potential - entropy flux potential pair (Φ, Ψ) *are given by*

$$\Phi = \frac{1}{2}gh^2, \qquad \Psi = \frac{1}{2}gh^2u, \tag{2.45}$$

and fulfill the symmetry property

$$\Phi_{\vec{q}} = \vec{w}, \qquad \Psi_{\vec{q}} = \vec{f}. \tag{2.46}$$

Proof. See Appendix A.1. ■

Similarly we show that the total energy (2.41) is a mathematical entropy function for the two dimensional equations (2.33) as well.

Lemma 2 (Entropy function for 2D shallow water equations)**.** *The total energy is an entropy function for the two dimensional shallow water equations* (2.33)

$$e := E_{total} = \frac{1}{2}h\left(u^2+v^2\right) + \frac{1}{2}gh^2 + ghb. \tag{2.47}$$

The entropy variables are given by

$$q_1 = g\,(h+b) - \frac{1}{2}u^2 - \frac{1}{2}v^2, \qquad q_2 = u, \qquad q_3 = v, \tag{2.48}$$

and the entropy fluxes are

$$\begin{aligned} \mathcal{F} &= \frac{1}{2}hu(u^2+v^2) + ghu(h+b), \\ \mathcal{G} &= \frac{1}{2}hv(u^2+v^2) + ghv(h+b). \end{aligned} \tag{2.49}$$

The entropy potential and entropy flux potentials are

$$\Phi = \frac{g}{2}h^2, \qquad \Psi_f = \frac{g}{2}h^2u, \qquad \Psi_g = \frac{g}{2}h^2v. \tag{2.50}$$

The derivatives of the entropy potential and entropy potential flux fulfill the symmetry condition (2.27)*,*

$$\Phi_{\vec{q}} = \vec{w}, \qquad (\Psi_f)_{\vec{q}} = \vec{f}, \qquad (\Psi_g)_{\vec{q}} = \vec{g}. \tag{2.51}$$

Proof. See Appendix A.2. ■

In Lemma 1 and Lemma 2 we have shown that the total energy is a mathematical entropy function for the shallow water equations in one and two dimensions. Assuming smooth solutions, we contract the shallow water equations (2.29), respectively (2.33), with the entropy variables (2.43), respectively (2.48), to find the entropy equality

$$e_t + \nabla\cdot\vec{\mathcal{F}} = 0, \tag{2.52}$$

with $\vec{\mathcal{F}} = \mathcal{F}$ in one dimension and $\vec{\mathcal{F}} = (\mathcal{F}, \mathcal{G})^T$ in two dimensions. In the presence of shocks, the entropy equality is generalized to an entropy inequality,

$$e_t + \nabla \cdot \vec{\mathcal{F}} \leq 0, \tag{2.53}$$

which is to be understood in the sense of distributions. In the development of numerical schemes for the shallow water equations we will enforce the entropy inequality as a criterion to filter out physically irrelevant solutions. In the field of gas dynamics this is equivalent to obtaining a solution that obeys the second law of thermodynamics.

3. Numerical Methods

This chapter will serve as an introduction into numerical methods for hyperbolic conservation laws. We will provide a brief overview of basic numerical schemes that frequently serve as a basis for more sophisticated methods. The first methods that come to mind are finite difference (FD) and finite volume (FV) methods. They differ on a fundamental level in the sense in which the equations are solved and how the solution is approximated. While finite difference methods provide point-wise approximations to the solution of the conservation law in differential form, finite volume methods are based on the integral form of the equations and approximate mean values over certain control volumes. Other types of approximations include polynomials, splines and similar shape functions. These choices obviously affect the properties of the discretization and the resulting numerical methods are more or less suitable for a problem dependent on, among others, the type of physics studied and the computational processing power available to the user.

We continue by briefly discussing the basics of each type of scheme and their advantages and disadvantages before we argue why the nodal discontinuous Galerkin method will serve as the basis for the numerical method developed in this work.

3.1. An overview

3.1.1. Finite Difference Methods

Finite difference (FD) methods are among the first methods to have been developed for solving PDEs. They have been extensively researched and are widely applied to this date [99]. The basic idea relies on approximating derivatives in space and time with finite difference quotients. The solution is then obtained at the individual grid points. The simplest cases are the first order approximations in space and time

$$\begin{aligned}\frac{\partial w}{\partial x} &= \frac{w(x+\Delta x,t)-w(x,t)}{\Delta x} + \mathcal{O}(\Delta x),\\ \frac{\partial w}{\partial t} &= \frac{w(x,t+\Delta x)-w(x,t)}{\Delta t} + \mathcal{O}(\Delta t),\end{aligned} \tag{3.1}$$

and the second order central difference in space

$$\frac{\partial w}{\partial x} = \frac{w(x+h,t)-w(x-\Delta x,t)}{2\Delta x} + \mathcal{O}(\Delta x^2), \tag{3.2}$$

with spatial step size Δx between two grid points and temporal step size Δt.

Considering a conservation law

$$w_t + f(w)_x = 0, \tag{3.3}$$

on the domain $\Omega = [a, b] \times [0, T]$, the first step towards a simple explicit finite difference scheme is to divide the spatial and temporal domain into grid points

$$\begin{aligned} a = x_0 < x_1 < \ldots < x_N < x_{N+1} = b, \\ 0 = t^0 < t^1 < \ldots < t^{N+1} = T, \end{aligned} \tag{3.4}$$

and denoting the approximation of w at node x_i and time level t^n by w_i^n. Then, the derivatives are approximated at each spatial node $i = 1, \ldots N$ and time step n by one of the formulas given in (3.1) and (3.2). For example, a simple forward-time central-space scheme is given by

$$\frac{w_i^{n+1} - w_i^n}{\Delta t} + \frac{f_{i+1}^{n+1} - f_{i-1}^n}{2h}, \qquad i = 1, \ldots, N, \tag{3.5}$$

with suitable boundary and initial conditions. The scheme (3.5) is, however, unstable, even for linear PDEs [151, 155, 224]. An unconditionally stable scheme can be obtained by considering an implicit approximation

$$\frac{w_i^{n+1} - w_i^n}{\Delta t} + \frac{f_{i+1}^{n+1} - f_{i-1}^{n+1}}{2h}. \tag{3.6}$$

The trade-off here is stability for computational complexity, since a (possibly large) system of linear equations must be solved in each time step. High-order finite difference approximations can be manufactured relatively simple from the Taylor series expansion. Another advantage is the relatively low computational intensity which can lead to cost-effective schemes. A downside of finite difference approximations is the geometric flexibility. It is generally difficult to develop good finite difference approximations on complex geometries.

Additionally, approximating the derivatives based on the Taylor expansion is problematic at shocks, where the classical derivatives are no longer valid. To overcome this issue, researchers have found ways to reduce the effect of discontinuities by selecting the stencil for the approximation based on smoothness measures. Selecting the smoothest stencil leads to finite difference essentially non-oscillatory (ENO) schemes. Considering a convex combination of several stencils weighed by their smoothness measures leads to weighted essentially non-oscillatory (WENO) schemes [119, 217].

3.1.2. Finite Volume Methods

Contrary to the finite difference schemes, finite volume methods are based on the integral form of the equations. The domain is divided into cells $[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] \times [t^n, t^{n+1}]$ with cell interfaces located at $x_{j\pm\frac{1}{2}} = x_j \pm \frac{1}{2}\Delta x$, where Δx denotes the size of the control volumes. Considering the conservation law (3.3) again, the integral form defined on the cell with center x_j is given by

$$\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \Big[w(x, t^{n+1}) - w(x, t^n)\Big]\, dx = \int_{t_n}^{t_{n+1}} \Big[f(w(x_{j+\frac{1}{2}}, t^n)) - f(w(x_{j-\frac{1}{2}}, t^n))\Big]\, dt. \tag{3.7}$$

The idea is then to not approximate point values as in the FD methods but mean values for certain control volumes. The mean value for cell j at time level t^n is defined by

$$\overline{w}_j^n = \frac{1}{\Delta x} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} w(x, t^n) dx. \tag{3.8}$$

We define the interface fluxes by

$$F_{j+\frac{1}{2}}^n := \frac{1}{\Delta t} \int_{t^n}^{t_{n+1}} f(w(x_{j+\frac{1}{2}}, t)) dt, \tag{3.9}$$

and use Euler forward in time, to find the FV cell update scheme

$$\overline{w}_j^{n+1} = \overline{w}_j^n - \frac{\Delta t}{\Delta x} \left[F_{j+\frac{1}{2}}^n - F_{j-\frac{1}{2}}^n \right]. \tag{3.10}$$

The approximation of cell averages leads to discontinuities at the cell interfaces. Thus, the interface fluxes given in (3.9) are not uniquely defined. This difficulty is overcome by interpreting the evaluation of the interface fluxes as a Riemann problem, as depicted in Figure 3.1. A vast variety of approximate or exact Riemann solvers exist and we refer

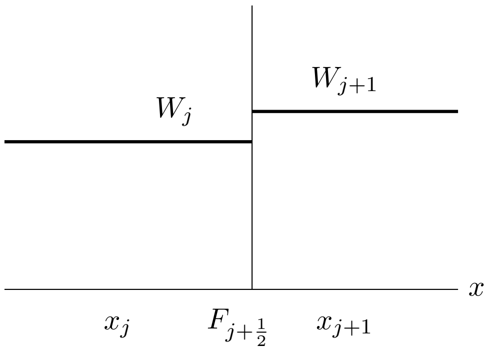


Figure 3.1.: Discontinuous solution at the interface between Left and right states viewed at the cell interface between the two control volumes centered at x_j and x_{j+1}.

to Toro [240] for an extensive discussion. In practice, the Riemann solver is typically chosen based on the physical problem. For instance, there are more dissipative Riemann solvers that increase robustness but possibly smear out important details if they are applied carelessly. A very popular choice for the Riemann solver is the Lax-Friedrichs numerical flux [213]

$$F^* \left(w^+, w^- \right) = \frac{1}{2} \left(f(w^+) + f(w^-) \right) - \frac{\alpha}{2} \left(w^+ - w^- \right), \tag{3.11}$$

where $\alpha = |f'(w)|$ is the maximum wave speed.

The scheme (3.10) is by construction conservative, ensuring correct shock speeds. However, it is only first order accurate. High-order FV methods are obtained by reconstructing a polynomial across a stencil of multiple cells. This polynomial is then evaluated to obtain high-order approximations of the interface values $w_{j\pm\frac{1}{2}}$. Further improvements have been made to construct more robust high-order schemes by examining multiple candidate stencils and selecting (ENO) or weighing (WENO) the stencil contributions based on smoothness indicators [102, 160, 219, 220]. So, while it is possible to develop high-order finite volume schemes (and widely used in practice), the necessary steps are quite involved. Especially WENO schemes of third and fifth order are commonly used, since explicit formulas for the calculation of the smoothness indicators are available in the literature, e.g. [218]. High-order FV methods that require large stencils quickly become computationally expensive. Complex geometries further intensify this problem. Large stencils lead to poor parallelizability as cell coupling and the amount of information exchanges increases.

3.1.3. Finite Element Methods

Another broad class of numerical methods are Finite Element Methods (FEM). In the FEM framework, the domain is subdivided into many smaller elements of varying shapes. The most frequently used are triangles and quadrilaterals with possibly curved sides. The element-local approximation is usually based on a variational formulation of the physical equations. In a variational formulation the requirements on the solution are lowered and weak solutions with respect to specific test functions are considered. A common choice for the local approximations are polynomials. This has the advantage that high-order approximations are achievable by simply increasing the polynomial degree. The element-local approximations are then coupled across interfaces either continuously or discontinuously [137]. A continuous approximation across interfaces leads to a global coupling and a large set of algebraic equations, while the choice of discontinuous interfaces leads to the issue of non-unique interface states at the element boundaries, similar to finite volume methods.

This general framework allows for a wide variety of methods each with its own advantages and disadvantages. However, two common advantages for most finite element methods are the simplicity of obtaining high-order approximations as well as the capability of handling curved geometries by constructing the mesh accordingly.

Discontinuous interfaces lead to a minimal coupling between the elements and, thus, allow for high parallelizability. Keeping the computational efficiency in mind, it might be advantageous to follow the collocation principle and pick the test functions and basis functions identically. Along with a nodal polynomial approximation we arrive at the Discontinuous Galerkin Spectral Element Method (DGSEM). The DGSEM serves as a building block in the development of the numerical method presented in this work. It combines the mesh-flexibility and high-order capabilities of Finite Element methods with the high parallelizability of low order Finite Volume methods as elements are only coupled through their interface communications with direct neighbours. It is computationally very efficient as many terms cancel due to the collocation principle and the simplifications they bring to the numerical quadratures. We explain the DGSEM in one and two dimensions in the following section.

3.2. Discontinuous Galerkin Spectral Element Method

In this section we introduce the specific type of discontinuous Galerkin method that we use as the basis for the entropy stable numerical scheme for the shallow water equations. DG schemes have been originally introduced by Reed and Hill, [209], and have been continuously researched and improved upon. In the literature, vast discussions of these type of methods can be found [137, 106, 123, 45, 107]. For our purposes we focus on a nodal DG method with spectral elements, the DGSEM. We will introduce the scheme for one dimensional scalar conservation laws first and subsequently extend it to systems of balance laws on curved quadrilateral grids.

3.2.1. DGSEM in one dimension

We consider a scalar one dimensional conservation law on the physical domain $\Omega \subset \mathbb{R}$

$$w_t + f(w)_x = 0, \tag{3.12}$$

equipped with suitable boundary and initial conditions. In line with the finite element approach, we divide the domain Ω into non-overlapping elements E_i, $i = 1, \dots, K$, such that $\cup_{i=1}^{K} E_i = \Omega$. The global approximation $w_h(x,t)$ can then be expressed as the union of all the local approximations,

$$w(x,t) \approx w_h(x,t) = \bigoplus_{k=1}^{K} w_k(x,t), \tag{3.13}$$

and we can focus our efforts on the element local approximation w_k. Omitting the element index k, we map each element E with center x_0 and element width Δx to reference space $E_0 = [-1,1]$ by the linear coordinate mapping

$$x(\xi) = x_0 + \frac{\Delta x}{2}\xi, \qquad \xi \in [-1,1]. \tag{3.14}$$

The Jacobian of this transformation is given by $\mathcal{J} = x_\xi = \frac{\Delta x}{2}$. We use the chain rule on the flux function

$$f(w)_x = f(w)_\xi \xi_x = \mathcal{J}^{-1} f(w)_\xi, \tag{3.15}$$

to find the conservation law for element E in reference space,

$$\mathcal{J} w_t + f_\xi = 0. \tag{3.16}$$

We want to solve the conservation law in reference space (3.16) in a weak sense. Thus, in a first step we multiply (3.16) by a smooth test function $\phi(\xi)$ and integrate the equation over the reference element to find

$$\int_{E_0} \mathcal{J} w_t \phi \, d\xi + \int_{E_0} f_\xi \phi \, d\xi = 0. \tag{3.17}$$

We use integration by parts on the flux integral to move the derivative from the flux onto the test function

$$\int_{E_0} f_\xi \phi \, d\xi = [f\phi]_{-1}^{+1} - \int_{E_0} f\phi_\xi \, d\xi. \tag{3.18}$$

Inserting (3.18) into (3.17), we find the weak formulation of (3.16)

$$\int_{E_0} \mathcal{J} w_t \phi \, d\xi + [f\phi]_{-1}^{+1} - \int_{E_0} f\phi_\xi \, d\xi = 0. \tag{3.19}$$

Since we approximate the solution locally on each element, the interface state is not uniquely defined. However, if the flux on the interface in (3.18) is not the same for both adjacent elements, the scheme is not globally conservative. This issue is addressed by interpreting the interface situation as a (generalized) Riemann problem as shown in Figure 3.2. We note, that we have seen this approach before in the context of finite volume methods in Section 3.1.2. The Riemann problem can be solved either analytically

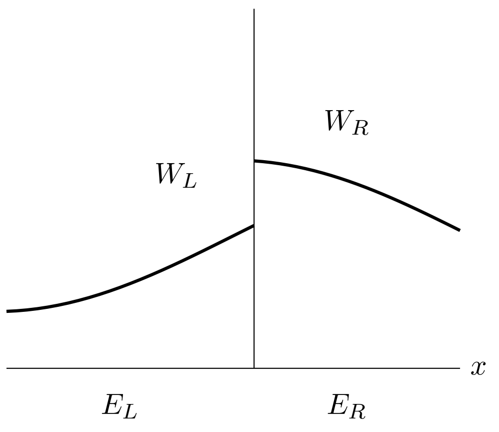


Figure 3.2.: Discontinuous solution at the interface between Left and right states viewed at the shared interface of elements E_L and E_R.

or approximately. For a thorough discussion on Riemann problems as well as exact and approximate Riemann solvers we refer to the literature, specifically the book by Toro [240]. For the DGSEM we choose to solve the Riemann problem approximately due to the cost efficiency of this approach. Thus, we approximate the interface flux in (3.18) for both adjacent elements by the same numerical Riemann solver. Typical choices of Riemann solvers include the Lax-Friedrichs or Rusanov numerical flux (3.11) [213], the Roe flux [210] or the Harten-Lax-Leer (HLL) flux [105]. This treatment of the discontinuity between elements leads to a locally and globally conservative scheme [107].

Replacing the surface flux in (3.19) by a numerical Riemann solver F^*, we obtain the basis for the weak form discontinuous Galerkin approximation

$$\int\limits_{E_0} \mathcal{J}\, w_t \, \phi \, d\xi + [F^* \phi]_{-1}^{+1} - \int\limits_{E} f\phi_\xi d\xi = 0. \tag{3.20}$$

Assuming that the volume integral in (3.20) is only defined by interior information, we use integration by parts on the flux integral once more to find

$$\int\limits_{E} f\phi_\xi d\xi = [f\phi]_{-1}^{+1} - \int\limits_{E} f_\xi \phi \, d\xi. \tag{3.21}$$

Inserting into (3.20) yields the basis for strong form DG approximations

$$\int\limits_{E_0} \mathcal{J}\, w_t \,\phi d\xi + [(F^* - f)\phi]_{-1}^{+1} + \int\limits_{E_0} f_\xi \phi \, d\xi = 0. \tag{3.22}$$

Thus far, we have not addressed the approximation of the element local quantities. For the DGSEM, we approximate quantities of interest such as the conservative variable w and the flux f by polynomials of degree N. Generally, we denote the polynomial approximation by capital letters such as W for the approximation of w. Alternatively, we also indicate a nodal approximation of a quantity by the indexing. For example, in the context of the shallow water equations h_i refers to the approximated value of quantity h at node i, where $i \in [0, \dots, N]$.

Since interpolating polynomials are unique, we can generally choose either a nodal or a modal representation. While both representations are mathematically equivalent, numerically there are advantages and disadvantages for either one. The nodal representation allows the use of collocated quadrature nodes. This reduces the computational complexity of the approximation of the integrals in the variational form. Therefore, we choose a nodal polynomial representation based on Legendre-Gauss-Lobatto (LGL) points $\{\xi_i\}_{i=0}^N$ in the reference element. The associated quadrature rule is exact for polynomials of degree $2N-1$. Also, the LGL nodes include the end points in the approximation and the associated derivative operators have the summation-by-parts property, which we will discuss in more detail in Chapter 4. We choose Lagrange basis functions for the interpolant, which are defined by

$$\ell_j(\xi) = \prod_{i=0, i\neq j}^{N} \frac{\xi - \xi_i}{\xi_j - \xi_i}, \qquad j = 0, \dots, N. \tag{3.23}$$

and satisfy the cardinal property

$$\ell_j(\xi_i) = \delta_{ij}, \qquad i,j = 0, \dots, N, \tag{3.24}$$

where δ_{ij} denotes Kronecker's symbol with $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ for $i \neq j$. The element-wise polynomial approximation (e.g for W in element E) is then written as

$$w(x,t)\big|_E = w(x(\xi),t) \approx W(\xi,t) := \sum_{i=0}^{N} W_i(t)\, \ell_i(\xi), \tag{3.25}$$

where $\{W_i(t)\}_{i=0}^N$ are the time dependent nodal degrees of freedom. For ease of notation, we neglect the time dependency of approximations and variables from now on, unless necessary.

Following the collocation principle, derived quantities are approximated in the same way. For instance, for the shallow water equations, the velocity u is approximated by a polynomial of degree N as well and its nodal values are directly computed from the coefficients of the approximations of momentum and water height:

$$u_i := \frac{(hu)_i}{h_i}, \quad i = 0, \dots, N. \tag{3.26}$$

Spatial derivatives are approximated element-wise directly from the derivative of the polynomial approximation, e.g.,

$$\frac{\partial}{\partial \xi} W(\xi) = \sum_{i=0}^{N} W_i \frac{\partial}{\partial \xi} \ell_i(\xi) \,. \tag{3.27}$$

We introduce the polynomial derivative matrix $\mathbf{D}$ with entries

$$D_{ij} := \frac{\partial}{\partial \xi} \ell_j(\xi_i), \qquad i, j = 0, \dots, N, \tag{3.28}$$

which is used to calculate the discrete derivative with respect to ξ. Thus, we can directly use the derivative operator (3.28) to find the derivatives at the interpolation nodes by

$$(W_\xi)_i := \frac{\partial}{\partial \xi} W(\xi_i) = \sum_{l=0}^{N} D_{il}\, W_l \tag{3.29}$$

where $i = 0, \dots, N$.

Integrals are approximated numerically by the LGL quadrature with (lumped) mass matrix $\mathbf{M} = \operatorname{diag}(\omega_0, \omega_1, \dots, \omega_N)$ with LGL quadrature weights, $\{\omega_j\}_{j=0}^{N}$, on the diagonal. The corresponding one dimensional quadrature of degree N is given by

$$\int\limits_{-1}^{1} w(\xi)\, d\xi \approx \sum_{i=0}^{N} w(\xi_i)\omega_i =: \int\limits_{E,N} w(\xi) dE. \tag{3.30}$$

We choose the test functions ϕ in (3.20)-(3.22) to be polynomials in the reference element E as well, represented by

$$\phi^E(\xi) = \sum_{i=0}^{N} \phi_i^E \ell_i(\xi), \tag{3.31}$$

with arbitrary coefficients ϕ_i^E and Lagrange basis functions as in (3.23). As the Jacobian of the transformation, $\mathcal{J} = \frac{\Delta x}{2}$, is constant, its polynomial approximation denoted by J is exact. We insert the polynomial approximation (3.25) of variables and fluxes into the weak (3.20) and strong formulation (3.22) and approximate integrals with the quadrature rule (3.30) to find the discontinuous Galerkin weak formulation

$$\int\limits_{E,N} J\, W_t\, \phi \,\mathrm{dE} + [F^* \phi]_{-1}^{+1} - \int\limits_{E,N} F \frac{\partial}{\partial \xi} \phi \,\mathrm{dE} = 0, \tag{3.32}$$

and the discontinuous Galerkin strong formulation

$$\int\limits_{E,N} J\,W_t\,\phi\,\mathrm{dE} + [F^* - F\phi]_{-1}^{+1} + \int\limits_{E,N} F_\xi \phi\,\mathrm{dE} = 0. \tag{3.33}$$

The representation ϕ^E of the arbitrary test functions ϕ with the same basis functions as the approximating polynomials (3.31) allows us to simplify the integrals in (3.32) and (3.33). We are able to derive a set of pointwise equations for the DGSEM formulation. For the weak form we see

$$\begin{aligned}
&\int\limits_{E,N} J\,W_t \phi^E\,d\xi - \int\limits_{E,N} F\,\phi^E_\xi\,d\xi + \left[F^* \phi^E\right]_{-1}^{+1} \\
=&\int\limits_{E,N} J\,W_t \sum_{i=0}^{N} \phi_i^E \ell_i(\xi)\,d\xi - \int\limits_{E,N} F\left(\sum_{i=0}^{N} \phi_i^E \frac{\partial}{\partial\xi}\ell_i(\xi)\right) d\xi + \left[\sum_{i=0}^{N} \phi_i^E \ell_i(\xi)\,F^*\right]_{-1}^{+1} \\
=&\sum_{i=0}^{N} \phi_i^E \left(\int\limits_{E,N} J\,W_t \ell_i(\xi)\,d\xi - \int\limits_{E,N} F\,\frac{\partial}{\partial\xi}\ell_i(\xi)\,d\xi + [\ell_i(\xi)\,F^*\,]_{-1}^{+1}\right).
\end{aligned} \tag{3.34}$$

The same steps are applied to the strong form formulation (3.33). This yields a set of pointwise equations for each node i that need to be satisfied for the weak form

$$\int\limits_{E,N} J\,W_t\,\ell_i\,d\xi + [\ell_i(\xi)\,F^*\,]_{-1}^{+1} - \int\limits_{E,N} F\,\frac{\partial}{\partial\xi}\,\ell_i\,d\xi = 0, \qquad i = 0,\ldots,N, \tag{3.35}$$

and the strong form

$$\int\limits_{E,N} J\,W_t\,\ell_i\,d\xi + [\ell_i(\xi)\ (F^* - F)\,]_{-1}^{+1} + \int\limits_{E,N} F_\xi\,\ell_i\,d\xi = 0, \qquad i = 0,\ldots,N. \tag{3.36}$$

Inserting the definitions of the polynomial approximations and the LGL quadrature rule allows us to simplify further. We start with the time derivative and see

$$\begin{aligned}
\int\limits_{E,N} JW_t\ell_i(\xi)d\xi &= \sum_{n=0}^{N} \ell_i(\xi_n)\left(\sum_{k=0}^{N}(W_k)_t\ell_k\,(\xi_n)\right)\omega_n J \\
&= \omega_i J (W_t)_i.
\end{aligned} \tag{3.37}$$

Similarly, for the weak form volume integral we find

$$\begin{aligned}
-\int\limits_{E,N} F\frac{\partial}{\partial\xi}\ell_i d\xi &= -\sum_{n=0}^{N}\left(\sum_{k=0}^{N} F_k\ell_k\,(\xi_n)\right)\frac{\partial}{\partial\xi}\ell_i\,(\xi_n)\,\omega_n \\
&= -\sum_{n=0}^{N} F_n\frac{\partial}{\partial\xi}\ell_i\,(\xi_n)\,\omega_n = -\sum_{n=0}^{N} F_n D_{ni}\omega_n \\
&= \sum_{n=0}^{N} F_n\hat{D}_{in}\omega_i,
\end{aligned} \tag{3.38}$$

with $\hat{D}_{ij} := -D_{ji}\frac{\omega_j}{\omega_i}$. The corresponding strong form volume integral is

$$\begin{aligned}\int\limits_{E,N} F_\xi \ell_i d\xi &= \sum_{n=0}^{N}\left(\sum_{k=0}^{N} F_k \frac{\partial}{\partial \xi}\ell_k\left(\xi_n\right)\right)\ell_i\left(\xi_n\right)\omega_n \\ &= \sum_{n=0}^{N}\left(\sum_{k=0}^{N} F_k D_{nk}\right)\ell_i\left(\xi_n\right)\omega_n \\ &= \sum_{k=0}^{N} F_k D_{ik}\omega_i.\end{aligned} \tag{3.39}$$

The surface integral in one dimension simply reduces to

$$\oint\limits_{\partial E,N} \ell_i\, F^*\ \mathrm{dS} = \left[\ell_i(\xi)\, F^*\,\right]_{-1}^{+1} = \delta_{iN}F_N^* - \delta_{i0}F_0^*, \tag{3.40}$$

for the weak form and to

$$\oint\limits_{\partial E,N} \ell_i\, F^*\ \mathrm{dS} = \left[\ell_i(\xi)\ \left(F^* - F\right)\right]_{-1}^{+1} = \delta_{iN}\ \left(F_N^* - F_N\right) - \delta_{i0}\ \left(F_0^* - F_0\right), \tag{3.41}$$

for the strong form. Gathering terms from (3.37), (3.38) and (3.40) and dividing by ω_i, we find

$$J(W_i)_t = -\sum_{m=0}^{N}\hat{D}_{im}F_m - \frac{1}{\omega_i}\left(\delta_{iN}\,F_N^* - \delta_{i0}\,F_0^*\right). \tag{3.42}$$

Similarly, we find the strong form DGSEM from (3.37), (3.39) and (3.41)

$$J(W_i)_t = -\sum_{m=0}^{N} D_{im}F_m - \left(\frac{\delta_{iN}}{\omega_i}\left(F_N^* - F_N\right) - \frac{\delta_{i0}}{\omega_i}\left(F_0^* - F_0\right)\right). \tag{3.43}$$

All the steps, demonstrated above for a scalar conservation law, also hold for a system of balance laws. We write the DGSEM for a system of one dimensional balance laws

$$J\vec{W}_t + \vec{\mathcal{L}}_\xi = \vec{S}, \tag{3.44}$$

with weak form spatial operator

$$\left(\vec{\mathcal{L}}_\xi\right)_i = \frac{1}{\omega_i}\left(\delta_{iN}\left[\vec{F}^*\right]_N - \delta_{i0}\left[\vec{F}^*\right]_0\right) + \sum_{m=0}^{N}\hat{D}_{im}\vec{F}_m, \tag{3.45}$$

or the respective strong form spatial operator

$$\left(\vec{\mathcal{L}}_\xi\right)_i = \frac{1}{\omega_i}\left(\delta_{iN}\left[\vec{F}^* - \vec{F}\right]_N - \delta_{i0}\left[\vec{F}^* - \vec{F}\right]_0\right) + \sum_{m=0}^{N} D_{im}\vec{F}_m. \tag{3.46}$$

As mentioned earlier, the numerical interface flux can be chosen in various ways. However, when we refer to the standard DGSEM in this work, we mean the variant where the Lax-Friedrichs flux (3.11) is used on the interfaces. A possible source term is dependent on the equations and must be discretized appropriately. Typically, the same polynomial approximation is used for the source term as well. For instance, in the shallow water equations the bottom topography $b(x)$ is also approximated by a polynomial of degree N with the same LGL interpolation nodes.

3.2.2. DGSEM in two dimensions

In this section we extend the ideas of the one dimensional DGSEM to two spatial dimensions. To capture more general physical domains, we allow the mesh to be unstructured and feature curvilinear sides for its elements. We require the mesh elements to be quadrilaterals to allow for a simple tensor product extension of the discontinuous Galerkin method. We highlight the details of this approach by first discussing a scalar two dimensional conservation law, written as

$$w_t + \nabla \cdot (f, g) = 0. \tag{3.47}$$

Adopting the same notation as in the one dimensional case, we divide the physical domain Ω into non-overlapping quadrilateral elements E_i, $i = 1, \dots, K$, such that $\cup_{i=1}^{K} E_i = \Omega$. Dropping the index, each element E is then mapped to the two dimensional reference element $E_0 = [-1,1]^2$. A commonly used transformation between the reference square and an arbitrary curve-sided quadrilateral element is the transfinite interpolation with linear blending [137]. The mapping between the coordinates of the reference square (ξ, η) and the physical coordinates $\vec{x}(\xi,\eta) = (x(\xi,\eta), y(\xi,\eta))$ is given by

$$\begin{aligned} \vec{x}(\xi,\eta) = {} & \frac{1}{2}\left[(1-\xi)\vec{\Gamma}_4(\eta) + (1+\xi)\vec{\Gamma}_2(\eta) + (1-\eta)\vec{\Gamma}_1(\xi) + (1+\eta)\vec{\Gamma}_3(\xi)\right] \\ & - \frac{1}{4}\left[(1-\xi)\{(1-\eta)\vec{\Gamma}_1(-1) + (1+\eta)\vec{\Gamma}_3(-1)\}\right. \\ & \left. + (1+\xi)\{(1-\eta)\vec{\Gamma}_1(1) + (1+\eta)\vec{\Gamma}_3(1)\}\right], \end{aligned} \tag{3.48}$$

where we assume that each convex element is bounded by four curves $\vec{\Gamma}_j$, $j = 1, 2, 3, 4$. We show the mapping in Figure 3.3.

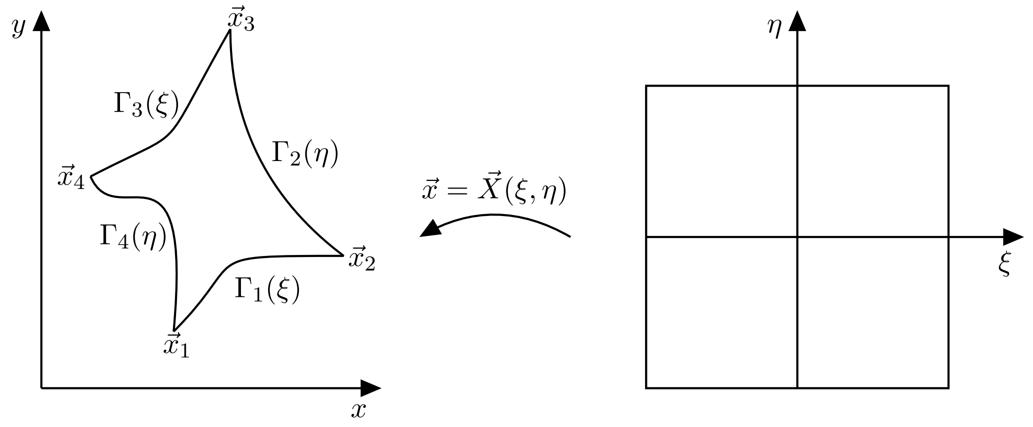


Figure 3.3.: Transformation of the reference square E_0 to a curved quadrilateral E.

The Jacobian of the transformation is computed by

$$\mathcal{J} = x_\xi \, y_\eta - x_\eta \, y_\xi. \tag{3.49}$$

The gradients in physical space $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)^T$ and reference space $\hat{\nabla} = \left(\frac{\partial}{\partial \xi}, \frac{\partial}{\partial \eta}\right)^T$ are related by the chain rule

$$\nabla = \frac{1}{\mathcal{J}} \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix} \hat{\nabla}. \tag{3.50}$$

If the mapping is sufficiently smooth, we can use (3.50) to replace the physical x and y derivatives in (3.47). We then multiply by the Jacobian $\mathcal{J}$ to obtain the conservation law in reference space

$$\mathcal{J} w_t + \hat{\nabla} \cdot (\tilde{f}, \tilde{g}) = 0, \tag{3.51}$$

where we introduce the contravariant fluxes defined by

$$\begin{aligned} \tilde{f}(w) &= y_\eta \, f(w) - x_\eta \, g(w), \\ \tilde{g}(w) &= -y_\xi \, f(w) + x_\xi \, g(w). \end{aligned} \tag{3.52}$$

To find the weak form of the two dimensional conservation law in reference space, (3.51), we repeat the steps from Section 3.2.1. First, we multiply by an arbitrary smooth test function ϕ and integrate over the reference element. Then, we use integration by parts to move the differentiation of the fluxes, $\hat{\nabla} \cdot (\tilde{f}, \tilde{g})$, onto the test function. Finally, we replace the fluxes across element interfaces by numerical interface fluxes $\tilde{F}^*$ and $\tilde{G}^*$ and obtain the weak form

$$\int_E \mathcal{J} \, w_t \, \phi \, \mathrm{dE} + \oint_{\partial E} \phi \, \left(\tilde{F}^*, \tilde{G}^*\right) \cdot \hat{n} \, \mathrm{dS} - \int_E \left(\tilde{f}, \tilde{g}\right) \cdot \hat{\nabla}\phi \, \mathrm{dE} = 0. \tag{3.53}$$

Integrating by parts once more yields the strong form

$$\int_E \mathcal{J} \, w_t \, \phi \, \mathrm{dE} + \oint_{\partial E} \phi \, \left(\tilde{F}^* - \tilde{f}, \tilde{G}^* - \tilde{g}\right) \cdot \hat{n} \, \mathrm{dS} + \int_E \hat{\nabla} \cdot \left(\tilde{f}, \tilde{g}\right) \phi \, \mathrm{dE} = 0. \tag{3.54}$$

We note that the normals $\hat{n}$ in the surface integrals in (3.53) and (3.54) are the normal vectors in reference space. They are either $\hat{n} = (\pm 1, 0)^T$ at the $\xi = \text{const}$ interfaces or $\hat{n} = (0, \pm 1)^T$ in case $\eta = \text{const}$.

We keep the notation similar to the one dimensional case, denoting polynomial approximations by capital letters. Also, indexed quantities such as h_{ij} in the context of the shallow water equations refer to the approximated value of quantity h at node i, j, where $i, j \in [0, \dots, N]$. We stick to the nodal form of the interpolation with nodes defined at the Legendre-Gauss-Lobatto (LGL) points $\{\xi_i\}_{i=0}^N$ and $\{\eta_j\}_{j=0}^N$ in the reference square. The two dimensional element-wise polynomial approximation (e.g for W in element E) is defined by

$$w(x, y, t)\big|_E = w(x(\xi, \eta), y(\xi, \eta), t) \approx W(\xi, \eta, t) := \sum_{i=0}^{N} \sum_{j=0}^{N} W_{ij}(t)\, \ell_i(\xi)\, \ell_j(\eta), \tag{3.55}$$

where $\{W_{ij}(t)\}_{i,j=0}^{N,N}$ are the time dependent nodal degrees of freedom and ℓ_i, ℓ_j are the one-dimensional Lagrange basis functions defined in (3.23). We neglect the time dependency for now in favor of a simpler notation.

We apply the typical collocation strategy also to the contravariant fluxes, where we interpolate the metric terms at the same nodes, e.g.

$$\tilde{F}_{ij} = y_\eta(\xi_i, \eta_j)\, g(W_{ij}) - x_\eta(\xi_i, \eta_j)\, g(W_{ij}), \qquad i, j = 0, \dots, N. \tag{3.56}$$

Similarly, all the metric terms are approximated by this strategy. For instance, the Jacobian is approximated by the polynomial of degree N with nodal values

$$J_{ij} := \mathcal{J}(\xi_i, \eta_j). \tag{3.57}$$

Due to the tensor product ansatz in ξ and η direction, we can simply use the one dimensional derivative operator (3.28) in each spatial direction to approximate the derivatives. Thus, the partial derivatives in ξ and η direction are computed by

$$(W_\xi)_{ij} = \sum_{l=0}^{N} D_{il}\, W_{lj} \qquad \text{and} \qquad (W_\eta)_{ij} = \sum_{l=0}^{N} D_{jl} W_{il}, \tag{3.58}$$

where $i, j = 0, \dots, N$. The respective two-dimensional quadrature is

$$\int_{-1}^{1} \int_{-1}^{1} w(\xi, \eta)\, \mathrm{d}\xi \mathrm{d}\eta \approx \sum_{i=0}^{N} \sum_{j=0}^{N} w(\xi_i, \eta_j) \omega_i \omega_j =: \int_{E,N} w(\xi, \eta) dE, \tag{3.59}$$

with the LGL quadrature weights $\{\omega_j\}_{j=0}^N$.

We represent the test functions ϕ in (3.53)-(3.54) as polynomials in the reference element E with the same basis functions

$$\phi := \phi^E = \sum_{i=0}^{N} \sum_{j=0}^{N} \phi_{ij}^E \ell_i(\xi) \ell_j(\eta), \tag{3.60}$$

and arbitrary coefficients ϕ_{ij}^E for $i, j = 0, \ldots, N$. This choice of test functions and the collocation of the interpolation and quadrature nodes enable us to once again simplify the integrals analogously to the one dimensional case (3.34).

We demand that the discretisation preserves free-stream solutions, i.e., constant solutions of the balance law (3.51) should remain constant for all times. A sufficient condition for constant state preservation is that the metric identities

$$\frac{\partial}{\partial \xi}\mathcal{J}\vec{a}^{\,1} + \frac{\partial}{\partial \eta}\mathcal{J}\vec{a}^{\,2} = \vec{0}, \tag{3.61}$$

are satisfied discretely [137]. The volume weighted contravariant basis vectors, $\mathcal{J}\vec{a}^{\,1}$ and $\mathcal{J}\vec{a}^{\,2}$, are given by

$$\mathcal{J}\vec{a}^{\,1} = \left(y_\eta, -x_\eta\right)^T, \qquad \mathcal{J}\vec{a}^{\,2} = \left(-y_\xi, x_\xi\right)^T. \tag{3.62}$$

The metric identities can be expressed with the discrete derivative operators (3.58) by

$$\begin{aligned} \sum_{l=0}^{N} D_{il}\,(y_\eta)_{lj} - \sum_{l=0}^{N} D_{jl}(y_\xi)_{il} &= 0, \\ -\sum_{l=0}^{N} D_{il}\,(x_\eta)_{lj} + \sum_{l=0}^{N} D_{jl}(x_\xi)_{il} &= 0, \end{aligned} \tag{3.63}$$

and are not trivially satisfied for any mesh. A sufficient condition to ensure that the metric identities hold is if differentiation and interpolation commute [137]. Kopriva showed, that free-stream preservation is guaranteed for the DGSEM, if the linear blending formula (3.48) is used and the boundary curves of the quadrilateral elements are approximated by polynomials with an order less than or equal to the polynomial order of the approximate solution N [135].

To fulfill these requirements, we ensure that each boundary curve is approximated by a polynomial of at most order N. We use the Lagrange basis functions given in (3.23) to approximate the boundary curves

$$\vec{\Gamma}_k = \sum_{j=0}^{N} \vec{\Gamma}_k(\zeta_j)\ell_j(\zeta), \qquad \zeta \in [-1, 1], \qquad k = 1, \ldots, 4, \tag{3.64}$$

where the interpolation nodes $\{\zeta_j\}_{j=0}^N$ are typically chosen to be the Chebyshev-Gauss-Lobatto or Legendre-Gauss-Lobatto nodes as these show robust interpolation properties [137]. The polynomial boundary curve approximations (3.64) are used to construct the mapping (3.48) for each element. As the mapping is a polynomial in ξ and η, the derivatives necessary to obtain the metric terms and the normal vectors are computed directly in the discrete derivative sense (3.58). An explicit formula for the free-stream preserving computation of the metric terms is given in [135] for three dimensional meshes:

$$J\vec{a}_n^{\,i} = -\hat{x}_i \cdot \hat{\nabla} \times \left(x_l \hat{\nabla} x_m\right), \qquad i = 1, 2, 3, \quad n = 1, 2, 3, \quad (n, m, l) \text{ cyclic}, \tag{3.65}$$

where $\hat{x}_i$, for $i = 1, 2, 3$, are the unit vectors in three dimensional reference space. For meshes in two dimensions this formula simplifies to a direct discretization of the metric terms from (3.62). Further details of the isoparametric polynomial approximation of boundary curves can be found in [137].

Replacing the integrals in (3.53) and (3.54) with the quadrature rule (3.59) and inserting the polynomial approximations (3.55) for variables, fluxes and test function leads to the pointwise DGSEM in integral form

$$\begin{aligned}\int\limits_{E,N} J\, W_t\, \ell_i(\xi)\ell_j(\eta)\,\mathrm{dE} - \int\limits_{E,N} \left(\tilde{F}, \tilde{G}\right) \cdot \hat{\nabla}\ell_i(\xi)\ell_j(\eta)\,\mathrm{dE} \\ + \oint\limits_{\partial E,N} \ell_i(\xi)\ell_j(\eta)\, \left(\tilde{F}^*, \tilde{G}^*\right)\, \cdot\, \hat{n}\,\mathrm{dS} = 0,\end{aligned} \tag{3.66}$$

and the strong form

$$\begin{aligned}\int\limits_{E,N} \mathcal{J}\, W_t\, \tilde{\phi}_{ij}\,\mathrm{dE} + \int\limits_{E,N} \hat{\nabla} \cdot \left(\tilde{F}, \tilde{G}\right) \ell_i(\xi)\ell_j(\eta)\,\mathrm{dE} \\ + \oint\limits_{\partial E,N} \ell_i(\xi)\ell_j(\eta)\, \left(\tilde{F}^* - \tilde{F}, \tilde{G}^* - \tilde{F}\right)\, \cdot\, \hat{n}\,\mathrm{dS} = 0.\end{aligned} \tag{3.67}$$

The numerical interface fluxes $\tilde{F}^*$ and $\tilde{G}^*$ are defined in the outward pointing normal direction and couple neighbouring elements. They implicitly depend on the “outer” and “inner” values $\vec{W}^+, \vec{W}^-$ along the normal vector in reference space $\hat{n}$. The (discrete) boundary integrals in (3.66) and (3.67) describe the quadrature along the four edges of the element E.

As witnessed in the one dimensional case, we can simplify the integrals due to the collocation of interpolation and quadrature nodes. We will illustrate this for the integral over the time derivative as well as for each volume integral and the surface integral. For the former, the integral can be reduced to a single term,

$$\begin{aligned}&\int\limits_{E,N} JW_t\ell_i(\xi)\ell_j(\eta)\,\mathrm{dE} \\ &\qquad = \sum_{n,m=0}^{N} \ell_i(\xi_n)\ell_j(\eta_m) \left(\sum_{k,l=0}^{N} (W_t)_{kl}\, \ell_k\left(\xi_n\right) \ell_l\left(\eta_m\right)\right) \omega_n\omega_m J_{nm} \\ &\qquad = \omega_i\omega_j J_{ij}(W_t)_{ij},\end{aligned} \tag{3.68}$$

for $i, j = 0, 1, 2, \ldots, N$. For the volume integrals of the weak and the strong form, we are able to reduce the integrals to one-sum expressions, which remain necessary to evaluate the derivatives. To standardize the expressions we introduce the weak form derivative operator $\hat{\mathbf{D}}$, which is found from the standard derivative operator $\mathbf{D}$ by setting

$$\hat{D}_{ij} := -D_{ji}\frac{\omega_j}{\omega_i}. \tag{3.69}$$

With this substitution we make sure that each term is scaled by the weights $\omega_i\omega_j$, which are then divided off. The simplified expression of the weak form volume integral is then

$$\begin{aligned}
&\int\limits_{E,N} \left(\tilde{F},\tilde{G}\right)\cdot\hat{\nabla}(\ell_i(\xi)\ell_j(\eta))\,\mathrm{dE}\\
&\qquad\qquad =\int\limits_{E,N} \tilde{F}\left(\frac{\partial}{\partial\xi}\ell_i(\xi)\ell_j(\eta)\right)+\tilde{G}\left(\ell_i(\xi)\frac{\partial}{\partial\eta}\ell_j(\eta)\right)\,\mathrm{dE}\\
&\qquad\qquad =-\sum_{m=0}^{N} D_{mi}\tilde{F}_{mj}\omega_m\omega_j-\sum_{n=0}^{N} D_{nj}\tilde{G}_{in}\omega_i\omega_n\\
&\qquad\qquad =\sum_{m=0}^{N}\hat{D}_{im}\tilde{F}_{mj}\omega_i\omega_j+\sum_{n=0}^{N}\hat{D}_{jn}\tilde{G}_{in}\omega_i\omega_j.
\end{aligned}\tag{3.70}$$

Similar steps lead to the simplified expression for the strong form DGSEM volume integral:

$$\int\limits_{E,N}\hat{\nabla}\cdot\left(\tilde{F},\tilde{G}\right)\ell_i(\xi)\ell_j(\eta)\,\mathrm{dE}=\sum_{m=0}^{N} D_{im}\tilde{F}_{mj}\omega_m\omega_j+\sum_{n=0}^{N} D_{jn}\tilde{G}_{in}\omega_i\omega_n.\tag{3.71}$$

We note that this representation of the volume integrals shows that to switch between weak and strong form volume integrals, one simply has to exchange the derivative operator and use an identical algorithm otherwise.

The other difference between weak and strong form DGSEM lies in the surface terms. For the strong form, the surface terms involve fluxes based on interior values. Thus, the surface integrals for weak and strong form are slightly different. For the weak form surface integral we find

$$\begin{aligned}
&\oint\limits_{\partial E,N}\ell_i(\xi)\ell_j(\eta)\left(\tilde{F}^*,\tilde{G}^*\right)\cdot\vec{n}\,dS\\
&\quad=-\int_{-1}^{1}\ell_i(-1)\ell_j(\eta)\,\tilde{F}^*(-1,\eta)\,d\eta+\int_{-1}^{1}\ell_i(1)\ell_j(\eta)\,\tilde{F}^*(1,\eta)\,d\eta\\
&\qquad-\int_{-1}^{1}\ell_i(\xi)\ell_j(-1)\,\tilde{G}^*(\xi,-1)\,d\xi+\int_{-1}^{1}\ell_i(\xi)\ell_j(1)\,\tilde{G}^*(\xi,1)\,d\xi\\
&\quad=\delta_{iN}\,\tilde{F}^*_{iN}\omega_j-\delta_{i0}\,\tilde{F}^*_{i0}\omega_j+\delta_{jN}\,\tilde{G}^*_{Nj}\omega_i-\delta_{j0}\,\tilde{G}^*_{0j}\omega_i.
\end{aligned}\tag{3.72}$$

In the strong form surface integral a flux term based on interior data is subtracted from the numerical interface flux at each surface node:

$$\begin{aligned}
&\oint\limits_{\partial E,N}\ell_i(\xi)\ell_j(\eta)\left(\tilde{F}^*,\tilde{G}^*\right)\cdot\vec{n}\,dS\\
&\quad=\delta_{iN}\left(\tilde{F}^*_{iN}-\tilde{F}_{iN}\right)\omega_j-\delta_{i0}\left(\tilde{F}^*_{i0}-\tilde{F}_{i0}\right)\omega_j\\
&\qquad+\delta_{jN}\left(\tilde{G}^*_{Nj}-\tilde{G}_{NJ}\right)\omega_i-\delta_{j0}\left(\tilde{G}^*_{0j}-\tilde{G}_{0j}\right)\omega_i.
\end{aligned}\tag{3.73}$$

We collect the simplified expressions and divide by the quadrature weights $\omega_i\omega_j$ to formulate the pointwise nodal DGSEM approximation. For the weak form DGSEM we gather terms from (3.68), (3.70) and (3.72) to find

$$\begin{aligned} J_{ij}(W_{ij})_t &+ \sum_{m=0}^{N} \hat{D}_{im}\tilde{F}_{mj} + \sum_{n=0}^{N} \hat{D}_{jn}\tilde{G}_{in} \\ &= \frac{1}{\omega_i}\left(\delta_{i0}\,\tilde{F}^*_{i0} - \delta_{iN}\,\tilde{F}^*_{iN}\right) + \frac{1}{\omega_j}\left(\delta_{j0}\,\tilde{G}^*_{0j} - \delta_{jN}\,\tilde{G}^*_{Nj}\right). \end{aligned} \tag{3.74}$$

For the strong form DGSEM, terms from the respective simplified expressions (3.68), (3.71) and (3.73) are combined to obtain

$$\begin{aligned} J_{ij}(W_{ij})_t &+ \sum_{m=0}^{N} D_{im}\tilde{F}_{mj} + \sum_{n=0}^{N} D_{jn}\tilde{G}_{in} \\ &= \frac{1}{\omega_i}\left(\delta_{i0}\,(\tilde{F}^*_{i0} - \tilde{F}_{i0}) - \delta_{iN}\,(\tilde{F}^*_{iN} - \tilde{F}_{iN})\right) \\ &\quad + \frac{1}{\omega_j}\left(\delta_{j0}\,(\tilde{G}^*_{0j} - \tilde{G}_{0j}) - \delta_{jN}\,(\tilde{G}^*_{Nj} - \tilde{G}_{Nj})\right). \end{aligned} \tag{3.75}$$

The DGSEM can be extended to systems of balance laws. In this case, we introduce dimension-split spatial operators to compactly write the weak and strong form DGSEM as

$$J\vec{W}_t + \vec{\mathcal{L}}_\xi + \vec{\mathcal{L}}_\eta = \vec{\tilde{S}} \tag{3.76}$$

with spatial operators

$$\begin{aligned} \left(\vec{\mathcal{L}}_\xi\right)_{ij} &= \frac{1}{\omega_i}\left(\delta_{iN}\vec{\tilde{F}}^*_{Nj} - \delta_{i0}\vec{\tilde{F}}^*_{0j}\right) + \sum_{m=0}^{N} \tilde{\mathbf{D}}_{im}\vec{\tilde{F}}_{mj}, \\ \left(\vec{\mathcal{L}}_\eta\right)_{ij} &= \frac{1}{\omega_j}\left(\delta_{Nj}\vec{\tilde{G}}^*_{iN} - \delta_{0j}\vec{\tilde{G}}^*_{i0}\right) + \sum_{m=0}^{N} \tilde{\mathbf{D}}_{mj}\vec{\tilde{G}}_{im}. \end{aligned} \tag{3.77}$$

In this formulation, the choice of derivative operator determines whether it is in strong or weak form:

$$\tilde{\mathbf{D}} = \begin{cases} \mathbf{D} + \mathbf{S} & \text{if strong form} \\ \hat{\mathbf{D}} & \text{if weak form} \end{cases}, \tag{3.78}$$

where $\mathbf{S} = \operatorname{diag}\left(\frac{1}{w_0}, 0, \dots, 0, -\frac{1}{w_N}\right)$ is the surface matrix.

Here, $\vec{\tilde{S}}$ refers to the discretization of the source term vector in curvilinear coordinates. The exact discretization depends on the underlying physics. For the shallow water equations, the source terms are given by $\vec{s} = -gh(0, b_x, b_y)^T$. If we approximate the bottom topography b by a polynomial at the LGL nodes as well, a direct discretization of the source term is found by

$$
\begin{aligned}
(\tilde{s}_2)_{ij} &= -gh_{ij}\left((y_\eta)_{ij}\sum_{m=0}^{N} D_{im}b_{mj} - (y_\xi)_{ij}\sum_{m=0}^{N} D_{jm}b_{im}\right),\\
(\tilde{s}_3)_{ij} &= -gh_{ij}\left(-(x_\eta)_{ij}\sum_{m=0}^{N} D_{im}b_{mj} + (x_\xi)_{ij}\sum_{m=0}^{N} D_{jm}b_{im}\right).
\end{aligned}
\tag{3.79}
$$

For the *standard* DGSEM described here, we use the curvilinear version of the numerical Lax-Friedrichs flux given by

$$
\begin{aligned}
\vec{\tilde{F}}^{*}\left(\vec{W}^{+},\vec{W}^{-}\right) &= \frac{1}{2}\left(\vec{\tilde{F}}^{+}+\vec{\tilde{F}}^{-}\right) - \frac{\alpha_f}{2}\left(\vec{W}^{+}-\vec{W}^{-}\right),\\
\vec{\tilde{G}}^{*}\left(\vec{W}^{+},\vec{W}^{-}\right) &= \frac{1}{2}\left(\vec{\tilde{G}}^{+}+\vec{\tilde{G}}^{-}\right) - \frac{\alpha_g}{2}\left(\vec{W}^{+}-\vec{W}^{-}\right).
\end{aligned}
\tag{3.80}
$$

where α_f and α_g are the maximum absolute eigenvalue of the associated flux Jacobians. The described discontinuous Galerkin method is high-order accurate (rate of convergence is $N+1$) and geometrically flexible. Although we restrict ourselves to quadrilaterals here, these can have various shapes due to the curved boundaries. The method is highly parallelizable as elements are coupled only through the numerical interrface flux. Also, due to the collocation principle, the computational effort has been drastically reduced. While this standard DGSEM has many desirable properties for us, it is not a stable scheme for systems of non-linear PDEs in the sense that it is not **entropy stable**. We demonstrate this by a numerical example in Section 7.2.

3.3. Numerical principles and entropy stability

While there are many numerical methods, each with its own advantages and disadvantages, there are some fundamental properties that are important regardless of numerical context. First and foremost, the concept of discrete conservation is a staple in the development of numerical methods. Discrete conservation mimics the physical conservative properties of conservation laws in a discrete sense, such that continuously conserved quantities are guaranteed to be conserved by the numerical scheme, as well. While it is often desirable to carry over physical characteristics to the numerical scheme, discrete conservation is **necessary** to guarantee that a numerical scheme yields an actual weak solution to the conservation law. If a numerical method is not conservative, it can generate incorrect shock speeds, which in turn can lead to unphysical solutions. This remarkable result was found by Lax and Wendroff and is famously known as the Lax-Wendroff Theorem [147]. It states, that *if a conservative numerical scheme for a hyperbolic conservation law converges, the solution is a weak solution to the conservation law.* However, it is still unclear, whether the obtained weak solution is physically relevant. As we have seen in the continuous analysis in Chapter 2, weak solutions are not unique and it is a challenging task to choose one specific, physically relevant solution out of the infinite family of weak solutions. An interesting technique to filter out such a solution is to design the numerical method to mimic continuous entropy behaviour. Physically, the entropy is a conserved quantity in all parts of the domain except for shock regions, where entropy is built up (or dissipated mathematically). A numerical scheme that implicitly satisfies a conservation law for the entropy is called **entropy conservative**. If the entropy dissipation at shocks is considered and thus a discrete entropy inequality holds, the scheme is called **entropy stable**.

This approach was first introduced by Lax for scalar conservation laws [146], who proved entropy stability for the Lax-Friedrichs method. Harten, Hyman and Lax generalized this result for a whole sub-class of schemes, the so-called monotone schemes [104]. This result was further extended to E-schemes by Osher [192], which are entropy stable for all entropy pairs and total variation diminishing (TVD). Monotone schemes are defined by their update function. The update function R is an operator which calculates the solution in the next time level $n+1$ based on a stencil of solution values at time n. Considering p solution values to the left and q solution values to the right the update function at node j can be expressed in one dimension as

$$W_j^{n+1} = R\left(W_{j-p-1}^n, \dots, W_j^n, \dots, W_{j+q}^n\right). \tag{3.81}$$

The exact representation of the solution (nodes, cells, elements) and the number of values used for the update is dependent on the numerical method. A numerical scheme is called monotone if the update operator $R(\cdot, \cdot, \dots, \cdot)$ is increasing in all arguments. It is known that all monotone schemes satisfy a local maximum principle and are TVD [107]. Due to this property a solution obtained by a monotone scheme satisfies all entropy conditions for scalar conservation laws [104]. This property also holds for Harten's E-schemes, which are defined by having no-less viscosity than Godunov's method [233]. However, any monotone scheme and any E-scheme is at most first order accurate [105]. This is a severe constriction on the accuracy and efficiency of any monotone method.

High-order methods are advantageous as they offer not only improvements in terms of solution quality but also in terms of computational efficiency [54]. Due to their low dispersion and dissipation errors, such methods are able to resolve even the finer features of a solution such as contact waves [1]. It has also been shown that fewer degrees of freedom are needed to obtain an approximation at a given accuracy [107, 54]. Also, errors in long time simulations may deteriorate, leading to unacceptable solutions by low order schemes [217]. In exchange, the computational work required for each degree of freedom is generally increased and the exact efficiency gains of high-order approximations depend on the numerical method and implementation [107]. A more pressing downside of high-order approximations is encountered in the presence of shocks. While low-order schemes are very robust in their handling, high-order methods suffer under possibly severe oscillations when approximating the solution across a shock. In extreme cases, overshoots in the solution may cause a numerical scheme to simply crash. This is a consequence of producing unphysical values, such as negative densities or pressures in the Euler equations or negative water heights in the shallow water equations. Methods and treatments that deal with these oscillations are referred to as **shock capturing** methods. We present one suitable treatment in Section 5.4 but note that this is a subject of ongoing research and there are many different ways to approach this problem [202, 159].

One approach to not abandon the concept of entropy stability in the context of high-order methods for non-linear systems of conservation laws is to develop entropy stable methods with respect to specific entropy pairs. This idea was first introduced in 1987 by Tadmor for a second order FV type scheme which can also be interpreted as a piecewise linear finite element method [234]. Similar entropy stable second order methods for scalar conservation laws were proposed by Osher [193]. Tadmor's work shows that the development of entropy stable methods is strongly dependent on the construction of entropy stable interface fluxes. As found by Tadmor [234], a numerical flux is entropy conservative if the numerical flux, contracted by the jump in entropy variables, is equal to the jump in the entropy flux potential,

$$\langle\, [\![\vec{q}\,]\!],\, \vec{F}^{*}\,\rangle = [\![\Psi]\!]. \tag{3.82}$$

Based on condition (3.82), Tadmor presents a family of entropy stable schemes, where the evaluation of the entropy conservative flux is based on a path integral in phase space [235]. Entropy stability is obtained by a comparison principle, constructing schemes that are more dissipative than entropy conservative schemes. Additionally, the entropy stability is extended to the time discretization as well, providing fully discrete entropy stable methods, albeit at no more than second order accuracy (at a three point stencil).

LeFloch and Rohde build on Tadmor's foundation to find fully discrete entropy stable schemes of third order accuracy [152]. They also include the possibly non-convex entropy functions of hyperbolic-elliptic systems in their derivations. Furthermore, they prove that fully discrete entropy *conservative* methods do not exist, thus only entropy stable methods should be considered in that case [152, 153]. An entropy stable method for the Navier Stokes equations, where the entropy dissipation is solely based on the viscous and heat flux terms, was proposed by Tadmor and Zhong [236]. They derive an explicit formula for the precise entropy decay rate and construct the numerical fluxes in such a way that this entropy decay rate is obeyed. In case of the Euler equations, the scheme reduces to a purely entropy conserving one, which is problematic for non-smooth solutions. Also, the computation of their entropy stable numerical flux still involves an integration along a path in phase space, which may be expensive to evaluate and detrimental to the computational efficiency [69].

A framework for entropy stable schemes of arbitrary order was introduced by Fjordholm, Mishra and Tadmor in 2012 [69]. The so-called *TeCNO* schemes are based on finite volume methods equipped with entropy conservative numerical fluxes. Then, numerical diffusion operators are added in terms of the entropy variables to obtain entropy stability on a semi-discrete level. The high-order accuracy is achieved by a specific ENO type reconstruction for the cell edge values [69]. Instead of evaluating the numerical fluxes in phase space, they use explicit algebraic solutions of (3.82) for specific physical systems. They also provide an entropy conservative numerical flux for the shallow water equations and the Euler equations, the latter previously proposed by Ismael and Roe [117].

These second order two-point entropy conserving fluxes can be used as building blocks for high-order accurate entropy conservative fluxes following the procedure by LeFloch, Mercier and Rohde [154]. As often seen in the literature, Fjordholm et al. also propose to add numerical diffusion proportional to the jump in entropy variables and provide instructions on how to obtain such diffusion terms of high-order accuracy [69]. Similar to the TeCNO schemes, Ismail and Roe also construct an entropy conservative scheme first, and then add entropy dissipation at the shocks. They take special precautions to ensure that the entropy dissipation is at an appropriate level in an effort to achieve entropy consistency [117]. A fully discrete extension to the TeCNO schemes for scalar conservation laws was found by Zakerzadeh and Fjordholm in 2015 by deriving a suitable CFL condition [267]. An application of the TeCNO methodology on unstructured grids is presented in [208], but is restricted to second order accuracy.

Since then, compact closed form expressions for entropy conservative and entropy stable numerical fluxes have been found for many other systems as well. Chandrashekar [42] proposes a kinetic energy preserving and entropy stable finite volume method with a numerical flux in compact form for the Euler and Navier-Stokes equations. Entropy conservative and entropy stable numerical fluxes for the ideal MHD equations as well as a discussion of the entropy stable dissipation operator can be found in [257, 255, 53, 52]. Kumar and Mishra propose an entropy conservative flux to develop a second order finite difference scheme for the two-fluid ideal plasma equations [142].

Another important concern is the correct (entropy stable) treatment of domain boundaries. Entropy stable boundary conditions have been developed, such as far-field and wall boundary conditions for the Euler equations [230] as well as solid wall boundary conditions for the compressible and incompressible Navier Stokes equations [199, 185].

A breakthrough for high-order entropy stable methods on unstructured, possibly curved, quadrilateral meshes was finally achieved by Carpenter and Fisher in 2013. By constructing WENO finite difference schemes (or equivalent DG methods) based on diagonal norm summation-by-parts (SBP) operators, they are able to achieve entropy stable schemes of arbitrary order for systems of non-linear conservation laws such as the Navier-Stokes equations [67, 63, 65, 66, 64, 30]. Their work plays a pivotal role in the development of the entropy stable method for the shallow water equations presented in this thesis and we will describe their method as well as summation-by-parts in more detail in Chapter 4.

Many researchers have since built upon the works of Carpenter and Fisher. Gassner established a skew-symmetric DG discretization and highlights the connection to the SBP-SAT framework in [78]. Here, SAT stands for simultaneous approximation terms. Following this principle, entropy stable DG methods were developed for the shallow water equations [83, 256], the Euler equations [82] and the ideal MHD as well as resistive MHD equations on two dimensional Cartesian meshes [161] and general curvilinear meshes [18]. An extension to unstructured triangular meshes was found by Chen et al. in [43], using suitable quadrature rules and multi-dimensional SBP operators as introduced by Hicken and Crean [108, 47]. Furthermore, variants for non-diagonal norm SBP operators were introduced by Chan [38], proposing that diagonal norm SBP operators are not the most efficient for triangular meshes. Finally, extensions to space-time entropy stable DG methods were recently published by Friedrich et al. [71].

The contribution of the numerical method presented in this thesis is to extend the works on entropy stable DG schemes for the shallow water equations in [83, 256] to unstructured two dimensional meshes with curved geometries. Furthermore, artificial viscosity is introduced as a shock capturing mechanism and a positivity preservation technique is added. Both of these expansions are designed to uphold the entropy stable nature of the scheme. The details of this method are presented in Chapter 5. A computationally efficient implementation on modern GPU architectures is presented in Chapter 6. Furthermore this method is applied to the real world example of the 2004 Indian Ocean tsunami. The details and results are presented in Chapter 8.

3.4. Well-balanced & positivity preserving schemes

In the specific context of the shallow water equations another numerical property becomes important: The capability of a numerical scheme to preserve the steady state solution of the "lake-at-rest" (2.37) for all time.

Such a numerical scheme is said to be **well-balanced**, or to have the "exact conservation property" (exact C-property), a concept first proposed by Bermudez and Vazquez [14]. It is of critical importance for shallow water approximations since many real world applications are modeled as only slight variations from an equilibrium state. The initial conditions of the Indian Ocean tsunami in Chapter 8 include only a small deviation from a flat sea level at the location of the earthquake. If a scheme is not well-balanced, numerical waves can build up in unperturbed regions and pollute the solution, or in the worst case, even crash the simulation. Consequently this is an immensely active field of research and well-balanced numerical methods for the shallow water equations have been proposed by many researchers for a variety of numerical methods, e.g. [143, 120, 243, 94, 95, 247, 28, 196, 195, 164, 29, 74].

One fundamental strategy to achieve the exact C-property is to base the numerical method on a *pre-balanced* version of the shallow water equations [211]. Well-balanced schemes following this approach can, for example, be found in [125, 272, 157].

Another approach to design well-balanced methods is due to Audasse et al., who introduced the idea of using a *hydrostatic reconstruction* on the interface fluxes in their second order finite volume schemes [6]. The hydrostatic reconstruction involves a modification of the left and right states at each interface based on the solution as well as a specific source term discretization [6]. The high-order well-balanced FV-WENO schemes by Noelle et al. are based on the same hydrostatic reconstruction approach [180, 181]. Following these ideas, Xing and Shu proposed a novel way of constructing well-balanced high-order FV-WENO and DG methods for hyperbolic conservation laws, specifically the shallow water equations [262]. Their high-order FV-WENO scheme can be considered a generalization of the scheme by Noelle, allowing more freedom in defining the polynomials approximating the water height and bottom topography [262].

These schemes have been extended with the capability of handling dry by Xing [265, 263]. The positivity is secured by introducing an extra limiting step in each time step (or Runge-Kutta stage). The limiter is based on a linear scaling of the solution values around cell averages. To guarantee positive values, this method requires the numerical fluxes to fulfill the so-called positivity property. Simply put, the numerical fluxes must guarantee that cell updates for first order finite volume schemes preserve non-negative values. This strategy was originally introduced by Perthame and Shu for finite volume methods applied to the Euler equations [204] and was further developed in [270, 271]. It has been adapted to many different numerical methods and applied to several conservation laws, such as the Euler equations [269] or the MHD equations [44]. These schemes exist for quadrilateral [265] and triangular meshes [264]. The positivity preserving technique presented in this work for the entropy stable DGSEM in Chapter 5 is also based on the positivity preserving approach of Xing [265].

In [112], Hiltebrand and Mishra design a fully discrete space-time DG scheme that is entropy stable and well-balanced and works on unstructured grids. The key idea behind this scheme is the use of entropy variables as degrees of freedom. Also, entropy conservative numerical fluxes are used and diffusion is added in terms of the entropy variables. This is very similar to the way dissipation is added to the entropy stable DG scheme as described in Section 5.3. The space-time scheme is completed with a suitable discretization of the bottom topography as well as streamline diffusion and shock-capturing operators. The scheme works on unstructured triangular grids. The downside of this scheme is that the proofs assume exact integration of the integrals. In practice, this is difficult to guarantee. Also, the computational cost of space-time schemes is typically high and computations with high polynomial degrees require special care.

4. SBP: Split forms and Flux differencing

Numerical discretizations based on the split form of a system of conservation or balance laws often show remarkable properties such as increased robustness or energy/entropy stability [16, 114, 59, 134, 82, 258]. The downside of this approach is, however, that such discretizations are typically not conservative. This is due to the fact that the continuous product rule

$$ab_x + a_x b = (ab)_x, \tag{4.1}$$

generally does not hold discretely. Similarly, the continuous integration by parts reduces the volume integral of product rule terms over a domain Ω to a surface integral on the boundary S,

$$\int\limits_{\Omega} ab_x \, \mathrm{d}\Omega + \int\limits_{\Omega} a_x b \, \mathrm{d}\Omega = \oint\limits_{\partial\Omega} ab \, \mathrm{dS}, \tag{4.2}$$

and the same is not guaranteed to hold in a discrete sense.

Thus, split form discretizations are, by design, not in conservative form and it is not guaranteed that the approximation remains globally conservative. This is a critical issue, since solutions obtained by a non-conservative numerical approximation of a hyperbolic conservation law do not satisfy the Lax-Wendroff theorem. Solutions of non-conservative schemes tend to produce wrong shock speeds and may not approximate an actual weak solution to the conservation law. Thus, it is very important to construct conservative numerical schemes.

In an effort to overcome this difficulty, so called summation-by-parts operators have been introduced by Kreiss et al. for the finite difference community [141]. Coupled with a simultaneous approximation term on the interfaces [34, 73], this methodology has been successfully applied to a wide variety of problems, including the Euler equations [168, 109] and Navier-Stokes equations [169, 184, 194]. SBP methods have been picked up and applied to many other numerical methods, e.g., by Nordström [182, 183] and Svärd and Nordström [227] for finite volume methods. The technique has also been applied to flux reconstruction schemes [207] and spectral methods [33]. Recently the SBP framework was applied to achieve stable time discretizations, as well [186, 165, 187]. Del Rey Fernández et al. introduce a systematic approach for constructing nodal first derivative SBP operators from quadrature rules in [60].

In the context of discontinuous Galerkin methods, SBP operators have been successfully applied to find entropy stable schemes for the Burgers' equation [78], the Euler & Navier-Stokes equations [30], shallow water equations [82, 256, 253] and resistive MHD equations [18].

The first section of this chapter is dedicated to a formal introduction of SBP operators and their useful properties. In the following section we describe the flux-differencing method, which is based on a sub-class of these SBP operators with diagonal norm. Further, we describe the advantages of the flux-differencing formulation and the relationship to discontinuous Galerkin methods. In the final section of this chapter we demonstrate how the flux-differencing formulation can be applied in the approximation of conservation laws in split form.

4.1. Summation-by-parts

SBP operators are first derivative operators coupled with a quadrature rule that mimic the continuous property of integration by parts on a discrete level. Specifically, SBP operators require the discrete integration by parts to hold:

$$\int\limits_{\Omega,N} a b_x \,\mathrm{d}\Omega + \int\limits_{\Omega,N} a_x b \,\mathrm{d}\Omega \overset{!}{=} \oint\limits_{\partial\Omega,N} ab \,\mathrm{dS}. \tag{4.3}$$

This can be achieved by imposing the following requirements on the derivative operator.

Definition 1 (SBP operator). A matrix $\mathbf{D} \in \mathbb{R}^{(n+1)\times(n+1)}$ is a SBP operator if it has the form $\mathbf{D} = \mathbf{M}^{-1}\mathbf{Q}$ with

- weight matrix $\mathbf{M} \in \mathbb{R}^{(n+1)\times(n+1)}$ that is symmetric positive definite,
- $\mathbf{Q} \in \mathbb{R}^{(n+1)\times(n+1)}$ satisfies the SBP property

$$\mathbf{Q} + \mathbf{Q}^T = \mathbf{B} := \operatorname{diag}(-1, 0, \dots, 0, 1). \tag{4.4}$$

SBP operators with diagonal weight matrix $\mathbf{M}$ are of particular interest as they lead to stable approximations on curvilinear grids [225]. A downside of diagonal norm SBP operators for finite difference schemes is that the order of accuracy is limited to $s + 1$ for stencil size s [98]. In contrast, for dense mass matrices $\mathbf{M}$, an interior order of $2s$ is achievable [111]. In the context of finite difference methods, higher order accuracy is also achievable for functionals based on the solution of dual consistent diagonal-norm SBP discretizations [110].

Although the finite difference and spectral element approximations differ (e.g., the spectral element approximate solution is known everywhere, including in between the nodes), we aim to use the established properties and advantages of SBP with regard to discontinuous Galerkin methods. As a first step, we note that the LGL derivative operator is a diagonal norm SBP operator [78].

Lemma 3 (SBP-Properties)**.** *Let* $\mathbf{D} := \mathbf{M}^{-1}\mathbf{Q}$ *be the LGL derivative operator. The matrix* $\mathbf{Q}$ *has the SBP-property* (4.4)*.*

Proof. See, for example, [140, 141, 78, 30, 80]. ■

Remark 1. The SBP-property can be used to obtain alternative expressions for the derivative matrix. The surface operator $\mathbf{S}$ is defined by

$$\mathbf{S} := -\mathbf{M}^{-1}\mathbf{B} = \operatorname{diag}\left(\frac{1}{\omega_0}, 0, \dots, 0, -\frac{1}{\omega_N}\right). \tag{4.5}$$

We find

$$\mathbf{D} = \mathbf{M}^{-1}\mathbf{Q} = \mathbf{M}^{-1}(\mathbf{B} - \mathbf{Q}^T) = -\mathbf{S} - \mathbf{M}^{-1}\mathbf{Q}^T. \tag{4.6}$$

So, the derivative matrix $\hat{\mathbf{D}}$ of the weak LGL-DG formulation [137] given by

$$\hat{\mathbf{D}} = -\mathbf{M}^{-1}\mathbf{Q}^T = -\mathbf{M}^{-1}\mathbf{D}^T\mathbf{M}, \tag{4.7}$$

satisfies the relation

$$\mathbf{D} = \hat{\mathbf{D}} - \mathbf{S}. \tag{4.8}$$

It follows that for diagonal norm SBP operators, weak and strong form DG discretizations are equivalent on a discrete level. This was previously noted by Kopriva and Gassner [138].

Corollary 1 (Properties of SBP operators). For derivative operators $\mathbf{D} := \mathbf{M}^{-1}\mathbf{Q}$ with SBP property, the SBP-operator $\mathbf{Q}$ has several useful properties, that follow directly from the definition:

- $\sum\limits_{j=0}^{N} Q_{ij} = 0$ for $i = 0, \dots, N$ (rows sum to zero),
- $\sum\limits_{i=0}^{N} Q_{ij} = \begin{cases} -1 & \text{if } j = 0 \\ 0 & \text{if } j = 1, \dots, N-1 \\ 1 & \text{if } j = N \end{cases}$ (column sums),
- $Q_{ii} = \begin{cases} -\frac{1}{2} & \text{if } i = 0 \\ 0 & \text{if } i = 1, \dots, N-1 \\ \frac{1}{2} & \text{if } i = N \end{cases}$ (diagonal entries),
- $Q_{ij} = -Q_{ji}$ for $i, j = 0, \dots, N$ and $i \neq j$ (nearly skew symmetric).

A key property of SBP operators is, that it allows us to discretize split forms of conservation laws in a conservative way. Split forms are usually derived by a weighted sum of conservative and non-conservative terms. For example, split forms of the simple term $(ab)_x$ are found by applying the product rule and using a combination of both formulations,

$$(ab)_x = \lambda(ab)_x + (1-\lambda)\left(a_x b + a b_x\right), \qquad \lambda \in [0,1]. \tag{4.9}$$

The limiting cases of $\lambda = 0$ and $\lambda = 1$ yield the purely advective formulation and the conservative formulation respectively. Any split form (4.9) can be reformulated as

$$(ab)_x = (ab)_x + (1-\lambda)\left(-(ab)_x + a_x b + a b_x\right), \qquad \lambda \in [0,1], \tag{4.10}$$

which is the the sum of the original derivative and a product rule error term. This product rule error term is zero on the continuous level but generally not zero in the discrete case. Thus, discretely, the different formulations obtained from (4.9) are not identical. While we know that a discontinuous Galerkin discretization of the integral of the original derivative $(ab)_x$ is conservative, the same is not guaranteed when discretizing the split form equation. This is where the SBP property of the LGL derivative operator in the DG approximation becomes important. With this property it is possible to show that the introduced discrete product rule error terms vanish when integrated over an element. Thus, a discretization of the split formulation is still a locally conservative approximation. We show this for the two dimensional case in the following Lemma.

Lemma 4 (Non-Linear Correction Terms)**.** *For a derivative operator* $\mathbf{D}$ *with SBP property, the error in the discrete product rule in two dimensions*

$$\left(\mathcal{E}_{\alpha\beta}\right)_{ij} := -\sum_{m=0}^{N} D_{im}\alpha_{mj}\beta_{mj} + \alpha_{ij}\sum_{m=0}^{N} D_{im}\beta_{mj} + \beta_{ij}\sum_{m=0}^{N} D_{im}\alpha_{mj}, \tag{4.11}$$

is zero when discretely integrated over an element,

$$\sum_{i,j=0}^{N} \left(\mathcal{E}_{\alpha\beta}\right)_{ij}\omega_i\omega_j = 0, \tag{4.12}$$

for all $\alpha_{ij}, \beta_{ij} \in \mathbb{R}$ *for* $i, j = 0\dots, N$. *Furthermore the discretization of the split form* (4.9)

$$\lambda\sum_{m=0}^{N} D_{im}\alpha_{mj}\beta_{mj} + (1-\lambda)\left(\alpha_{ij}\sum_{m=0}^{N} D_{im}\beta_{mj} + \beta_{ij}\sum_{m=0}^{N} D_{im}\alpha_{mj}\right), \tag{4.13}$$

where $\boldsymbol{\alpha}$ *and* $\boldsymbol{\beta}$ *are the point-wise nodal representations of functions* α *and* β*, is a consistent and conservative approximation of the derivative of the corresponding flux* $(\alpha\,\beta)_x$ *for all* $\lambda \in [0,1]$.

Proof. See Appendix A.3 ■

Corollary 2 (Special case: One dimension)*.* The result from Lemma 4 holds for one dimension, as well. Non-linear correction terms such as

$$\left(\mathcal{E}_{\alpha\beta}\right)_{i} := -\sum_{m=0}^{N} D_{im}\alpha_{m}\beta_{m} + \alpha_{i}\sum_{m=0}^{N} D_{im}\beta_{m} + \beta_{i}\sum_{m=0}^{N} D_{im}\alpha_{m}, \tag{4.14}$$

also do not destroy global conservation

$$\sum_{i=0}^{N} \left(\mathcal{E}_{\alpha\beta}\right)_i \omega_i = 0. \tag{4.15}$$

This was shown in [80]. The proof is analogous to that of Lemma 4 by simply dropping the outer sum over index j in the two dimensional proof, specifically in (A.39).

This Lemma is a very important tool when considering split form discontinuous Galerkin approximations. It immediately follows that DG schemes based on the split form are still (at least locally) conservative, if the derivative operator has the SBP property. It it is also helpful to show that certain derived quantities, that are not directly approximated, are conserved quantities as well. To this end, a discrete expression for the time derivative of such a quantity has to be found. Then, if the spatial derivatives can be recast into a split form, such that there is a consistent and conservative approximation to a spatial flux and all the remaining terms can be recast in the shape of (4.13), the derived quantity is locally conserved. If the interface fluxes are consistent as well, the derived quantity is globally conserved [106, 107]. For the purposes of this work, we are mainly interested in the entropy evolution, but other quantities such as the potential vorticity could be considered as well [237, 200, 3].

While the use of SBP operators for discretizations of split formulations are a powerful tool to construct entropy conservative schemes, the resulting formulations appear to be impractical. Robust and entropy conserving split form schemes may require the evaluation of distinct terms. Especially on curved meshes that require contravariant fluxes as in (3.52), a suitable split form quickly leads to an unmanageable and inefficient amount of volume integrals in the computation. Luckily, there is a more efficient way formulate and implement the split form discretization. As we will see, the discretized split forms are equivalent to certain flux differencing schemes as introduced by Fisher and Carpenter [65]. We will introduce this class of schemes in in Section 4.2 and establish their equivalency to split forms in Section 4.3.

4.2. The Flux Differencing Framework by Carpenter & Fisher

Fisher and Carpenter found that SBP derivative operators with diagonal mass matrix $\mathbf{M}$ are algebraically equivalent to a finite volume differencing formulation on a sub-cell grid [65]. In this section, we describe these flux differencing schemes and start with the one dimensional version. As these formulations require frequent evaluations of averages between nodal values, we introduce an abbreviated notation for the average values in one dimension

$$\{\{a\}\}_{(i,j)} = \frac{a_i + a_j}{2}, \tag{4.16}$$

and two dimensions

$$\begin{aligned} \{\{a\}\}_{(i,m),j)} &= \frac{a_{ij} + a_{mj}}{2}, \\ \{\{a\}\}_{i,(j,m))} &= \frac{a_{ij} + a_{im}}{2}. \end{aligned} \tag{4.17}$$

4.2.1. Flux differencing in one dimension

The flux differencing schemes use a complimentary, staggered sub-cell grid, where the spacing is defined by the quadrature weights ω_i. While the new nodes $\bar{\xi}_j$ coincide with the solution nodes on the interfaces, they are always located between two solution nodes on the interior:

$$\begin{aligned} \bar{\xi}_0 &= \xi_0, \\ \bar{\xi}_j &= \bar{\xi}_{j-1} + \omega_{j-1}\,, \qquad\qquad j = 1, \ldots, N, \\ \bar{\xi}_{N+1} &= \xi_N. \end{aligned} \tag{4.18}$$

The flux differencing method uses special fluxes, $\overline{F}$, which are evaluated on the complimentary grid at the new flux nodes. The spatial derivative of a flux f at a solution node ξ_i is then approximated by

$$(f_\xi)_i \approx \frac{1}{\omega_i}\left(\overline{F}_{i+1} - \overline{F}_i\right), \qquad\qquad i = 0, \ldots, N. \tag{4.19}$$

We show such a grid for the LGL nodes and weights at $N = 3$ in Figure 4.1. The flux

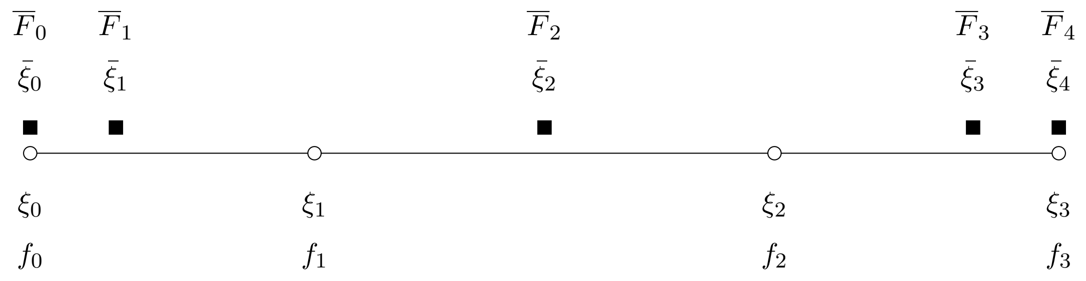


Figure 4.1.: One dimensional complimentary grid for LGL nodes with $N = 3$. The solution nodes (circles) and fluxes are labeled below the graph and the new flux points (squares) and differencing fluxes are labeled above.

differencing approximation to the flux derivative in (4.19) can also be stated in matrix vector form by

$$(f_\xi)_i \approx \sum_{m=0}^{N} M_{ii}^{-1} \Delta_{im} \overline{F}_m, \tag{4.20}$$

where $\mathbf{\Delta}$ is the $N \times N+1$ differencing matrix

$$\mathbf{\Delta} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}. \tag{4.21}$$

In [66] it is shown that such an equivalent flux differencing formulation exists for all diagonal norm SBP operators $\mathbf{D}$. The spatial approximation (4.19) is very similar in structure to finite volume methods, provided the fluxes $\overline{F}$ are interpreted as numerical fluxes on the interfaces of a cell with size ω_i. Also, similar to finite volume methods, which achieve high-order approximations by considering larger stencils in their reconstruction step, we can also obtain high-order approximations by considering more values in the calculation of $\overline{F}$. If all the interpolation nodes are available as input, Fisher and Carpenter show that for a specific choice of $\overline{F}$ the order of accuracy $N+1$ is obtained. We summarize this result from [65] in the following Lemma.

Lemma 5 (High-order flux). *If the flux in* (4.19) *is determined by*

$$\begin{aligned} \overline{F}_0 &:= f_0, \\ \overline{F}_i &:= \sum_{m=i}^{N} \sum_{l=0}^{i-1} 2\, Q_{lm} F^{\#}_{(l,m)}, \\ \overline{F}_{N+1} &:= f_N, \end{aligned} \tag{4.22}$$

with a consistent and symmetric numerical two-point flux

$$F^{\#}_{(i,m)} := F^{\#}\left(W_i, W_m,\right), \tag{4.23}$$

Then the approximation to the spatial flux derivative is of order $N+1$. Further, if the two-point flux $F^{\#}$ is entropy conservative, then the high-order flux remains entropy conservative as well.

Proof. See [64]. ■

Considering the approximation to the derivative (4.20) and the construction of higher order fluxes according to Lemma 5, we find that the expressions in the former can be greatly simplified.

Lemma 6 (Simplified Flux Differencing). *The structure of the SBP matrix $\mathbf{Q}$ and the construction of the high-order fluxes on the complimentary grid given in Lemma 5 allow the approximation of the spatial derivative to be re-written in the following way:*

$$\sum_{m=0}^{N} M_{ii}^{-1} \Delta_{im} \overline{F}_m = \frac{\overline{F}_{i+1} - \overline{F}_i}{\omega_i} = \sum_{m=0}^{N} 2\, D_{im}\, F^{\#}(W_i, W_m), \tag{4.24}$$

for $i = 0, \ldots, N$.

Proof. We recall the high-order flux extension (4.22) and consider the special cases $i=0$ and $i=N$ first. From the property $Q_{00} = -\frac{1}{2}$ we find for $i=0$

$$\overline{F}_1 - \overline{F}_0 = \sum_{m=1}^{N} 2Q_{0m}\, F^{\#}_{(0,m)} - f_0 = \sum_{m=0}^{N} 2Q_{0m}\, F^{\#}_{(0,m)}. \tag{4.25}$$

Similarly, from $Q_{NN} = \frac{1}{2}$ and the skew-symmetry it follows for $i=N$

$$\overline{F}_{N+1} - \overline{F}_N = f_N - \sum_{m=0}^{N-1} 2Q_{mN}\, F^{\#}_{(m,N)} = \sum_{m=0}^{N} 2Q_{Nm}\, F^{\#}_{(m,N)}. \tag{4.26}$$

For the interior points $i = 1, \ldots, N-1$, we look at the construction of the high-order fluxes (4.22) to find

$$\begin{aligned}
\overline{F}_{i+1} &= \sum_{m=i+1}^{N} \sum_{l=0}^{i} 2Q_{lm}\, F^{\#}_{(l,m)} \\
&= \sum_{m=i}^{N} \sum_{l=0}^{i} 2Q_{lm}\, F^{\#}_{(l,m)} - \sum_{l=0}^{i} 2Q_{li}\, F^{\#}_{(l,i)} \\
&= \sum_{m=i}^{N} \sum_{l=0}^{i-1} 2Q_{lm}\, F^{\#}_{(l,m)} + \sum_{m=i}^{N} 2Q_{im}\, F^{\#}_{(i,m)} - \sum_{l=0}^{i} 2Q_{li}\, F^{\#}_{(l,i)} \\
&= \overline{F}_i + \sum_{m=0}^{N} 2Q_{im}\, F^{\#}_{(i,m)},
\end{aligned} \tag{4.27}$$

where we used that $Q_{ij} = -Q_{ij}$ for $i \neq j$ and $Q_{ii} = 0$ for $i = 1, \ldots, N-1$ in the last step. Dividing by the integration weights, we find the simplified expression for the calculation of the flux difference for $i = 0, \ldots N$

$$\frac{\overline{F}_{i+1} - \overline{F}_i}{\omega_i} = \sum_{m=0}^{N} 2D_{im}\, F^{\#}_{(i,m)}. \tag{4.28}$$

■

Lemma 5 and Lemma 6 are useful as they establish a new efficient and implementation friendly way to approximate interior split form volume integrals. They provide a relatively simple way to construct entropy conservative discretizations, as the entropy conservation of low-order FV fluxes carries over to the high-order fluxes from Lemma 5. The simplified expression from Lemma 6 greatly reduces the computational cost, as practically one sum is eliminated when compared to an a priori evaluation of all the high-order fluxes $\overline{F}$ and subtracting them afterwards. A numerical method however is only complete with a suitable interface coupling between neighbouring elements as well as an appropriate way of incorporating physical boundary conditions. A widely used technique from the finite difference community is the introduction of simultaneous approximation terms [34, 228, 226, 229, 184, 13]. These terms preserve the accuracy of the scheme and also provide a convenient way of adding dissipation while remaining consistent with the Lax-Wendroff Theorem [65, 66, 64].

We choose a slightly different approach. To obtain a complete numerical scheme, we start with the strong form of the conservation law

$$\int_{E_0} \mathcal{J}\, w_t\, \phi d\xi + \left[(F^* - f)\phi\right]_{-1}^{+1} + \int_{E_0} f_\xi \phi\, d\xi = 0. \tag{4.29}$$

We insert the polynomial ansatz and the LGL quadrature as in the derivation of the discontinuous Galerkin method, but approximate the volume integral with the telescopic flux differencing (4.20) in simplified form (4.24). The resulting numerical method is written completely in a form similar to the standard strong form DGSEM derived in (3.44) and (3.46):

$$JW_t + \mathcal{L}_\xi = 0. \tag{4.30}$$

While the surface terms in the spatial operator are identical to the DGSEM, the volume integral is computed slightly differently

$$\left(\mathcal{L}_\xi\right)_i = \frac{1}{\omega_i}\left(\delta_{iN}\left[F^* - F\right]_N - \delta_{i0}\left[F^* - F\right]_0\right) + \frac{1}{\omega_i}\sum_{m=0}^{N} 2\, Q_{im}\, F^{\#}(W_i, W_m). \tag{4.31}$$

For ease of notation, we have only considered scalar conservation laws thus far. However, the same steps hold for systems of balance laws as well. We proceed to outline the flux differencing scheme in two dimensions and on curved meshes.

4.2.2. Flux differencing in two dimensions

The staggered grid in two dimensions is obtained by a simple tensor product ansatz. Thus, we have a set of $(N+2)^2$ flux nodes $(\bar{\xi}_i, \bar{\eta}_j)$ for $i, j = 0, \ldots, N+1$, where both coordinates are constructed according to (4.18). The major difference between one and two dimensions is the possibility of curved geometries. As a consequence, we must consider the contravariant fluxes (3.52) for the flux differencing

$$\begin{aligned}
(\tilde{f}_\xi)_{ij} &\approx \frac{1}{\omega_i}\left(\overline{\tilde{F}}_{i+1,j} - \overline{\tilde{F}}_{i,j}\right), \qquad &(\tilde{f}_\eta)_{ij} &\approx \frac{1}{\omega_j}\left(\overline{\tilde{F}}_{i,j+1} - \overline{\tilde{F}}_{i,j}\right), \\
(\tilde{g}_\xi)_{ij} &\approx \frac{1}{\omega_i}\left(\overline{\tilde{G}}_{i+1,j} - \overline{\tilde{G}}_{i,j}\right), \qquad &(\tilde{g}_\eta)_{ij} &\approx \frac{1}{\omega_j}\left(\overline{\tilde{G}}_{i,j+1} - \overline{\tilde{G}}_{i,j}\right),
\end{aligned} \tag{4.32}$$

for $i, j = 0, \ldots, N$. Once again, the crucial task is to find sub-cell fluxes on the staggered grid $\overline{\tilde{F}}$ and $\overline{\tilde{G}}$ that are consistent, high-order accurate and entropy conservative. To allow general curved meshes, the metric terms need to be incorporated into the definition of these complimentary fluxes. As in the one dimensional case, a way to construct consistent high-order fluxes at the flux nodes is given by Fisher & Carpenter in [30, 63]. We summarize their results in the following Lemma. This Lemma holds for systems of conservation or balance laws, but we state it in scalar form to make the notation more digestible.

Lemma 7 (High-Order Flux). *A two-point high-order flux in x-direction can be constructed by*

$$\begin{aligned}
\overline{\tilde{F}}_{0,j} &:= \tilde{F}_{0,j}, \\
\overline{\tilde{F}}_{i,j} &:= \sum_{m=i}^{N}\sum_{l=0}^{i-1} 2\, Q_{lm}\left(\tilde{F}^{\#}_{(l,m),j} + \tilde{G}^{\#}_{(l,m),j}\right), \\
\overline{\tilde{F}}_{N+1,j} &:= \tilde{F}_{N,j},
\end{aligned} \tag{4.33}$$

for $i = 1, \dots, N$ and a fixed $y-$direction. Analogously, the high-order flux in y-direction is constructed by

$$\begin{aligned}
\overline{\tilde{G}}_{i,0} &:= \tilde{G}_{i,0}, \\
\overline{\tilde{G}}_{i,j} &:= \sum_{m=i}^{N}\sum_{l=0}^{j-1} 2\,Q_{lm}\left(\tilde{F}^{\#}_{i,(l,m)} + \tilde{G}^{\#}_{i,(l,m)}\right), \\
\overline{\tilde{G}}_{i,N+1} &:= \tilde{G}_{i,N},
\end{aligned} \tag{4.34}$$

where $\tilde{F}_{0,j}$, $\tilde{F}_{N,j}$, $\tilde{G}_{i,0}$ and $\tilde{G}_{i,N}$ are the typical contravariant fluxes. The numerical two-point fluxes are simplified for brevity by the following notation, that incorporates the metric terms

$$\begin{aligned}
\tilde{F}^{\#}_{(i,m),j} &:= \tilde{F}^{\#}\left(W_{i,j}, W_{i,m}, (y_\eta)_{i,j}, (y_\eta)_{i,m}\right) \\
\tilde{G}^{\#}_{(i,m),j} &:= \tilde{G}^{\#}\left(W_{i,j}, W_{i,m}, (-x_\eta)_{i,j}, (-x_\eta)_{i,m}\right) \\
\tilde{F}^{\#}_{i,(j,m)} &:= \tilde{F}^{\#}\left(W_{i,j}, W_{i,m}, (-y_\xi)_{i,j}, (-y_\xi)_{i,m}\right) \\
\tilde{G}^{\#}_{i,(j,m)} &:= \tilde{G}^{\#}\left(W_{i,j}, W_{i,m}, (x_\xi)_{i,j}, (x_\xi)_{i,m}\right).
\end{aligned} \tag{4.35}$$

Again, these fluxes must be symmetric and consistent to the physical fluxes of the considered equation. Furthermore, if the two-point fluxes are entropy conservative then the high-order fluxes (4.33) *and* (4.34) *are entropy conservative as well.*

Proof. This Lemma from [30] is proven for the Cartesian case in [65], and extended to curvilinear meshes in [63, Section 4.5 and Appendix A.3]. ■

Remark 2. An entropy conservation condition on the numerical two-point fluxes for a system of balance laws is given by

$$[\![\vec{q}]\!]^T(\vec{F}^{\#} + \vec{G}^{\#}) = [\![\phi_f + \phi_g]\!] + \{\{\vec{q}\}\}^T\,\vec{s}, \tag{4.36}$$

with discretized source term vector $\vec{S}$. This source term extension to the entropy conservation condition (4.36) can, for instance, be found in [68].

Again, we are interested in simplifying the calculation of the flux difference. Analogously to the one dimensional case, we are able to cancel one of the sums in the telescoping sum $\sum_{m=0}^{N} M_{ii}^{-1}\Delta_{im}\overline{\tilde{F}}_{mj}$ if the high-order flux $\overline{\tilde{F}}$ is constructed as in (4.33).

Lemma 8 (Simplified Flux Differencing - 2D)**.** *One can use the structure of the SBP matrix* $\mathbf{Q}$ *and the fluxes on the complimentary grid to eliminate one of the sums in the telescoping flux sum, to write the flux difference in the $x-$direction in the indicial form*

$$\sum_{m=0}^{N} M_{ii}^{-1}\Delta_{im}\overline{\tilde{F}}_{mj} = \frac{\overline{\tilde{F}}_{i+1,j} - \overline{\tilde{F}}_{i,j}}{\omega_i} = \sum_{m=0}^{N} 2\,D_{im}\left(\tilde{F}^{\#}_{(i,m),j} + \tilde{G}^{\#}_{(i,m),j}\right), \tag{4.37}$$

for $i = 0, \dots, N$. The flux difference in the y-direction can also be expressed in a similar indicial form

$$\sum_{m=0}^{N} M_{jj}^{-1} \Delta_{jm} \overline{\tilde{G}}_{im} = \frac{\overline{\tilde{G}}_{i,j+1} - \overline{\tilde{G}}_{i,j}}{\omega_j} = \sum_{m=0}^{N} 2\, D_{mj} \left(\tilde{F}^{\#}_{i,(j,m)} + \tilde{G}^{\#}_{i,(j,m)} \right), \tag{4.38}$$

for $j = 0, \dots, N$.

Proof. The proof is nearly identical to the one dimensional proof from Lemma 6 in each spatial direction with the additional consideration of the metric terms and can be found in Appendix A.4. ■

Following the same approach as in the previous section, we derive the flux differencing scheme from the variational formulation of the scalar conservation law in strong form (3.54). The only difference to the standard DGSEM is the different approximation of the volume integral, which is now performed by flux differencing. This leads to the following semi-discrete scheme,

$$J W_t + \mathcal{L}_\xi + \mathcal{L}_\eta = 0 \tag{4.39}$$

where the spatial operators are defined by

$$\begin{aligned}
(\mathcal{L}_\xi)_{ij} =& \frac{1}{\omega_i} \left(\delta_{iN} \left[\tilde{F}^* - \tilde{F} \right]_{Nj} - \delta_{i0} \left[\tilde{F}^* - \tilde{F} \right]_{0j} \right) \\
&+ \sum_{m=0}^{N} 2\, D_{im} \left(\tilde{F}^{\#}_{(i,m),j} + \tilde{G}^{\#}_{(i,m),j} \right), \\
(\mathcal{L}_\eta)_{ij} =& \frac{1}{\omega_j} \left(\delta_{Nj} \left[\tilde{G}^* - \tilde{G} \right]_{iN} - \delta_{0j} \left[\tilde{G}^* - \tilde{G} \right]_{i0} \right) \\
&+ \sum_{m=0}^{N} 2\, D_{mj} \left(\tilde{F}^{\#}_{i,(j,m)} + \tilde{G}^{\#}_{i,(j,m)} \right),
\end{aligned} \tag{4.40}$$

with the curvilinear two-point fluxes $\tilde{F}^{\#}$ and $\tilde{G}^{\#}$ according to (4.35) and a suitable numerical interface flux, e.g., the Lax-Friedrichs numerical flux (3.80). All these steps work analogously for systems of balance laws and, again, we have only considered a scalar conservation law to simplify notation. An entropy conservative or entropy stable flux differencing scheme can be constructed with the selection of appropriate entropy conservative high-order numerical two-point fluxes according to Lemma 7 and entropy conservative or stable interface fluxes.

The similarity between DGSEM and flux differencing scheme is, of course, no coincidence. In fact, the DGSEM is a special case of the flux differencing method for one specific choice of two-point fluxes $F^{\#}$ and $G^{\#}$ [78]. For this specific choice, one finds that the derivatives of the contravariant fluxes in the DGSEM can be written into a telescoping flux form

$$\begin{aligned}
\sum_{m=0}^{N} D_{im}\tilde{F}_{mj} &= \sum_{m=0}^{N} M_{ii}^{-1}Q_{im}\tilde{F}_{mj} = \sum_{m=0}^{N} M_{ii}^{-1}\Delta_{im}\overline{\tilde{F}}_{mj},\\
\sum_{m=0}^{N} D_{jm}\tilde{G}_{im} &= \sum_{m=0}^{N} M_{jj}^{-1}Q_{jm}\tilde{G}_{im} = \sum_{m=0}^{N} M_{jj}^{-1}\Delta_{jm}\overline{\tilde{G}}_{im}.
\end{aligned} \tag{4.41}$$

Other choices of the numerical two-point flux lead to DG discretization of split forms of the physical equations. We explore this venue further in the next Section.

The standard curvilinear DGSEM from Section 3.2.2 is equivalent to the flux differencing method (4.39) and (4.40) if the following numerical two-point fluxes are chosen:

$$\begin{aligned}
\tilde{F}^{\#}_{(i,m),j} &:= \{\{(y_\eta)\,F\}\}_{(i,m),j}\,,\\
\tilde{G}^{\#}_{(i,m),j} &:= \{\{(-x_\eta)\,G\}\}_{(i,m),j}\,,\\
\tilde{F}^{\#}_{i,(j,m)} &:= \{\{(-y_\xi)\,F\}\}_{i,(j,m)}\,,\\
\tilde{G}^{\#}_{i,(j,m)} &:= \{\{(x_\xi)\,G\}\}_{i,(j,m)}\,,
\end{aligned} \tag{4.42}$$

with the notation for averages given in (4.17). We see the equivalence between the two formulations by inserting (4.42) into (4.37) and simplifying terms by using the SBP properties,

$$\begin{aligned}
&\sum_{m=0}^{N} 2\,D_{im}\left(\tilde{F}^{\#}_{(i,m),j} + \tilde{G}^{\#}_{(i,m),j}\right)\\
&\quad= \sum_{m=0}^{N} 2\,D_{im}\left(\{\{(y_\eta)\,F\}\}_{(i,m),j} + \{\{(y_\eta)\,G\}\}_{(i,m),j}\right)\\
&\quad= \sum_{m=0}^{N} D_{im}\left(\left((y_\eta)_{i,j}\,F_{i,j} + (y_\eta)_{m,j}\,F_{m,j}\right)\right.\\
&\qquad\left. + \left((-x_\eta)_{i,j}\,G_{i,j} + (-x_\eta)_{m,j}\,G_{m,j}\right)\right)\\
&\quad= \sum_{m=0}^{N} D_{im}\left((y_\eta)_{m,j}\,F_{m,j} + (-x_\eta)_{m,j}\,G_{m,j}\right)\\
&\quad= \sum_{m=0}^{N} D_{im}\tilde{F}_{m,j}.
\end{aligned} \tag{4.43}$$

Before we explicitly state the DGSEM in flux differencing form, we note that due to the SBP property (4.8), the strong and weak form DG schemes are equivalent here. Thus, we are able to directly include the interior surface terms from the strong form into the volume sum. We modify the discrete derivative operator further to also incorporate the factor 2 of the flux differencing formulation. This leads to a modified derivative operator that simplifies the implementation and saves computational cost,

$$\tilde{\mathbf{D}} = 2\mathbf{D} + \mathbf{S}. \tag{4.44}$$

We are now ready to reformulate the DGSEM for the shallow water equations in flux differencing form and do so in the following Theorem.

Theorem 1 (Flux Differencing DGSEM)**.** *The DGSEM approximation to the shallow water equations on curvilinear meshes* (3.76) *can be written in flux differencing form by*

$$J\vec{W}_t + \vec{\mathcal{L}}_\xi + \vec{\mathcal{L}}_\eta = \vec{S} \tag{4.45}$$

with

$$\begin{aligned}\left(\vec{\mathcal{L}}_\xi\right)_{ij} &= \frac{1}{\omega_i}\left(\delta_{iN}\left[\vec{\tilde{F}}^*\right]_{Nj} - \delta_{i0}\left[\vec{\tilde{F}}^*\right]_{0j}\right) + \sum_{m=0}^{N} \tilde{D}_{im}\vec{\tilde{F}}_{(i,m),j},\\ \left(\vec{\mathcal{L}}_\eta\right)_{ij} &= \frac{1}{\omega_j}\left(\delta_{Nj}\left[\vec{\tilde{G}}^*\right]_{iN} - \delta_{0j}\left[\vec{\tilde{G}}^*\right]_{i0}\right) + \sum_{m=0}^{N} \tilde{D}_{mj}\vec{\tilde{G}}_{i,(m,j)},\end{aligned} \tag{4.46}$$

with curvilinear high-order numerical volume fluxes

$$\begin{aligned}\vec{\tilde{F}}_{(i,m),j} &:= \vec{\tilde{F}}^{\#}_{(i,m),j} + \vec{\tilde{G}}^{\#}_{(i,m),j},\\ \vec{\tilde{G}}_{i,(m,j)} &:= \vec{\tilde{F}}^{\#}_{i,(j,m)} + \vec{\tilde{G}}^{\#}_{i,(j,m)},\end{aligned} \tag{4.47}$$

where the numerical two-point fluxes are defined by

$$\begin{aligned}\vec{\tilde{F}}^{\#}_{(i,m),j} &= \left\{\!\!\left\{(y_\eta)\,\vec{\tilde{F}}\right\}\!\!\right\}_{(i,m),j},\\ \vec{\tilde{G}}^{\#}_{(i,m),j} &= \left\{\!\!\left\{(-x_\eta)\,\vec{\tilde{G}}\right\}\!\!\right\}_{(i,m),j},\\ \vec{\tilde{F}}^{\#}_{i,(j,m)} &= \left\{\!\!\left\{(-y_\xi)\,\vec{\tilde{F}}\right\}\!\!\right\}_{i,(j,m)},\\ \vec{\tilde{G}}^{\#}_{i,(j,m)} &= \left\{\!\!\left\{(x_\xi)\,\vec{\tilde{G}}\right\}\!\!\right\}_{i,(j,m)}.\end{aligned} \tag{4.48}$$

The numerical interface flux is the Lax-Friedrichs numerical flux (3.80)*. The discretization of the source term, $\vec{S}$, is left unchanged and is given by* (3.79)*.*

4.3. Equivalent split forms

With the compact form for the flux differencing formulation derived in the previous section, we now address other choices of the two-point fluxes $\vec{F}^{\#}$ and $\vec{G}^{\#}$. We have seen one choice in (4.48) that leads to the standard DGSEM in flux differencing form. In general, we are allowed to choose other two-point fluxes as long as they are symmetric and consistent to the physical (contravariant) fluxes. We will see that other choices can lead to discretizations of various split forms of the PDE. Numerical methods based on a split form of the PDE often show desirable properties such as increased robustness through de-aliasing or conservation of certain derived quantities such as kinetic energy or entropy [16, 56, 124, 256, 258]. The obtained properties depend on the specific type of split form that is used. If we are able to write a split form discretization in the telescoping flux difference forms (4.37) and (4.38), then global conservation of the approximation is guaranteed and the Lax-Wendroff Theorem holds. Thus, we are interested in split form schemes that can be expressed in flux differencing form such that we have the stabilization and robustness properties as well as global conservation in a Lax-Wendroff sense. These are remarkable and fundamental properties that allow us to construct the entropy stable numerical schemes in Chapter 5. Another advantage of the flux differencing formulation is that it greatly reduces the computational cost compared to a direct discretization of all the split form terms. Furthermore, we will see that split form discretizations are particularly useful for constructing well-balanced schemes for the shallow water equations.

We aim to make the connection between the choice of the two-point fluxes $F^{\#}$ and $G^{\#}$ and the resulting split form more clear. A split form for the derivative of a product of two quantities is

$$(a\,b)_x = \lambda\,(a\,b)_x + (1-\lambda)\,(a_x\,b + a\,b_x)\,, \qquad \lambda \in [0,1]. \tag{4.49}$$

We are interested in finding two-point fluxes, such that the telescoping flux difference sum, (4.24) in one dimension or (4.37) and (4.38) in two dimensions, are equivalent to a DG discretization of the split form. It is not trivial to find such two-point fluxes in general. For the special cases $\lambda \in \{0, \frac{1}{2}, 1\}$, the corresponding two-point fluxes are relatively simple. For $\lambda = 1$, we find that selecting the two-point flux

$$F^{\#}\,(a,b) = \{\{ab\}\}\,, \tag{4.50}$$

recovers the standard DG discretization. We show this by substituting (4.50) into the 1D compact flux differencing form (4.24) and observing

$$\begin{aligned}
\frac{1}{\omega_i}\sum_{m=0}^{N} 2\,Q_{im}\,\{\{ab\}\}_{(i,m)} &= \frac{1}{\omega_i}\sum_{m=0}^{N} Q_{im}\,((ab)_i + (ab)_m) \\
&= \frac{1}{\omega_i}\sum_{m=0}^{N} Q_{im}(ab)_m + \frac{1}{\omega_i}(ab)_i\sum_{m=0}^{N} Q_{im} \\
&= \frac{1}{\omega_i}\sum_{m=0}^{N} Q_{im}(ab)_m \\
&= \sum_{m=0}^{N} D_{im}(ab)_m.
\end{aligned} \tag{4.51}$$

This shows that the flux differencing scheme can be equivalent to a straight forward non-split form discretization when the two-point flux is simply the average of the product of two quantities. In the $\lambda = \frac{1}{2}$ split form case, once again a simple two-point flux is sufficient to obtain the DG split form discretization [82]. Instead of averaging the product of a and b, we take the product of the two individual averages,

$$F^{\#}(a,b) = \{\{a\}\}\,\{\{b\}\}\,, \tag{4.52}$$

to find the DG discretization of the split form (4.49) with $\lambda = \frac{1}{2}$,

$$\begin{aligned}\frac{1}{\omega_i}\sum_{m=0}^{N} 2\,Q_{im}\,\{\{a\}\}_{(i,m)}\,\{\{b\}\}_{(i,m)} &= \frac{1}{2\omega_i}\sum_{m=0}^{N} Q_{im}\,(a_i + a_m)\,(b_i + b_m)\\ &= \frac{1}{2\omega_i}\sum_{m=0}^{N} Q_{im}\,(a_i b_i + a_i b_m + a_m b_i + a_m b_m)\\ &= \frac{1}{2\omega_i}\sum_{m=0}^{N} Q_{im} a_m b_m + \frac{1}{2\omega_i} a_i \sum_{m=0}^{N} Q_{im} b_m + \frac{1}{2\omega_i} b_i \sum_{m=0}^{N} Q_{im} a_m\\ &= \frac{1}{2}\sum_{m=0}^{N} D_{im} a_m b_m + \frac{1}{2} a_i \sum_{m=0}^{N} D_{im} b_m + \frac{1}{2} b_i \sum_{m=0}^{N} D_{im} a_m.\end{aligned} \tag{4.53}$$

The third case we consider here, $\lambda = 0$ is slightly more involved. For the rather unintuitive choice

$$F^{\#}(a, b) = 2\,\{\{a\}\}\,\{\{b\}\} - \{\{ab\}\}\,, \tag{4.54}$$

we see that a discretization of the purely non-conservative terms is recovered,

$$\begin{aligned}&\frac{1}{\omega_i}\sum_{m=0}^{N} 2\,Q_{im}\left(2\,\{\{a\}\}_{(i,m)}\,\{\{b\}\}_{(i,m)} - \{\{ab\}\}_{(i,m)}\right)\\ &\quad= \frac{1}{\omega_i}\sum_{m=0}^{N} Q_{im}\,((a_i + a_m)\,(b_i + b_m) - (a_i b_i + a_m b_m))\\ &\quad= \frac{1}{\omega_i}\sum_{m=0}^{N} Q_{im}\,(a_i b_m + a_m b_i)\\ &\quad= a_i \sum_{m=0}^{N} D_{im} b_m + b_i \sum_{m=0}^{N} D_{im} a_m.\end{aligned} \tag{4.55}$$

It is worth mentioning that even though this discretization is clearly in non-conservative form, the numerical scheme is still conservative. This can be seen by reformulating the discretization from (4.55) as

$$\begin{aligned}&a_i \sum_{m=0}^{N} D_{im} b_m + b_i \sum_{m=0}^{N} D_{im} a_m\\ &\quad= \sum_{m=0}^{N} D_{im} a_m b_m - \sum_{m=0}^{N} D_{im} a_m b_m + a_i \sum_{m=0}^{N} D_{im} b_m + b_i \sum_{m=0}^{N} D_{im} a_m\\ &\quad= \sum_{m=0}^{N} D_{im} a_m b_m + \mathcal{E}_{ab},\end{aligned} \tag{4.56}$$

with non-linear discrete product rule error term

$$\mathcal{E}_{ab} := -\sum_{m=0}^{N} D_{im} a_m b_m + a_i \sum_{m=0}^{N} D_{im} b_m + b_i \sum_{m=0}^{N} D_{im} a_m. \tag{4.57}$$

Lemma 4 states that terms with this structure do not destroy the discrete conservation and the first term is a conservative approximation.

Practically relevant systems of conservation laws such as the shallow water equations, the Euler equations or the Navier-Stokes equations feature physical fluxes that are more than the simple product of two quantities. The more complicated (or non-linear) these fluxes are, the greater is the variety of possible split forms. In the case of the Euler equations, Kennedy and Gruber propose a split form for products of three quantities [124]. A discontinuous Galerkin discretization of these split forms can be recovered by using two-point fluxes that contain combinations of averaged products of quantities within the flux differencing framework. The Kennedy and Gruber flux is, for instance, recovered by the product of three individual averages [82]. In similar ways many popular split form discretizations can be obtained with flux differencing schemes by using appropriate two-point fluxes. However, it is important to point out, that not for all choices of two-point fluxes the equivalent split form is known. So, in some sense, the flux differencing framework covers all the direct split forms but even more. This highlights the flexibility of the flux differencing formulation (4.37). We refer to [82] for a more extensive study for split forms of the Euler equations.

We summarize the connections between the three important two-point fluxes consisting of one, two and three averages and the equivalent split form DG discretizations in a Lemma. We also introduce an abbreviation for the extended split form discretization to make the schemes discussed in the following Chapters more readable. The Lemma is formulated in one dimension, but we also give abbreviations for the two dimensional case afterwards.

Lemma 9 (Split form connections)**.** *A single average inserted into the flux differencing form is equivalent to a straight forward discretization of the derivative of the quantitity:*

$$\sum_{m=0}^{N} 2\,D_{im}\,\{\{a\}\}_{(i,m)} = \sum_{m=0}^{N} D_{im}a_m =: (D\circ a)_i\,. \tag{4.58}$$

Products of two or three averages are equivalent to discretizations of the original derivative in split form. The product of two averages recovers the quadratic split form discretization

$$\begin{aligned}&\sum_{m=0}^{N} 2\,D_{im}\,\{\{a\}\}_{(i,m)}\,\{\{b\}\}_{(i,m)}\\ &\quad=\frac{1}{2}\left(\sum_{m=0}^{N} D_{im}a_m b_m + a_i\sum_{m=0}^{N} D_{im}b_m + b_i\sum_{m=0}^{N} D_{im}a_m\right)\\ &\quad=: (D\circ a\circ b)_i\end{aligned} \tag{4.59}$$

And, lastly, the flux differencing formulation using the product of three averages is equiv-

alent to the discretization of the cubic splitting,

$$\begin{aligned}
&\sum_{m=0}^{N} 2\, D_{im} \{\{a\}\}_{(i,m)} \{\{b\}\}_{(i,m)} \{\{c\}\}_{(i,m)} \\
&\quad = \frac{1}{4}\left(\sum_{m=0}^{N} D_{im} a_m b_m c_m \right. \\
&\qquad + a_i \sum_{m=0}^{N} D_{im} b_m c_m + b_i \sum_{m=0}^{N} D_{im} a_m c_m + c_i \sum_{m=0}^{N} D_{im} a_m b_m \\
&\qquad \left. + b_i c_i \sum_{m=0}^{N} D_{im} a_m + a_i c_i \sum_{m=0}^{N} D_{im} b_m + a_i b_i \sum_{m=0}^{N} D_{im} c_m \right) \\
&\quad =: \left(D \circ a \circ b \circ c\right)_i .
\end{aligned} \tag{4.60}$$

These extensions hold for higher dimensions as well, since the averaging is always performed in only one of the spatial dimensions in a tensor product way.

Proof. All the extensions can be found by straight multiplication of the averages and the consistency of the derivative operator $\mathbf{D}$. ■

We now introduce similar abbreviated expressions for the two dimensional case. We perform derivatives in reference space and need to distinguish between derivatives in ξ and η direction.

Definition 2 (Notation for 2D product extensions). In the new notation, we abbreviate the conventional derivatives by

$$(D \circ a)^{\xi}_{ij} := \left(\sum_{m=0}^{N} D_{im} a_{mj} \right), \qquad (D \circ a)^{\eta}_{ij} := \left(\sum_{m=0}^{N} D_{jm} a_{im} \right). \tag{4.61}$$

The quadratic split form discretizations are equivalent to the flux differencing formulation of the product of two averages and abbreviated for each spatial direction by

$$\begin{aligned}
(D \circ a \circ b)^{\xi}_{ij} &:= \frac{1}{2}\left(\sum_{m=0}^{N} D_{im} a_{mj} b_{mj} + a_{ij} \sum_{m=0}^{N} D_{im} b_{mj} + b_{ij} \sum_{m=0}^{N} D_{im} a_{mj} \right), \\
(D \circ a \circ b)^{\eta}_{ij} &:= \frac{1}{2}\left(\sum_{m=0}^{N} D_{jm} a_{im} b_{im} + a_{ij} \sum_{m=0}^{N} D_{jm} b_{im} + b_{ij} \sum_{m=0}^{N} D_{jm} a_{im} \right).
\end{aligned} \tag{4.62}$$

Similarly, we write the extension of a derivative of a triple product as

$$\begin{aligned}
(D\circ a\circ b\circ c)^{\xi}_{ij} := \frac{1}{4}\Bigg(&\sum_{m=0}^{N} D_{im}a_{mj}b_{mj}c_{mj} \\
&+a_{ij}\sum_{m=0}^{N} D_{im}b_{mj}c_{mj} + b_{ij}\sum_{m=0}^{N} D_{im}a_{mj}c_{mj} + c_{ij}\sum_{m=0}^{N} D_{im}a_{mj}b_{mj} \\
&+b_{ij}c_{ij}\sum_{m=0}^{N} D_{im}a_{mj} + a_{ij}c_{ij}\sum_{m=0}^{N} D_{im}b_{mj} + a_{ij}b_{ij}\sum_{m=0}^{N} D_{im}c_{mj}\Bigg), \\
(D\circ a\circ b\circ c)^{\eta}_{ij} := \frac{1}{4}\Bigg(&\sum_{m=0}^{N} D_{jm}a_{im}b_{im}c_{im} \\
&+a_{ij}\sum_{m=0}^{N} D_{jm}b_{im}c_{im} + b_{ij}\sum_{m=0}^{N} D_{jm}a_{im}c_{im} + c_{ij}\sum_{m=0}^{N} D_{jm}a_{im}b_{im} \\
&+b_{ij}c_{ij}\sum_{m=0}^{N} D_{jm}a_{im} + a_{ij}c_{ij}\sum_{m=0}^{N} D_{jm}b_{im} + a_{ij}b_{ij}\sum_{m=0}^{N} D_{jm}c_{im}\Bigg).
\end{aligned} \tag{4.63}$$

When we introduced the two dimensional flux differencing method in Section 4.2.2, we only discussed the specific two-point fluxes that result in a numerical flux differencing scheme equivalent to the standard DGSEM. Now that we better understand the relationship between split forms and flux differencing schemes, we are able to address specific choices for the two-point fluxes and their equivalent split form discretizations.

Apart from the various split forms that can be derived based on the physical quantities present in the fluxes, there are also metric terms to consider on curvilinear meshes. In the DG methods considered in this work, these metric terms are approximated by polynomials from the same space as physical quantities and fluxes. The non-linearity that stems from the product of metric terms and physical quantities is subject to aliasing errors due to inexact integration as well. Thus, it is sensible to use a split form for this product as well. We call a split form for the metric terms a *geometric splitting* and refer to the increased robustness as *geometric de-aliasing.*

In (4.35) we stated that the metric terms should be treated as an additional input quantity for the two-point fluxes. Consequently, the entire discretization of the volume terms is determined by only the choice of numerical two-point fluxes. This allows the formal flexibility to use split forms for physical quantities and/or metric terms as to suit the desired purposes. The simplest choice to incorporate geometric de-aliasing is to average metric terms and the rest of the flux individually, e.g.,

$$\begin{aligned}
\tilde{F}^{\#}\left(W_{i,j}, W_{m,j}, (y_\eta)_{i,j}, (y_\eta)_{m,j}\right) &:= F^{\#}\left(W_{i,j}, W_{m,j}\right)\{\{y_\eta\}\}_{(i,m),j}, \\
\tilde{G}^{\#}\left(W_{i,j}, W_{m,j}, (-x_\eta)_{i,j}, (-x_\eta)_{m,j}\right) &:= G^{\#}\left(W_{i,j}, W_{m,j}\right)\{\{-x_\eta\}\}_{(i,m),j}, \\
\tilde{F}^{\#}\left(W_{i,j}, W_{i,m}, (-y_\xi)_{i,j}, (-y_\xi)_{i,m}\right) &:= F^{\#}\left(W_{i,j}, W_{i,m}\right)\{\{-y_\xi\}\}_{i,(j,m)}, \\
\tilde{G}^{\#}\left(W_{i,j}, W_{i,m}, (x_\xi)_{i,j}, (x_\xi)_{i,m}\right) &:= G^{\#}\left(W_{i,j}, W_{i,m}\right)\{\{x_\xi\}\}_{i,(j,m)}.
\end{aligned} \tag{4.64}$$

We show the equivalent split form of the metric term by choosing the formulation of the remaining flux part, to be

$$\begin{aligned}F^{\#}\left(W_{i,j}, W_{m,j}\right) &:= \{\{F\}\}_{(i,m),j}\,,\\ G^{\#}\left(W_{i,j}, W_{m,j}\right) &:= \{\{G\}\}_{(i,m),j}\,,\\ F^{\#}\left(W_{i,j}, W_{i,m}\right) &:= \{\{F\}\}_{i,(j,m)}\,,\\ G^{\#}\left(W_{i,j}, W_{i,m}\right) &:= \{\{G\}\}_{i,(j,m)}\,.\end{aligned} \tag{4.65}$$

Inserting (4.64) and (4.65) into the flux differencing forms (4.37) and (4.38) we determine the geometrically de-aliased scheme. Starting with the spatial operator in the ξ direction we find

$$\begin{aligned}&\sum_{m=0}^{N} 2\,D_{im}\left(\tilde{F}^{\#}_{(i,m),j} + \tilde{G}^{\#}_{(i,m),j}\right)\\ =&\sum_{m=0}^{N} 2\,D_{im}\left(\{\{(y_\eta)\}\}_{(i,m),j}\,\{\{F\}\}_{(i,m),j} + \{\{(-x_\eta)\}\}_{(i,m),j}\,\{\{G\}\}_{(i,m),j}\right)\\ =&\frac{1}{2}\left(y_\eta\right)_{i,j}\sum_{m=0}^{N} D_{im}F_{m,j} + \frac{1}{2}\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{m,j}F_{m,j}\\ &+\frac{1}{2}\left(-x_\eta\right)_{i,j}\sum_{m=0}^{N} D_{im}G_{m,j} + \frac{1}{2}\sum_{m=0}^{N} D_{im}\left(-x_\eta\right)_{m,j}G_{m,j}\\ &+\frac{1}{2}F_{i,j}\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{m,j} + \frac{1}{2}G_{i,j}\sum_{m=0}^{N} D_{im}\left(-x_\eta\right)_{m,j}\,.\end{aligned} \tag{4.66}$$

Similar steps in the η direction yield

$$\begin{aligned}&\sum_{m=0}^{N} 2\,D_{jm}\left(\tilde{F}^{\#}_{i,(j,m)} + \tilde{G}^{\#}_{i,(j,m)}\right)\\ =&\sum_{m=0}^{N} 2\,D_{jm}\left(\{\{(-y_\xi)\}\}_{i,(j,m)}\,\{\{F\}\}_{i,(j,m)} + \{\{(x_\xi)\}\}_{i,(j,m)}\,\{\{G\}\}_{i,(j,m)}\right)\\ =&\frac{1}{2}\left(-y_\xi\right)_{i,j}\sum_{m=0}^{N} D_{jm}F_{i,m} + \frac{1}{2}\sum_{m=0}^{N} D_{jm}\left(-y_\xi\right)_{i,m}F_{i,m}\\ &+\frac{1}{2}\left(x_\xi\right)_{i,j}\sum_{m=0}^{N} D_{jm}G_{i,m} + \frac{1}{2}\sum_{m=0}^{N} D_{jm}\left(x_\xi\right)_{i,m}G_{i,m}\\ &+\frac{1}{2}F_{i,j}\sum_{m=0}^{N} D_{jm}\left(-y_\xi\right)_{i,m} + \frac{1}{2}G_{i,j}\sum_{m=0}^{N} D_{jm}\left(x_\xi\right)_{i,m}\,.\end{aligned} \tag{4.67}$$

This shows, that by averaging the metric terms and the fluxes individually, we obtain a quadratic split form of the metric term. We can simplify expressions further by combining terms from the last two lines of each equation, (4.66) and (4.67), and applying the metric identities (3.63)

$$
\begin{aligned}
\frac{1}{2}F_{i,j}\left(\sum_{m=0}^{N} D_{im}\left(y_{\eta}\right)_{m,j} + \sum_{m=0}^{N} D_{jm}\left(-y_{\xi}\right)_{i,m}\right) &= 0,\\
\frac{1}{2}G_{i,j}\left(\sum_{m=0}^{N} D_{im}\left(-x_{\eta}\right)_{m,j} + \sum_{m=0}^{N} D_{jm}\left(x_{\xi}\right)_{i,m}\right) &= 0.
\end{aligned}
\tag{4.68}
$$

We now have established that the flux differencing formulation grants us additional flexibility to construct approximation to various split forms of the PDE. From such alternative split forms of the shallow water equations it is possible to create an entropy conservative numerical approximation [83]. This gives us the motivation to select the volume and surface fluxes in a specific way, such that the total energy of the numerical scheme will be conserved discretely. We note that the only alteration needed to change a standard DGSEM scheme to an entropy stable one is to use the flux differencing form for the volume integral and select appropriate two-point fluxes and entropy stable numerical interface fluxes for the surface contributions. The flux differencing representation guarantees that with a chosen pair of symmetric two-point fluxes the approximation (4.46) remains high-order accurate and globally conservative. Additionally, if the two-point fluxes in (4.48) and the surface fluxes in (4.46) are carefully constructed then the approximation is also provably entropy conservative [65]. Note that one can independently select an entropy conservative two-point flux that has a different form than an entropy conservative surface flux. We will show in the next chapter that this additional flexibility allows the construction of an entropy conservative approximation for the shallow water equations that is also well-balanced.

5. Entropy Stable DGSEM

In this chapter we present a novel entropy stable discontinuous Galerkin spectral element method (ESDGSEM) for the shallow water equations that is well-balanced and works on unstructured, possibly curved, quadrilateral meshes. We will present versions of the ESDGSEM for the one and two dimensional shallow water equations, both of which are based on a split form of the respective set of equations. The general strategy is to develop entropy conservative schemes first and then carefully add dissipation to the scheme if shocks are present to guarantee entropy stability. This is a popular approach in the construction of entropy stable methods, previously used by many researchers, e.g. [236, 117, 69].

The split form is chosen in a specific way to ensure element-local entropy conservation as well as a balance between the pressure and source terms. Through combination with entropy conserving numerical interface fluxes, the local entropy conservation is extended globally. To mimic the physical behaviour at shocks, we must introduce a mechanism of dissipating entropy. This is handled by adding numerical dissipation proportional to the jump in entropy variables to the numerical interface fluxes. For smooth solutions, this jump is small and the scheme is not very dissipative. In case of shocks, the dissipation is larger and helps to stabilize the simulation.

To ensure that the DG methods based on split forms are conservative schemes, we rely on the SBP property of the LGL operators as discussed in Chapter 4. In fact, to make the schemes computationally viable, we rewrite the split form discretizations in their equivalent flux differencing form from Section 4.2. A straightforward implementation of the split form discretization would lead to many distinct volume integrals for each momentum equation in the full two dimensional curvilinear case. However, with the flux differencing form, we can derive a compact expression for the ESDGSEM that needs just as many volume integrals as the standard DGSEM, at the cost of more expensive fluxes. A detailed discussion of the computational side of the ESDGSEM and suggestions for efficient implementations can be found in Chapter 6.

We start this chapter by deriving the entropy conservative DG method for the one dimensional shallow water equations in Section 5.1. The thorough discussion of this simpler case serves to highlight the key ideas in the development of the method without over-complicating the equations. The following Section 5.2 is dedicated to the derivation of the entropy conservative method in two spatial dimensions on curvilinear meshes. The two dimensional ECDGSEM is mostly built from the same ideas as the one dimensional method, but has some additional complexity due to the possibly curved geometry. After the entropy conservative methods are derived, we describe how numerical dissipation can be added to the numerical fluxes to find the entropy stable methods in Section 5.3.

We will show in the numerical results section, that entropy stable schemes immediately demonstrate increased robustness in challenging test cases. Entropy stability alone, however, does not render schemes oscillation-free. The entropy stable schemes are, by design, not highly dissipative. Thus, shocks occurring in high-order simulations can lead to quite severe oscillations and can possibly cause a scheme to crash. For the shallow water equations, particularly negative water heights are dangerous in a numerical code. They are not only unphysical, but can cause NaN-errors when computing the eigenvalues such as $\lambda = u + \sqrt{gh}$. We mitigate this issue and stabilize the scheme even further by introducing artificial viscosity to the equations. However, the variables used in the viscous gradient as well as their discretization must be chosen very carefully to maintain the entropy stability of the scheme. In [81], the authors provide a condition on the relation between gradient variables and entropy variables such that a discretization according to Bassi and Rebay (BR1, [9]) renders the artificial viscosity term entropy stable. We propose a suitable choice of gradient variables and a dynamic shock detection mechanism according to Persson and Peraire [202] in Section 5.4.

In the presence of (nearly) dry areas, even the smallest oscillations can render the numerical scheme unstable. Special mechanisms or limiters are required in such cases, to guarantee non-negative water heights. A widely used positivity preserving limiter was proposed by Zhang et al. [271]. This limiter is constructed from a linear scaling around element averages and is provably entropy stable. However, the theoretical proof for the positivity limiters relies on special assumptions on the numerical flux. In this context, the contribution of this work is the proof that the positivity limiter in combination with the entropy stable numerical flux of the ESDGSEM guarantees positive mean water heights under certain additional time step restrictions. The details of this are presented in Section 5.5.

5.1. ECDGSEM in one dimension

In this section we derive the one dimensional entropy conservative discontinuous Galerkin method originally proposed by Gassner et al. [83]. The method is based on a split form of the shallow water equations. First, we recall the one dimensional shallow water equations

$$\begin{aligned} h_t + (hu)_x &= 0 \\ (hu)_t + (hu^2 + \frac{1}{2}gh^2)_x &= -ghb_x. \end{aligned} \tag{5.1}$$

We choose the split form such that its LGL-DG discretization satisfies two key requirements:

- If contracted with the entropy variables, a discrete entropy equality is obtained,
- The pressure gradient $\frac{g}{2}(h^2)_x$ must be in balance with the source term ghb_x.

A discrete balance between pressure gradient and source term can be achieved by applying the product rule to the pressure gradient

$$\frac{g}{2}(h^2)_x = ghh_x. \tag{5.2}$$

The alternative form for $(hu^2)_x$ is less intuitive as it is obtained by averaging conservative form and terms obtained by applying the product rule with respect to u and hu

$$(hu^2)_x = \frac{1}{2}\Big((hu^2)_x + huu_x + u(hu)_x\Big)\,. \tag{5.3}$$

With the two newly derived expressions (5.2) and (5.3) we can formulate the split form shallow water system:

$$\begin{aligned} h_t + (hu)_x &= 0 \\ (hu)_t + \frac{1}{2}\Big((hu^2)_x + huu_x + u(hu)_x\Big) + gh(h+b)_x &= 0. \end{aligned} \tag{5.4}$$

We note that on the continuous level, both formulations are equivalent. Thus, the entropy function derived for the one dimensional shallow water equations in Lemma 1 is also an entropy function for the split form shallow water equations (5.4). Since the product rule does not hold discretely, both formulations are numerically different.

We use the split form shallow water equations (5.4) as a basis for the discontinuous Galerkin discretization. The bottom topography b is approximated by a polynomial from the same space as the conservative variables and fluxes. To allow for discontinuous bottom topographies, we introduce a special interface penalty term to the source term discretization.

$$g\,\left(h\,b_x\right)_i \approx g\;h_i\sum_{m=0}^{N} D_{im}b_m - \frac{g}{2}\frac{\delta_{i0}}{\omega_i}\left[\{\{h\}\}\,[\![b]\!]\right]_0 + \frac{g}{2}\frac{\delta_{iN}}{\omega_i}\left[\{\{h\}\}\,[\![b]\!]\right]_N\,, \tag{5.5}$$

where the jumps and averages are defined on element interfaces, e.g., for interior value a_i and exterior value a_o and outward pointing normal $\hat{n} = \pm 1$:

$$\begin{aligned} \{\{a\}\} &:= \frac{1}{2}\left(a_i + a_o\right), \\ [\![a]\!] &:= \left(a_o - a_i\right)\,\hat{n}. \end{aligned} \tag{5.6}$$

This discretization assumes that any discontinuities in the bottom topography align with element interfaces. Following the steps from Section 3.2.1 for all other terms individually, we find the strong form DGSEM discretization of the split form shallow water equations

$$\begin{aligned} J_i\left(h_t\right)_i &= -\sum_{m=0}^{N} D_{im}\left(hu\right)_m + \left(\frac{\delta_{i0}}{\omega_i}\left(F_1^* - hu\right)\big|_0 - \frac{\delta_{iN}}{\omega_i}\left(F_1^* - hu\right)\big|_N\right) \\ J_i\left((hu)_t\right)_i &= -\frac{1}{2}\left(\sum_{m=0}^{N} D_{im}\left(hu^2\right)_m + (hu)_i\sum_{m=0}^{N} D_{im}u_m + u_i\sum_{m=0}^{N} D_{im}\left(hu\right)_m\right) \\ &\quad - \frac{g}{2}h_i\sum_{m=0}^{N} D_{im}\left(h_m + b_m\right) \\ &\quad + \left(\frac{\delta_{i0}}{\omega_i}\left(F_2^* - (hu^2 + \frac{g}{2}h^2)\right)\big|_0 - \frac{\delta_{iN}}{\omega_i}\left(F_2^* - (hu^2 + \frac{g}{2}h^2)\right)\big|_N\right) \\ &\quad + \frac{g}{2}\frac{\delta_{i0}}{\omega_i}\left[\{\{h\}\}\,[\![b]\!]\right]_0 - \frac{g}{2}\frac{\delta_{iN}}{\omega_i}\left[\{\{h\}\}\,[\![b]\!]\right]_N\,. \end{aligned} \tag{5.7}$$

This discretization is only complete with an appropriate choice for the numerical interface flux F^*. We can construct this flux in a way such that the scheme becomes entropy conservative. Following the methodology by Tadmor [233], an entropy conservative numerical flux for the shallow water equations was proposed by Fjordholm [68]. In a locally conservative DG method the time evolution of the mean values can be rewritten as a finite volume method. Thus, entropy conservative fluxes for the finite volume scheme, are entropy conservative for DG as well. The entropy conservative flux by Fjordholm reads

$$\vec{F}^{*,ec}(\vec{W}^+,\vec{W}^-) := \begin{pmatrix} \{\{h\}\}\,\{\{u\}\} \\ \{\{h\}\}\,\{\{u\}\}^2 + \frac{g}{2}\,\{\{h^2\}\} \end{pmatrix}. \tag{5.8}$$

We will show that replacing the numerical fluxes in the split form shallow water discretization (5.7) with the entropy conservative fluxes (5.8) leads to an entropy conservative and well-balanced numerical scheme.

Lemma 10 (Entropy conservation of split form DG)**.** *The strong form discontinuous Galerkin approximation of the split form shallow water equations given in* (5.7) *is entropy conservative.*

Proof. We contract the equations of the strong form DG discretization (5.7) with the entropy variables $\vec{q} = (q_1, q_2)^T = \left(g(h+b) - \frac{1}{2}u^2, u\right)^T$ to derive the discrete entropy equation. We are only considering the semi-discrete scheme and assume continuity in time. Thus, it is allowed to use the product rule to recover the time derivative of the entropy, $e = \frac{1}{2}hu^2 + \frac{1}{2}gh^2 + ghb$, from the discrete time derivatives in (5.9). After canceling terms, we find the semi-discrete entropy equation to be

$$\begin{aligned}
J_i(e_t)_i &+ gh_i\sum_{m=0}^{N} D_{im}h_m u_m + gb_i\sum_{m=0}^{N} D_{im}h_m u_m \\
&+ \frac{1}{2}u_i\sum_{m=0}^{N} D_{im}h_m u_m^2 + \frac{1}{2}h_i u_i^2\sum_{m=0}^{N} D_{im}u_m + \frac{g}{2}h_i u_i\sum_{m=0}^{N} D_{im}\left(h_m + b_m\right) \\
&= \left(g(h_i+b_i) - \frac{1}{2}u_i^2\right)\left(\frac{\delta_{i0}}{\omega_i}\left[F_1^{*,ec} - hu\right]_0 - \frac{\delta_{iN}}{\omega_i}\left[F_1^{*,ec} - hu\right]_N\right) \\
&+ u_i\left(\frac{\delta_{i0}}{\omega_i}\left[F_2^{*,ec} - \left(hu^2 + \frac{g}{2}h^2\right)\right]_0 - \frac{\delta_{iN}}{\omega_i}\left[F_2^{*,ec} - \left(hu^2 + \frac{g}{2}h^2\right)\right]_N\right) \\
&+ u_i\left(\frac{g}{2}\frac{\delta_{i0}}{\omega_i}\left[\{\{h\}\}\,[\![b]\!]\right]_0 - \frac{g}{2}\frac{\delta_{iN}}{\omega_i}\left[\{\{h\}\}\,[\![b]\!]\right]_N\right).
\end{aligned} \tag{5.9}$$

To show discrete entropy conservation, we need to write spatial terms in (5.9) in conservative form. We show this in two steps: First, we show that the approximation to the derivative of the entropy flux is conservative. Then, we demonstrate that the interface contributions cancel.

We reformulate the volume terms in (5.9) to find a formulation in terms of the discrete derivative of the entropy flux and discrete non-linear product rule error terms,

$$\sum_{m=0}^{N} D_{im}\left(\frac{1}{2}h_m u_m^3 + g h_m u_m (h_m + b_m)\right) + \mathcal{E}\left(u, hu^2\right)_i + \mathcal{E}\left(h, hu\right)_i + \mathcal{E}\left(b, hu\right)_i, \quad (5.10)$$

with error terms given by

$$\begin{aligned}
\mathcal{E}\left(h, hu\right)_i &:= \frac{1}{2}\left(-\sum_{m=0}^{N} D_{im} h_m^2 u_m + h_i \sum_{m=0}^{N} D_{im} h_m u_m + h_i u_i \sum_{m=0}^{N} D_{im} h_m\right), \\
\mathcal{E}\left(b, hu\right)_i &:= g\left(-\sum_{m=0}^{N} D_{im} h_m u_m b_m + b_i \sum_{m=0}^{N} D_{im} h_m u_m + h_i u_i \sum_{m=0}^{N} D_{im} b_m\right), \quad (5.11) \\
\mathcal{E}\left(u, hu^2\right)_i &:= g\left(-\sum_{m=0}^{N} D_{im} h_m u_m^3 + u_i \sum_{m=0}^{N} D_{im} h_m u_m^2 + h_i u_i^2 \sum_{m=0}^{N} D_{im} u_m\right).
\end{aligned}$$

We know that the error terms in (5.11) are conservative from Lemma 4, since the LGL derivative operator $\mathbf{D}$ has the SBP property.

We recall the SBP property (4.8),

$$\mathbf{D} = \hat{\mathbf{D}} - \mathbf{S}, \qquad \mathbf{S} := \operatorname{diag}\left(\frac{1}{\omega_0}, 0, \ldots, 0, -\frac{1}{\omega_N}\right), \quad (5.12)$$

and apply it to the first term in (5.10) to find a conservative approximation to the entropy flux derivative plus additional surface terms

$$\begin{aligned}
&\sum_{m=0}^{N} \hat{D}_{im}\left(\frac{1}{2}h_m u_m^3 + g h_m u_m (h_m + b_m)\right) + \mathcal{E}\left(u, hu^2\right)_i + \mathcal{E}\left(h, hu\right)_i + \mathcal{E}\left(b, hu\right)_i \\
&\qquad + \left(\frac{\delta_{iN}}{\omega_N}\left[\frac{1}{2}hu^3 + ghu(h+b)\right]_N - \frac{\delta_{i0}}{\omega_0}\left[\frac{1}{2}hu^3 + ghu(h+b)\right]_0\right).
\end{aligned} \quad (5.13)$$

We now consider the original surface terms from (5.9) as well as the additional surface terms from (5.13) to find

$$\begin{aligned}
&\text{Interface terms for a single element} \\
&= \frac{\delta_{iN}}{\omega_i}\Big[(g(h+b) - \frac{1}{2}u^2)\left\{\!\!\left\{h\right\}\!\!\right\}\left\{\!\!\left\{u\right\}\!\!\right\} \\
&\qquad + u\left\{\!\!\left\{h\right\}\!\!\right\}\left\{\!\!\left\{u\right\}\!\!\right\}^2 + \frac{g}{2}u\left\{\!\!\left\{h^2\right\}\!\!\right\} - \frac{g}{2}h^2 u - \frac{g}{2}u\left\{\!\!\left\{h\right\}\!\!\right\}[\![b]\!]\Big]_N \quad (5.14) \\
&\quad - \frac{\delta_{i0}}{\omega_i}\Big[(g(h+b) - \frac{1}{2}u^2)\left\{\!\!\left\{h\right\}\!\!\right\}\left\{\!\!\left\{u\right\}\!\!\right\}\Big] \\
&\qquad + u\left\{\!\!\left\{h\right\}\!\!\right\}\left\{\!\!\left\{u\right\}\!\!\right\}^2 + \frac{g}{2}u\left\{\!\!\left\{h^2\right\}\!\!\right\} - \frac{g}{2}h^2 u - \frac{g}{2}u\left\{\!\!\left\{h\right\}\!\!\right\}[\![b]\!]\Big]_0.
\end{aligned}$$

The scheme is entropy conservative if the interface contribution in one element is exactly opposite to the interface contribution for its neighbor element. Thus, adding the interface contributions of two neighboring elements at a fixed arbitrary interface must be zero.

Interface terms are defined in direction of outward pointing normal vectors, here $\hat{n} = \pm 1$. Thus, at one fixed interface, the outward pointing normals of each element have opposing signs. Adding interface contributions of both elements leads to jump terms denoted by $[\![\cdot]\!]$ and defined in (5.6). The special source term discretization (5.5) already contains jumps with outward pointing orientation. Thus, adding these terms, $\frac{1}{2}u\,\{\{h\}\}\,[\![b]\!]$, from two neighbouring element, leads to an average of the velocity $\{\{u\}\}$ with fixed jump orientation. It follows, that the interface contribution of the source term discretization cancels with the bottom topography term from the entropy scaling.

Neglecting the scaling by the integration weight in (5.14), we find the following total interface contribution of one fixed interface

$$\begin{aligned}&\text{Total contribution at a fixed interface}\\ &= g\,\{\{h\}\}\,\{\{u\}\}\,[\![h]\!] + \frac{g}{2}\left\{\!\!\left\{h^2\right\}\!\!\right\}[\![u]\!] - g[\![h^2u]\!]\\ &\quad - \frac{1}{2}\,\{\{h\}\}\,\{\{u\}\}\,[\![u^2]\!] + \{\{h\}\}\,\{\{u\}\}^2\,[\![u]\!].\end{aligned} \tag{5.15}$$

Noting $\frac{1}{2}[\![a^2]\!] = \{\{a\}\}\,[\![a]\!]$, we observe

$$\frac{1}{2}\,\{\{h\}\}\,\{\{u\}\}\,[\![u^2]\!] = \{\{h\}\}\,\{\{u\}\}^2\,[\![u]\!], \tag{5.16}$$

and

$$\frac{g}{2}[\![h^2u]\!] = g\,\{\{h\}\}\,\{\{u\}\}\,[\![h]\!] + \frac{g}{2}\left\{\!\!\left\{h^2\right\}\!\!\right\}[\![u]\!], \tag{5.17}$$

and see that all terms in (5.15) vanish. Thus, we conclude that the DGSEM discretization of the split form shallow water equations with the entropy conservative numerical interface flux is entropy conservative. ■

The other requirement for the split form shallow water equations was, that the discretization is well-balanced. We prove this property in the following Lemma.

Lemma 11 (Well-balanced property)**.** *The strong form DG approximation of the split form shallow water equations given in* (5.7) *is well-balanced.*

Proof. A numerical scheme is well-balanced if the lake at rest initial conditions,

$$(h + b, hu) = (\text{const}, 0)\,,$$

are preserved for all time. We start with the discrete time derivative of the water height from (5.7)

$$J_i\,(h_t)_i = -\sum_{m=0}^{N} D_{im}\,(hu)_m + \left(\frac{\delta_{i0}}{\omega_i}\,\left(F_1^* - hu\right)\big|_0 - \frac{\delta_{iN}}{\omega_i}\,\left(F_1^* - hu\right)\big|_N\right). \tag{5.18}$$

Considering $F_1^{*,ec} = \{\{h\}\}\,\{\{u\}\}$ and the initial condition $u = 0$, it is clear that the discrete time derivative in (5.18) is exactly zero. Inserting $F_2^{*,ec} = \{\{h\}\}\,\{\{u\}\}^2 + \frac{g}{2}\,\{\{h^2\}\}$

into the momentum equation in (5.7) and canceling terms as $u = 0$, the remaining terms of the momentum equation are

$$\begin{aligned} J_i((hu)_t)_i &= -\frac{g}{2} h_i \sum_{m=0}^{N} D_{im}\left(h_m + b_m\right) \\ &+ \frac{\delta_{iN}}{\omega_i}\frac{g}{2}\left[h^2 - \left\{\left\{h^2\right\}\right\} - \{\{h\}\}\, [\![b]\!]\right]_N - \frac{\delta_{i0}}{\omega_i}\frac{g}{2}\left[h^2 - \left\{\left\{h^2\right\}\right\} - \{\{h\}\}\, [\![b]\!]\right]_0 . \end{aligned} \tag{5.19}$$

The volume term is zero due to the consistency of the derivative operator $\mathbf{D}$ and the total water height is $h + b = \text{const}$. Since we allow discontinuities in the bottom topography at element interfaces, it is possible that there is a jump in water height at the element interface. Thus, we cannot assume

$$\left\{\left\{h^2\right\}\right\} = h^2.$$

Instead, with definitions (5.6), we have at each interface

$$\left\{\left\{h^2\right\}\right\} - h^2 = \frac{h_i^2 + h_o^2}{2} - h_i^2 = \frac{h_o^2 - h_i^2}{2} = \frac{1}{2}[\![h^2]\!] = \{\{h\}\}\, [\![h]\!]\hat{n}, \tag{5.20}$$

where h_i denotes the inner and h_o the outer value. For each interface term in (5.19) we find

$$\left\{\left\{h^2\right\}\right\} - h^2 + \{\{h\}\}\, [\![b]\!] = \{\{h\}\}\, [\![h + b]\!] = 0. \tag{5.21}$$

We have shown that the time derivatives of both, the water height and the momentum are zero for the lake at rest initial conditions. This holds true for discontinuous bottom topographies, as long as the jumps align with element interfaces. Thus, we conclude that the split form DG is well-balanced. ■

The strong form DG split form discretization features four volume integrals for the momentum equation and is not in conservative form. One way to show that the scheme (5.7) remains a conservative approximation is by remembering that the derivative matrix $\mathbf{D}$ has the SBP property and recasting the formulation such that Lemma 4 can be applied. A simpler alternative is to find an equivalent flux differencing formulation. Schemes written in flux differencing form are immediately conservative as we have demonstrated in Section 4.2. Another benefit of the flux differencing formulation is the reduced computational complexity. The flux differencing formulation is described by the choice of the numerical two-point flux. In this case, we need to find a two-point flux $\vec{F}^{\#}$ that, when expanded, reproduces the split form discretizations in (5.7). We present such a two-point flux in Lemma 12.

Lemma 12 (Equivalent Discrete Split Form)**.** *If the following two-point flux is used for the volume integral of the 1D flux differencing DGSEM from Theorem 1,*

$$\vec{F}^{\#}(\vec{W}_i, \vec{W}_m) := \begin{pmatrix} \{\{hu\}\}_{(i,m)} \\ \{\{hu\}\}_{(i,m)}\, \{\{u\}\}_{(i,m)} + g\, \{\{h\}\}^2_{(i,m)} - \frac{1}{2}g\, \{\{h^2\}\}_{(i,m)} \end{pmatrix} \tag{5.22}$$

then it is equivalent to the strong form DGSEM discretization of the split form equations as in (5.7)*.*

Proof. We show the equivalence between the flux differencing formulation using the numerical two-point flux (5.22) and the direct discontinuous Galerkin discretization of the split form (5.4) given in (5.7) directly. To extend the flux differencing volume integral, we use the rules from Lemma 9 and the definition of $\tilde{\mathbf{D}}$ from (4.44). The surface modifications of the flux differencing operator $\tilde{\mathbf{D}}$ create the proper interior surface terms of the strong form DG discretization. This is true because the two-point flux evaluations on the interfaces, $\vec{F}^{\#}_{(0,0)}$ and $\vec{F}^{\#}_{(N,N)}$, are consistent to the physical fluxes.

We start with the continuity equation and recover the strong form DG terms by expanding the terms according to the rules from Lemma 9,

$$\begin{aligned}\sum_{m=0}^{N} \tilde{D}_{im} \left(F_1^{\#}\right)_{(i,m)} &= (D \circ hu)_i - \frac{\delta_{iN}}{\omega_N}\left[hu\right]_N + \frac{\delta_{i0}}{\omega_0}\left[hu\right]_0 \\ &= \sum_{m=0}^{N} D_{im}(hu)_m - \frac{\delta_{iN}}{\omega_N}\left[hu\right]_N + \frac{\delta_{i0}}{\omega_0}\left[hu\right]_0 .\end{aligned} \tag{5.23}$$

A direct expansion of the momentum equation terms according to Lemma 9 leads to

$$\begin{aligned}\sum_{m=0}^{N} \tilde{D}_{im} \left(F_2^{\#}\right)_{(i,m)} =& \,(D \circ hu \circ u)_i + g\,(D \circ h \circ h)_i - \frac{g}{2}\left(D \circ h^2\right)_i \\ & - \frac{\delta_{iN}}{\omega_N}\left[\left(hu^2 + \frac{g}{2}h^2\right)\right]_N + \frac{\delta_{i0}}{\omega_0}\left[\left(hu^2 + \frac{g}{2}h^2\right)\right]_0 .\end{aligned} \tag{5.24}$$

A closer investigation of the pressure terms yields

$$g\,(D \circ h \circ h)_i - \frac{g}{2}\left(D \circ h^2\right)_i = gh_i \sum_{m=0}^{N} D_{im} h_m, \tag{5.25}$$

which is exactly the discretization of the split form we constructed for the pressure term in (5.4). Expanding the double product $(D \circ hu \circ u)_i$ and inserting (5.25) into (5.24), we recover the interior parts of the strong form DG discretization (5.7),

$$\begin{aligned}&\sum_{m=0}^{N} \tilde{D}_{im} \left(F_2^{\#}\right)_{(i,m)} \\ &= \frac{1}{2}\left(\sum_{m=0}^{N} D_{im}(hu^2)_m + u_i \sum_{m=0}^{N} D_{im}(hu)_m + (hu)_i \sum_{m=0}^{N} D_{im}(u)_m\right) \\ &\quad + gh_i \sum_{m=0}^{N} D_{im} h_m - \frac{\delta_{iN}}{\omega_N}\left[\left(hu^2 + \frac{g}{2}h^2\right)\right]_N + \frac{\delta_{i0}}{\omega_0}\left[\left(hu^2 + \frac{g}{2}h^2\right)\right]_0 .\end{aligned} \tag{5.26}$$

Since the source term and remaining surface terms are identical in the flux differencing formulation, we have shown that the flux differencing scheme with numerical two-point flux as in (5.22) is equivalent to a direct strong form DG discretization of the split form shallow water equations as in (5.7). ■

We have derived a conservative, well-balanced discontinuous Galerkin scheme for the one dimensional shallow water equations that is entropy conservative in a semi-discrete sense. We summarize the scheme and its key properties in the following Theorem.

Theorem 2 (ECDGSEM - 1D)**.** *The semi-discrete strong form DG approximation to the one dimensional split form shallow water equations* (5.4) *in flux differencing formulation is given by equations*

$$J\vec{W}_t + \vec{\mathcal{L}}_\xi = \vec{S} \tag{5.27}$$

with spatial operator

$$\left(\vec{\mathcal{L}}_\xi\right)_i = \frac{1}{\omega_i}\left(\delta_{iN}\left[\vec{F}^{*,ec}\right]_N - \delta_{i0}\left[\vec{F}^{*,ec}\right]_0\right) + \sum_{m=0}^{N} \tilde{D}_{im}\vec{F}^{\#}_{(i,m)}, \tag{5.28}$$

where the entropy conservative two-point flux in the volume integral is defined by (5.22)*. The numerical interface flux is given by*

$$\vec{F}^{*,ec}(\vec{W}^+,\vec{W}^-) := \begin{pmatrix} \{\{h\}\}\,\{\{u\}\} \\ \{\{h\}\}\,\{\{u\}\}^2 + \frac{g}{2}\,\{\{h^2\}\} \end{pmatrix}. \tag{5.29}$$

The source term of the momentum equation is discretized by

$$(s_2)_i = g\ h_i \sum_{m=0}^{N} D_{im} b_m - \frac{g}{2}\frac{\delta_{i0}}{\omega_i}\left[\{\{h\}\}\,[\![b]\!]\right]_0 + \frac{g}{2}\frac{\delta_{iN}}{\omega_i}\left[\{\{h\}\}\,[\![b]\!]\right]_N. \tag{5.30}$$

The scheme described above is called the ECDGSEM and has the following properties:

12.1 Discrete conservation of the mass and discrete conservation of the momentum if the bottom topography is constant.

12.2 Discrete conservation of the total energy, which is an entropy function for the shallow water equations. Hence it fulfills the discrete entropy equality (2.52)*.*

12.3 Discrete well-balanced property for arbitrary bottom topographies as long as any discontinuities align with element interfaces.

Proof.

Proof of Part 12.1: The conservation of water height follows immediately from the conservative flux differencing form of the scheme. As long as the bottom topography is constant, the source term is exactly zero and the scheme is conservative by the same argument.

Proof of Part 12.2: The entropy conservation was proven in Lemma 10 for the DG scheme based on the split form with the entropy conserving numerical flux. The proposed flux differencing scheme is equivalent to the split form DG discretization as proven in Lemma 12.

Proof of Part 12.3: In Lemma 11 we have shown that the equivalent split form DG is well-balanced.

■

5.2. ECDGSEM in two dimensions

Now that we have demonstrated the development of the entropy conservative discontinuous Galerkin method for the one dimensional shallow water equations, we proceed to address the general two dimensional case. The major difference and difficulty lies in the increased geometric complexity on possibly curved meshes. The two dimensional mapping from physical space to the reference element leads to four distinct metric terms and the Jacobian at each node. As discussed in Section 3.2.2, the metric terms are approximated by polynomials from the same space as the conservative variables. In consequence, the contravariant fluxes (3.52) potentially increase the aliasing error of the numerical quadrature. To mitigate this effect, we introduce an additional geometric split form. This comes at the cost of computational complexity, as the split form shallow water equations with metric term splitting include many distinct terms, each requiring an individual volume integral computation if discretized directly.

Following the same steps as in one dimension, we first introduce a split form of the shallow water equations. Then, we show that a direct DGSEM discretization of this system is entropy conservative and well-balanced if appropriate numerical fluxes are chosen on the interfaces. The entropy conservative numerical fluxes are found by following the ideas of Tadmor and Fjordholm [233, 235, 69]. We, again, find an equivalent flux differencing formulation to prove that the method is conservative and to make it computationally viable.

The first step in the derivation of the entropy conservative method is to find the specific split form of the 2D shallow water equations (2.34) that we want to use as the basis of our scheme. Since we are interested in de-aliasing the metric terms as well, we immediately consider the (continuous) equations in reference space, which are found to be

$$\mathcal{J}\vec{w}_t + \vec{\tilde{f}}_\xi + \vec{\tilde{g}}_\eta = \vec{\tilde{s}} \tag{5.31}$$

with contravariant fluxes

$$\vec{\tilde{f}} = \begin{pmatrix} y_\eta hu - x_\eta hv \\ y_\eta hu^2 + \frac{1}{2}gy_\eta h^2 - x_\eta huv \\ y_\eta huv - x_\eta hv^2 - \frac{1}{2}gx_\eta h^2 \end{pmatrix}, \qquad \vec{\tilde{g}} = \begin{pmatrix} -y_\xi hu^2 - \frac{1}{2}gy_\xi h^2 + x_\xi huv \\ -y_\xi hu^2 - \frac{1}{2}gy_\xi h^2 + x_\xi huv \\ -y_\xi huv + x_\xi hv^2 + \frac{1}{2}gx_\xi h^2 \end{pmatrix}, \tag{5.32}$$

and curvilinear source term

$$\vec{\tilde{s}} = -gh \begin{pmatrix} 0 \\ (y_\eta b)_\xi - (y_\xi b)_\eta \\ (-x_\eta b)_\xi + (x_\xi b)_\eta \end{pmatrix}. \tag{5.33}$$

We are interested in deriving a split form of these equations, such that a DGSEM discretization of that split form is entropy conservative and well-balanced. Contrary to the one dimensional case, we now reformulate the continuity equation in split form as well to introduce a geometric splitting in the metric terms. We derive the split forms of each equation separately and sort terms according to the metric terms. The split form continuity equation is only in split form with regards to the metric terms and reads

$$\begin{aligned}\mathcal{J}h_t &+ \frac{1}{2}\Big[(y_\eta hu)_\xi + hu\,(y_\eta)_\xi + y_\eta\,(hu)_\xi\Big] - \frac{1}{2}\Big[(x_\eta hv)_\xi + x_\eta\,(hv)_\xi + hv\,(x_\eta)_\xi\Big] \\ &- \frac{1}{2}\Big[(y_\xi hu)_\eta + y_\xi (hu)_\eta + hu(y_\xi)_\eta\Big] + \frac{1}{2}\Big[(x_\xi hv)_\eta + x_\xi (hv)_\eta + hv(x_\xi)_\eta\Big] = 0.\end{aligned} \tag{5.34}$$

To find the split form of the momentum equations we use a similar strategy as in one dimension. Again, we choose a different splitting for the pressure term $\frac{g}{2}h^2$ and the velocity terms hu^2, hv^2 and huv. The physical quantities of the pressure term are split as before, but we introduce an additional metric term splitting, e.g.,

$$\frac{g}{2}(y_\eta h^2)_\xi = \frac{g}{2}h\left((y_\eta h)_\xi + y_\eta h_\xi\right). \tag{5.35}$$

The split form representations of the velocity terms are also similar to the one dimensional variant. The only difference is that each of the terms receives an additional geometric splitting, e.g.,

$$\begin{aligned}(y_\eta hu^2)_\xi = \frac{1}{4}\Big[&\left(y_\eta hu^2\right)_\xi + y_\eta \left(hu^2\right)_\xi + hu^2\,(y_\eta)_\xi + hu\,(y_\eta u)_\xi \\ &+ y_\eta u\,(hu)_\xi + u\,(y_\eta hu)_\xi + y_\eta hu\,(u)_\xi\Big].\end{aligned} \tag{5.36}$$

Applying similar split formulations to the other velocity terms hv^2 and huv as well as for the other metric terms leads to a full split formulation of the hu momentum equation

$$\begin{aligned}\mathcal{J}(hu)_t &+ \frac{1}{4}\Big[\left(y_\eta hu^2\right)_\xi + y_\eta\left(hu^2\right)_\xi + hu^2\,(y_\eta)_\xi + hu\,(y_\eta u)_\xi + y_\eta u\,(hu)_\xi + u\,(y_\eta hu)_\xi + y_\eta hu\,(u)_\xi\Big] \\ &+ \frac{1}{2}g\Big[h\,(y_\eta h)_\xi + y_\eta h\,(h)_\xi\Big] \\ &- \frac{1}{4}\Big[(x_\eta huv)_\xi + x_\eta\,(huv)_\xi + huv\,(x_\eta)_\xi + hv\,(x_\eta u)_\xi + x_\eta u\,(hv)_\xi + u\,(x_\eta hv)_\xi + x_\eta hv\,(u)_\xi\Big] \\ &- \frac{1}{4}\Big[\left(y_\xi hu^2\right)_\eta + y_\xi\left(hu^2\right)_\eta + hu^2\,(y_\xi)_\eta + hu\,(y_\xi u)_\eta + y_\xi u\,(hu)_\eta + u\,(y_\xi hu)_\eta + y_\xi hu\,(u)_\eta\Big] \\ &- \frac{1}{2}g\Big[h\,(y_\xi h)_\eta + y_\xi h\,(h)_\eta\Big] \\ &+ \frac{1}{4}\Big[(x_\xi huv)_\eta + x_\xi\,(huv)_\eta + huv\,(x_\xi)_\eta + hv\,(x_\xi u)_\eta + x_\xi u\,(hv)_\eta + u\,(x_\xi hv)_\eta + x_\xi hv\,(u)_\eta\Big] \\ &= \tilde{s}_2.\end{aligned} \tag{5.37}$$

The split form of the hv momentum equation is found analogously,

$$\begin{aligned}
&\mathcal{J}(hv)_t + \frac{1}{4}\left[(y_\eta huv)_\xi + y_\eta\,(huv)_\xi + huv\,(y_\eta)_\xi + hu\,(y_\eta v)_\xi + y_\eta v\,(hu)_\xi + v\,(y_\eta hu)_\xi + y_\eta hu\,(v)_\xi\right]\\
&\quad - \frac{1}{4}\left[\left(x_\eta hv^2\right)_\xi + x_\eta\left(hv^2\right)_\xi + hv^2\,(x_\eta)_\xi + hv\,(x_\eta v)_\xi + x_\eta v\,(hv)_\xi + v\,(x_\eta hv)_\xi + x_\eta hv\,(v)_\xi\right]\\
&\quad - \frac{1}{2}g\left[h\,(x_\eta h)_\xi + x_\eta h\,(h)_\xi\right]\\
&\quad - \frac{1}{4}\left[(y_\xi huv)_\eta + y_\xi\,(huv)_\eta + huv\,(y_\xi)_\eta + hu\,(y_\xi v)_\eta + y_\xi v\,(hu)_\eta + v\,(y_\xi hu)_\eta + y_\xi hu^2\,(v)_\eta\right]\\
&\quad + \frac{1}{4}\left[\left(x_\xi hv^2\right)_\eta + x_\xi\left(hv^2\right)_\eta + hv^2\,(x_\xi)_\eta + hv\,(x_\xi v)_\eta + x_\xi v\,(hv)_\eta + v\,(x_\xi hv)_\eta + x_\xi hv\,(v)_\eta\right]\\
&\quad + \frac{1}{2}g\left[h\,(x_\xi h)_\eta + x_\xi h\,(h)_\eta\right]\\
&\quad = \tilde{s}_3.
\end{aligned} \tag{5.38}$$

We use a geometric splitting for the source terms in (5.37) and (5.38), as well. This helps with de-aliasing but is also necessary for the well-balanced property: The pressure term and the source term must both have the same geometric splitting. The split form source terms are

$$\begin{aligned}
\tilde{s}_2 &= -gh\left[(y_\eta b)_\xi - (y_\xi b)_\eta\right]\\
&= -\frac{1}{2}gh\left[(y_\eta b)_\xi + y_\eta b_\xi + b\,(y_\eta)_\xi - (y_\xi b)_\eta - y_\xi\,(b)_\eta - b\,(y_\xi)_\eta\right]\\
&= -\frac{1}{2}gh\left[(y_\eta b)_\xi + y_\eta b_\xi - (y_\xi b)_\eta - y_\xi\,(b)_\eta\right],\\
\tilde{s}_3 &= -\frac{1}{2}gh\left[-(x_\eta b)_\xi - x_\eta b_\xi + (x_\xi b)_\eta + x_\xi\,(b)_\eta\right],
\end{aligned} \tag{5.39}$$

where we assume that the metric terms are sufficiently smooth such that

$$\begin{aligned}
(y_\eta)_\xi - (y_\xi)_\eta &= 0,\\
-(x_\eta)_\xi + (x_\xi)_\eta &= 0.
\end{aligned} \tag{5.40}$$

This is only valid discretely if the metric identities (3.61) are fulfilled. The split form shallow water equations given by (5.42), (5.46) and (5.47) and the split form of the source term (5.39) are the basis for the discretization. To make the discretized equations slightly more readable we will use the compact expressions for split form discretizations introduced in Definition 2.

The source term discretization is extended by interface penalty to allow for discontinuous bottom topographies at element interfaces. This specific form matches the rest of the discretization such that the scheme remains entropy conservative in the case of such discontinuities. Together with the usual DGSEM discretization with LGL nodes for the volume terms, the discrete version of the source term in split form (5.39) is then given by

$$\begin{aligned}
(\tilde{s}_2)_{ij} :=& -gh_{ij}\left((D\circ y_\eta \circ b)^{\xi}_{ij} - (D\circ y_\xi \circ b)^{\eta}_{ij}\right)\\
&- \frac{\delta_{iN}}{\omega_i}\left[\frac{g}{2}y_\eta \{\{h\}\}[\![b]\!]\right]_{Nj} - \frac{\delta_{i0}}{\omega_i}\left[\frac{g}{2}y_\eta \{\{h\}\}[\![b]\!]\right]_{0j}\\
&+ \frac{\delta_{Nj}}{\omega_j}\left[\frac{g}{2}y_\xi \{\{h\}\}[\![b]\!]\right]_{iN} + \frac{\delta_{0j}}{\omega_j}\left[\frac{g}{2}y_\xi \{\{h\}\}[\![b]\!]\right]_{i0},\\
(\tilde{s}_3)_{ij} :=& -gh_{ij}\left(-(D\circ x_\eta \circ b)^{\xi}_{ij} + (D\circ x_\xi \circ b)^{\eta}_{ij}\right)\\
&+ \frac{\delta_{iN}}{\omega_i}\left[\frac{g}{2}x_\eta \{\{h\}\}[\![b]\!]\right]_{Nj} + \frac{\delta_{i0}}{\omega_i}\left[\frac{g}{2}x_\eta \{\{h\}\}[\![b]\!]\right]_{0j}\\
&- \frac{\delta_{Nj}}{\omega_j}\left[\frac{g}{2}x_\xi \{\{h\}\}[\![b]\!]\right]_{iN} - \frac{\delta_{0j}}{\omega_j}\left[\frac{g}{2}x_\xi \{\{h\}\}[\![b]\!]\right]_{i0}.
\end{aligned} \tag{5.41}$$

We follow the steps described in Section 3.2.2 to find the semi-discrete form of the continuity equation (5.34)

$$\begin{aligned}
J_{ij}(h_t)_{ij} =& -(D\circ y_\eta\circ hu)^{\xi}_{ij} + (D\circ x_\eta\circ hv)^{\xi}_{ij} + (D\circ y_\xi\circ hu)^{\eta}_{ij} - (D\circ x_\xi\circ hv)^{\eta}_{ij}\\
&- \frac{\delta_{iN}}{\omega_i}\left([y_\eta(F_1^* - hu)]_{Nj} - [x_\eta(G_1^* - hv)]_{Nj}\right)\\
&+ \frac{\delta_{i0}}{\omega_i}\left([y_\eta(F_1^* - hu)]_{0j} - [x_\eta(G_1^* - hv)]_{0j}\right)\\
&+ \frac{\delta_{Nj}}{\omega_j}\left([y_\xi(F_1^* - hu)]_{iN} - [x_\xi(G_1^* - hv)]_{iN}\right)\\
&- \frac{\delta_{0j}}{\omega_j}\left([y_\xi(F_1^* - hu)]_{i0} - [x_\xi(G_1^* - hv)]_{i0}\right).
\end{aligned} \tag{5.42}$$

Similarly we find the semi-discrete strong form DG discretizations of the momentum equations. To write the pressure term in compact form, it is necessary that the metric identities hold. We see this by considering the discretization of the split form pressure terms, e.g. $\frac{1}{2}g\left[h\,(y_\eta h)_\xi + y_\eta h\,(h)_\xi\right]$, from (5.37) and rewriting as

$$\begin{aligned}
&\frac{g}{2}h_{ij}\left(\sum_{m=0}^{N} D_{im}\,(y_\eta)_{mj}\,h_{mj} + (y_\eta)_{ij}\sum_{m=0}^{N} D_{im}h_{mj}\right)\\
=&\frac{g}{2}h_{ij}\,(D\circ y_\eta\circ h)^{\xi}_{ij} - \frac{g}{2}h_{ij}^2\left(\sum_{m=0}^{N} D_{im}\,(y_\eta)_{mj}\right).
\end{aligned} \tag{5.43}$$

Following the same steps for $-\frac{1}{2}g\left[h\,(y_\xi h)_\eta + y_\xi h\,(h)_\eta\right]$, we find

$$\begin{aligned}
&-\frac{g}{2}h_{ij}\left(\sum_{m=0}^{N} D_{jm}\,(y_\xi)_{im}\,h_{im} + (y_\xi)_{ij}\sum_{m=0}^{N} D_{jm}h_{im}\right)\\
=&-\frac{g}{2}h_{ij}\,(D\circ y_\xi\circ h)^{\eta}_{ij} + \frac{g}{2}h_{ij}^2\left(\sum_{m=0}^{N} D_{jm}\,(y_\xi)_{im}\right).
\end{aligned} \tag{5.44}$$

We consider the final terms in (5.43) and (5.44) together to find

$$-\frac{g}{2}h_{ij}^2\left(\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj}\right)+\frac{g}{2}h_{ij}^2\left(\sum_{m=0}^{N} D_{jm}\left(y_\xi\right)_{im}\right)=0, \tag{5.45}$$

due to the metric identities (3.63). Thus, we can use the compact split form notation (4.62) to write the momentum equations as

$$\begin{aligned}
J_{ij}\left((hu)_t\right)_{ij} = &-\left(D\circ y_\eta\circ hu\circ u\right)_{ij}^{\xi}+\left(D\circ x_\eta\circ hv\circ u\right)_{ij}^{\xi}-gh_{ij}\left(\left(D\circ y_\eta\circ h\right)_{ij}^{\xi}+\left(D\circ y_\eta\circ b\right)_{ij}^{\xi}\right)\\
&+\left(D\circ y_\xi\circ hu\circ u\right)_{ij}^{\eta}-\left(D\circ x_\xi\circ hv\circ u\right)_{ij}^{\eta}+gh_{ij}\left(\left(D\circ y_\xi\circ h\right)_{ij}^{\eta}+\left(D\circ y_\xi\circ b\right)_{ij}^{\eta}\right)\\
&-\frac{\delta_{iN}}{\omega_i}\left(\left[y_\eta\left(F_2^*-hu^2-\frac{g}{2}h^2\right)\right]_{Nj}-\left[x_\eta\left(G_2^*-huv\right)\right]_{Nj}+\frac{g}{2}\left[y_\eta\left\{\left\{h\right\}\right\}[\![b]\!]\right]_{Nj}\right)\\
&+\frac{\delta_{i0}}{\omega_i}\left(\left[y_\eta\left(F_2^*-hu^2-\frac{g}{2}h^2\right)\right]_{0j}-\left[x_\eta\left(G_2^*-huv\right)\right]_{0j}-\frac{g}{2}\left[y_\eta\left\{\left\{h\right\}\right\}[\![b]\!]\right]_{0j}\right)\\
&+\frac{\delta_{Nj}}{\omega_j}\left(\left[y_\xi\left(F_2^*-hu^2-\frac{g}{2}h^2\right)\right]_{iN}-\left[x_\xi\left(G_2^*-huv\right)\right]_{iN}+\frac{g}{2}\left[y_\xi\left\{\left\{h\right\}\right\}[\![b]\!]\right]_{iN}\right)\\
&-\frac{\delta_{0j}}{\omega_j}\left(\left[y_\xi\left(F_2^*-hu^2-\frac{g}{2}h^2\right)\right]_{i0}-\left[x_\xi\left(G_2^*-huv\right)\right]_{i0}-\frac{g}{2}\left[y_\xi\left\{\left\{h\right\}\right\}[\![b]\!]\right]_{i0}\right),
\end{aligned} \tag{5.46}$$

and

$$\begin{aligned}
J_{ij}\left((hv)_t\right)_{ij} = &-\left(D\circ y_\eta\circ hu\circ v\right)_{ij}^{\xi}+\left(D\circ x_\eta\circ hv\circ v\right)_{ij}^{\xi}+gh_{ij}\left(D\circ x_\eta\circ h\right)_{ij}^{\xi}+gh_{ij}\left(D\circ x_\eta\circ b\right)_{ij}^{\xi}\\
&+\left(D\circ y_\xi\circ hu\circ v\right)_{ij}^{\eta}-\left(D\circ x_\xi\circ hv\circ v\right)_{ij}^{\eta}-gh_{ij}\left(D\circ x_\xi\circ h\right)_{ij}^{\eta}-gh_{ij}\left(D\circ x_\xi\circ b\right)_{ij}^{\eta}\\
&-\frac{\delta_{iN}}{\omega_i}\left(\left[y_\eta\left(F_3^*-huv\right)\right]_{Nj}-\left[x_\eta\left(G_3^*-hv^2-\frac{g}{2}h^2\right)\right]_{Nj}-\left[\frac{g}{2}x_\eta\left\{\left\{h\right\}\right\}[\![b]\!]\right]_{Nj}\right)\\
&+\frac{\delta_{i0}}{\omega_i}\left(\left[y_\eta\left(F_3^*-huv\right)\right]_{0j}-\left[x_\eta\left(G_3^*-hv^2-\frac{g}{2}h^2\right)\right]_{0j}+\left[\frac{g}{2}x_\eta\left\{\left\{h\right\}\right\}[\![b]\!]\right]_{0j}\right)\\
&+\frac{\delta_{Nj}}{\omega_j}\left(\left[y_\xi\left(F_3^*-huv\right)\right]_{iN}-\left[x_\xi\left(G_3^*-hv^2-\frac{g}{2}h^2\right)\right]_{iN}-\left[\frac{g}{2}x_\xi\left\{\left\{h\right\}\right\}[\![b]\!]\right]_{iN}\right)\\
&-\frac{\delta_{0j}}{\omega_j}\left(\left[y_\xi\left(F_3^*-huv\right)\right]_{i0}-\left[x_\xi\left(G_3^*-hv^2-\frac{g}{2}h^2\right)\right]_{i0}+\left[\frac{g}{2}x_\xi\left\{\left\{h\right\}\right\}[\![b]\!]\right]_{i0}\right).
\end{aligned} \tag{5.47}$$

To make the discretizations (5.42), (5.46) and (5.47) complete, we need to define the numerical interface fluxes $\vec{F}^*$ and $\vec{G}^*$. An entropy conservative numerical flux can be derived from the two dimensional version of the entropy condition (3.82) by Tadmor [236], which is given by

$$\langle[\![\vec{q}]\!],\vec{\vec{F}}^*+\vec{\vec{G}}^*\rangle=[\![\Psi]\!]+\langle\left\{\left\{\vec{q}\right\}\right\},\vec{\vec{S}}\,\rangle. \tag{5.48}$$

We note that the interface part of the source term discretization and the numerical flux must be chosen as a matching pair in order for (5.48) to hold. Numerical fluxes that satisfy condition (5.48) for the source term discretization (5.41) are based on the following entropy conservative numerical fluxes,

$$
\begin{aligned}
\vec{F}^{*,ec}(\vec{W}^+,\vec{W}^-) &:= \begin{pmatrix} \{\{h\}\}\,\{\{u\}\} \\ \{\{h\}\}\,\{\{u\}\}^2 + \frac{g}{2}\,\{\{h^2\}\} \\ \{\{h\}\}\,\{\{u\}\}\,\{\{v\}\} \end{pmatrix}, \\
\vec{G}^{*,ec}(\vec{W}^+,\vec{W}^-) &:= \begin{pmatrix} \{\{h\}\}\,\{\{v\}\} \\ \{\{h\}\}\,\{\{u\}\}\,\{\{v\}\} \\ \{\{h\}\}\,\{\{v\}\}^2 + \frac{g}{2}\,\{\{h^2\}\} \end{pmatrix},
\end{aligned}
\tag{5.49}
$$

which are combined to find the entropy conservative contravariant numerical fluxes:

$$
\begin{aligned}
\tilde{\vec{F}}^{*,ec} &= y_\eta \vec{F}^{*,ec} - x_\eta \vec{G}^{*,ec} \\
\tilde{\vec{G}}^{*,ec} &= -y_\xi \vec{F}^{*,ec} + x_\xi \vec{G}^{*,ec}.
\end{aligned}
\tag{5.50}
$$

We will prove that the DG discretization of the split form shallow water equations given in (5.42), (5.46) and (5.47) with source term discretization (5.41) and numerical interface fluxes (5.49) is conservative, well-balanced and entropy conservative. The general conservation property of the scheme follows immediately from the equivalent flux differencing formulation we will derive later. We start by showing that the scheme is entropy conservative in the following Lemma.

Lemma 13 (Entropy conservation of split form DG)**.** *The strong form discontinuous Galerkin approximation of the two dimensional split form shallow water equations presented in* (5.42)*,* (5.46) *and* (5.47) *with source term discretization* (5.41) *and numerical interface fluxes* (5.49) *is entropy conservative.*

Proof. To rigorously show that the scheme is entropy conservative we derive the discrete entropy equation by contracting the discrete system with the entropy variables $\vec{q} = \left(g(h+b) - \frac{1}{2}(u^2+v^2), u, v\right)$. Assuming continuity in time, this leads to a semi-discrete equation for the entropy $e = \frac{g}{2}h^2 + ghb + \frac{1}{2}h(u^2+v^2)$, as proven in Lemma 2. As the full discrete entropy equation is quite cumbersome to work with, we focus our analysis on one individual metric term. The steps below work analogously for each of the four metric terms and the complete proof can be found in [253]. The volume terms factoring the metric term y_η from the discrete equations (5.42), (5.46) and (5.47) contracted by the entropy variables are

$$
\begin{aligned}
&\left(g(h_{ij}+b_{ij}) - \frac{1}{2}u_{ij}^2 - \frac{1}{2}v_{ij}^2\right)(D \circ y_\eta \circ hu)_{ij}^{\xi} \\
&+ u_{ij}\,(D \circ y_\eta \circ hu \circ u)_{ij}^{\xi} + \frac{g}{2}u_{ij}h_{ij}\left((D \circ y_\eta \circ h)_{ij}^{\xi} + (D \circ y_\eta \circ b)_{ij}^{\xi}\right) \\
&+ v_{ij}\,(D \circ y_\eta \circ hu \circ v)_{ij}^{\xi}\,.
\end{aligned}
\tag{5.51}
$$

We expand the split form discretizations in (5.51) according to Definition 2 and rearrange terms such that we find the discrete entropy flux derivative $\mathcal{F}_\xi$ with

$$
\mathcal{F} = \frac{1}{2}h(u^3+uv^2) + ghu(h+b),
$$

as well as eight discrete product rule error terms and one term that includes the discrete derivative of the metric term y_η:

$$\begin{aligned}\sum_{m=0}^{N} D_{im}\left(\mathcal{F}_{mj}\left(y_\eta\right)_{mj}\right) + \mathcal{E}\left(u, hu^2 y_\eta\right) + \mathcal{E}\left(hu^2, u y_\eta\right) + \mathcal{E}\left(v, huv y_\eta\right)& \\ + \mathcal{E}\left(huv, v y_\eta\right) + \mathcal{E}\left(h, hu y_\eta\right) + \mathcal{E}\left(h y_\eta, hu\right) + \mathcal{E}\left(b, hu y_\eta\right) + \mathcal{E}\left(b y_\eta, hu\right)& \\ + \frac{g}{2} h_{ij} u_{ij}\left(b_{ij} + h_{ij}\right) \sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj}.&\end{aligned} \tag{5.52}$$

For completeness, we list the exact form of the abbreviated non-linear discrete product rule error terms from (5.52)

$$\begin{aligned}
\mathcal{E}\left(u, hu^2 y_\eta\right) &:= \frac{1}{4}\left(-\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj}^3 \left(y_\eta\right)_{mj}\right) + u_{ij}\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj}^2 \left(y_\eta\right)_{mj}\right) + h_{ij} u_{ij}^2 \left(y_\eta\right)_{ij} \sum_{m=0}^{N} D_{im}\left(u_{mj}\right)\right), \\
\mathcal{E}\left(hu^2, u y_\eta\right) &:= \frac{1}{4}\left(-\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj}^3 \left(y_\eta\right)_{mj}\right) + \left(y_\eta\right)_{ij} u_{ij}\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj}^2\right) + h_{ij} u_{ij}^2 \sum_{m=0}^{N} D_{im}\left(\left(y_\eta\right)_{mj} u_{mj}\right)\right), \\
\mathcal{E}\left(v, huv y_\eta\right) &:= \frac{1}{4}\left(-\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj} v_{mj}^2 \left(y_\eta\right)_{mj}\right) + v_{ij}\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj} v_{mj} \left(y_\eta\right)_{mj}\right) + h_{ij} u_{ij} v_{ij} \left(y_\eta\right)_{ij} \sum_{m=0}^{N} D_{im}\left(v_{mj}\right)\right), \\
\mathcal{E}\left(y_\eta v, huv\right) &:= \frac{1}{4}\left(-\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj} v_{mj}^2 \left(y_\eta\right)_{mj}\right) + \left(y_\eta\right)_{ij} v_{ij}\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj} v_{mj}\right) + h_{ij} u_{ij} v_{ij} \sum_{m=0}^{N} D_{im}\left(\left(y_\eta\right)_{mj} v_{mj}\right)\right), \\
\mathcal{E}\left(h, hu y_\eta\right) &:= \frac{g}{2}\left(-\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj} v_{mj}^2 \left(y_\eta\right)_{mj}\right) + h_{ij}\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj} \left(y_\eta\right)_{mj}\right) + h_{ij} u_{ij} \left(y_\eta\right)_{ij} \sum_{m=0}^{N} D_{im}\left(h_{mj}\right)\right), \\
\mathcal{E}\left(h y_\eta, hu\right) &:= \frac{g}{2}\left(-\sum_{m=0}^{N} D_{im}\left(h_{mj}^2 u_{mj} \left(y_\eta\right)_{mj}\right) + \left(y_\eta\right)_{ij} h_{ij}\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj}\right) + h_{ij} u_{ij} \sum_{m=0}^{N} D_{im}\left(\left(y_\eta\right)_{mj} h_{mj}\right)\right), \\
\mathcal{E}\left(b, hu y_\eta\right) &:= \frac{g}{2}\left(-\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj} v_{mj}^2 \left(y_\eta\right)_{mj}\right) + b_{ij}\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj} \left(y_\eta\right)_{mj}\right) + h_{ij} u_{ij} \left(y_\eta\right)_{ij} \sum_{m=0}^{N} D_{im}\left(b_{mj}\right)\right), \\
\mathcal{E}\left(b y_\eta, hu\right) &:= \frac{g}{2}\left(-\sum_{m=0}^{N} D_{im}\left(h_{mj}^2 u_{mj} \left(y_\eta\right)_{mj}\right) + \left(y_\eta\right)_{ij} b_{ij}\sum_{m=0}^{N} D_{im}\left(h_{mj} u_{mj}\right) + h_{ij} u_{ij} \sum_{m=0}^{N} D_{im}\left(\left(y_\eta\right)_{mj} b_{mj}\right)\right).
\end{aligned} \tag{5.53}$$

The discrete product rule error terms in (5.52) do not destroy conservation, as we know from Lemma 4. Thus, we only need to make an argument for the remaining discrete derivative of the metric term. If we follow analogous steps for the volume terms corresponding to the metric term y_ξ we find very similar expressions. Combining the terms with the discrete derivative of the metric terms we see that they cancel as long as the discrete metric identities (3.63) hold:

$$\frac{g}{2} h_{ij} u_{ij}\left(b_{ij} + h_{ij}\right)\left(\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj} - \sum_{m=0}^{N} D_{jm}\left(y_\xi\right)_{im}\right) = 0. \tag{5.54}$$

The same argument must be made for similar terms that cross-cancel between the x_η and x_ξ terms. We use the SBP property

$$D_{ij} = \frac{\delta_{iN}}{\omega_N} - \frac{\delta_{i0}}{\omega_0} - \hat{D}_{ij} \tag{5.55}$$

to move terms from the entropy flux derivative to the surface. The discrete integral of remaining flux derivative term is zero due to the consistency of the **D** Matrix,

$$\begin{aligned}
-\sum_{i,j=0}^{N}\omega_i\omega_j\sum_{m=0}^{N}\hat{D}_{im}\left(\mathcal{F}_{mj}\left(y_\eta\right)_{mj}\right) &= \sum_{i,j=0}^{N}\omega_j\sum_{m=0}^{N}D_{mi}\omega_m\left(\mathcal{F}_{mj}\left(y_\eta\right)_{mj}\right)\\
&= \sum_{m,j=0}^{N}\omega_j\omega_m\left(\mathcal{F}_{mj}\left(y_\eta\right)_{mj}\right)\sum_{i=0}^{N}D_{mi} = 0.
\end{aligned} \tag{5.56}$$

Thus, we conclude that the volume terms of the discrete entropy equation are in fact conservative. We collect the surface terms factoring y_η from (5.52) and the new terms due to the swap to conservative form to find

$$\begin{aligned}
&y_\eta\text{-interface terms for a single element}\\
&= -\frac{\delta_{iN}}{\omega_i}\Bigg[\left(g(h+b)-\frac{1}{2}u^2-\frac{1}{2}v^2\right)\left(y_\eta\left(\{\{h\}\}\{\{u\}\}-hu\right)\right)\\
&\quad + u\left(y_\eta\left(\{\{h\}\}\{\{u\}\}^2+\frac{g}{2}\left\{\left\{h^2\right\}\right\}-hu^2-\frac{g}{2}h^2-\frac{g}{2}\{\{h\}\}[\![b]\!]\right)\right)\\
&\quad + v\left(y_\eta\left(\{\{h\}\}\{\{u\}\}\{\{v\}\}-huv\right)\right)\\
&\quad +\frac{1}{2}h(u^3+uv^2)+ghu(h+b)\Bigg]_{Nj}\\
&\quad + \frac{\delta_{i0}}{\omega_i}\Bigg[\left(g(h+b)-\frac{1}{2}u^2-\frac{1}{2}v^2\right)\left(y_\eta\left(\{\{h\}\}\{\{u\}\}-hu\right)\right)\\
&\quad + u\left(y_\eta\left(\{\{h\}\}\{\{u\}\}^2+\frac{g}{2}\left\{\left\{h^2\right\}\right\}-hu^2-\frac{g}{2}h^2-\frac{g}{2}\{\{h\}\}[\![b]\!]\right)\right)\\
&\quad + v\left(y_\eta\left(\{\{h\}\}\{\{u\}\}\{\{v\}\}-huv\right)\right)\\
&\quad +\frac{1}{2}h(u^3+uv^2)+ghu(h+b)\Bigg]_{0j}.
\end{aligned} \tag{5.57}$$

We simplify the terms at an arbitrary one of the two interfaces in (5.57) and factor out the scaling by metric term and integration weight $\frac{y_\eta}{\omega_i}$. We also drop the outer index that indicates the interface node, to find

$$\begin{aligned}
&y_\eta\text{-terms at one interface node of a single element}\\
&=\left(g(h+b)-\frac{1}{2}u^2-\frac{1}{2}v^2\right)\{\{h\}\}\{\{u\}\}+u\{\{h\}\}\{\{u\}\}^2\\
&\quad+\frac{g}{2}u\left\{\left\{h^2\right\}\right\}+v\{\{h\}\}\{\{u\}\}\{\{v\}\}-\frac{g}{2}h^2u-\frac{g}{2}u\{\{h\}\}[\![b]\!].
\end{aligned} \tag{5.58}$$

The scheme is entropy conservative if the total interface contribution at any fixed interface node is zero. The metric terms on the interface are identical for two neighbouring elements but with opposite sign, since the normals are defined outward pointing. This reversed sign leads to jump terms, when both local contributions are added. As explained in the one dimensional proof, the jump in bottom topography $[\![b]\!]$ is defined as outward pointing for each element. Thus the sign of the jump is reversed for each element and we obtain an average of the velocity, $\{\{u\}\}$, when summing $\frac{g}{2}u\,\{\{h\}\}\,[\![b]\!]$ from both elements. The total contribution with metric term y_η at a fixed interface node is given by

$$\begin{aligned}
&\text{Total interface contribution with metric term } y_\eta \text{ at a single interface node}\\
=&g([\![h]\!]+[\![b]\!])\,\{\{h\}\}\,\{\{u\}\} - \frac{1}{2}[\![u^2]\!]\,\{\{h\}\}\,\{\{u\}\} - \frac{1}{2}[\![v^2]\!]\,\{\{h\}\}\,\{\{u\}\}\\
&+ [\![u]\!]\,\{\{h\}\}\,\{\{u\}\}^2 + \frac{g}{2}[\![u]\!]\left\{\left\{h^2\right\}\right\} + [\![v]\!]\,\{\{h\}\}\,\{\{u\}\}\,\{\{v\}\}\\
&- \frac{g}{2}[\![h^2u]\!] - g\,\{\{h\}\}\,\{\{u\}\}\,[\![b]\!].
\end{aligned} \tag{5.59}$$

From the rules for jumps and averages we know $\frac{1}{2}[\![a^2]\!] = \{\{a\}\}\,[\![a]\!]$ and can simplify (5.59) further to find

$$\begin{aligned}
&\text{Remaining terms with metric term } y_\eta \text{ at a single interface node}\\
=&g\,\{\{h\}\}\,\{\{u\}\}\,[\![h]\!] + \frac{g}{2}\left(\left\{\left\{h^2\right\}\right\}[\![u]\!] - [\![h^2u]\!]\right).
\end{aligned} \tag{5.60}$$

Furthermore, we can apply the rules to the more complex jump terms to see

$$\frac{g}{2}[\![h^2u]\!] = \frac{g}{2}\,\{\{u\}\}\,[\![h^2]\!] + \frac{g}{2}\left\{\left\{h^2\right\}\right\}[\![u]\!] = g\,\{\{h\}\}\,\{\{u\}\}\,[\![h]\!] + \frac{g}{2}\left\{\left\{h^2\right\}\right\}[\![u]\!]. \tag{5.61}$$

Thus, all the terms in (5.60) cancel. We have shown that all the terms associated with the metric term y_η cancel. Similar steps hold for the other metric terms as well and we conclude that the discrete entropy equation is fulfilled. ■

The next step is to prove that the scheme is well-balanced. We show this in the following Lemma.

Lemma 14 (Well-balanced property of two dimensional split form DG)**.** *The strong form discontinuous Galerkin approximation of the two dimensional split form shallow water equations presented in* (5.42)*,* (5.46) *and* (5.47) *with source term discretization* (5.41) *and numerical interface fluxes* (5.49) *is well-balanced.*

Proof. We need to verify that the lake at rest initial condition is preserved for all time. To do so, we examine the discretized equations in expanded split form, (5.42), (5.46) and (5.47). The specific source term discretization given in (5.41) is a key factor for this proof. Inserting the lake at rest initial condition (2.37) into the continuity equation (5.42), we see that all the terms vanish immediately as they all factor velocities which are zero. Without loss of generality, we examine the *hu* momentum equation. The *hv* equation follows analogously. We insert the lake at rest initial conditions into (5.46) to find

$$\begin{aligned}
J_{ij}\left((hu)_t\right)_{ij} &+ \frac{g}{2}h_{ij}\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj}\left(h_{mj}+b_{mj}\right) + \frac{g}{2}\left(y_\eta\right)_{ij}h_{ij}\sum_{m=0}^{N} D_{im}(h_{mj}+b_{mj})\\
&- \frac{g}{2}h_{ij}\sum_{m=0}^{N} D_{jm}\left(y_\xi\right)_{im}\left(h_{im}+b_{im}\right) - \frac{g}{2}\left(y_\xi\right)_{ij}h_{ij}\sum_{m=0}^{N} D_{mj}(h_{im}+b_{im})\\
&= -\frac{\delta_{iN}}{\omega_i}\frac{g}{2}\left(\left[y_\eta\left(\left\{\left\{h^2\right\}\right\} - h^2 + \{\{h\}\}\,[\![b]\!]\right)\right]_{Nj}\right)\\
&\quad + \frac{\delta_{i0}}{\omega_i}\frac{g}{2}\left(\left[y_\eta\left(\left\{\left\{h^2\right\}\right\} - h^2 + \{\{h\}\}\,[\![b]\!]\right)\right]_{0j}\right)\\
&\quad + \frac{\delta_{Nj}}{\omega_j}\frac{g}{2}\left(\left[y_\xi\left(\left\{\left\{h^2\right\}\right\} - h^2 + \{\{h\}\}\,[\![b]\!]\right)\right]_{iN}\right)\\
&\quad - \frac{\delta_{0j}}{\omega_j}\frac{g}{2}\left(\left[y_\xi\left(\left\{\left\{h^2\right\}\right\} - h^2 + \{\{h\}\}\,[\![b]\!]\right)\right]_{i0}\right).
\end{aligned} \tag{5.62}$$

The discrete derivatives of the constant total water height, $h+b = H_{\text{const}}$, are zero, since $\mathbf{D}$ is a consistent derivative operator. Also, we observe

$$\begin{aligned}
&\frac{g}{2}h_{ij}\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj}\left(h_{mj}+b_{mj}\right) - \frac{g}{2}h_{ij}\sum_{m=0}^{N} D_{jm}\left(y_\xi\right)_{im}\left(h_{im}+b_{im}\right)\\
=&\frac{g}{2}h_{ij}(H_{\text{const}})\left(\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj} - \sum_{m=0}^{N} D_{jm}\left(y_\xi\right)_{im}\right)\\
=&0,
\end{aligned} \tag{5.63}$$

due to the metric identities. Since we allow discontinuities in the bottom topography at element interfaces we must account for the jump in water height and we cannot guarantee that $\{\{h^2\}\} = h^2$. Instead, we have at each interface

$$\left\{\left\{h^2\right\}\right\} - h^2 = \frac{h_i^2 + h_o^2}{2} - h_i^2 = \frac{h_o^2 - h_i^2}{2} = \frac{1}{2}[\![h^2]\!] = \{\{h\}\}\,[\![h]\!], \tag{5.64}$$

where h_i denotes the inner and h_o the outer value. For each interface term in (5.62) we thus find

$$\left\{\left\{h^2\right\}\right\} - h^2 + \{\{h\}\}\,[\![b]\!] = \{\{h\}\}\,[\![h+b]\!] = 0. \tag{5.65}$$

Analogous steps are valid for the hv equation. Thus, the scheme preserves the lake at rest for all time and is well-balanced. ■

As we have established in the last chapter, split form discretizations can be written in the flux differencing form with appropriate numerical volume fluxes. Then, the flux differencing formulations given in (4.45) provide a compact notation of the strong form DG discretization of the split form equations on curvilinear meshes and previous results of Fisher and Carpenter for conservation and entropy conservation apply [65]. Also, we can avoid the calculation of the many individual volume integrals in the direct LGL-DGSEM discretization given by (5.42), (5.46) and (5.47). We present the equivalent flux differencing formulation in Lemma 15.

Lemma 15 (Equivalent Discrete Split Form)**.** *If the following curvilinear numerical volume fluxes are used in the DGSEM in flux differencing formulation from Theorem 1,*

$$
\begin{aligned}
\tilde{\vec{F}}_{(i,m),j} &:= \vec{F}^{\#}(\vec{W}_{i,j},\vec{W}_{m,j})\,\{\{y_\eta\}\}_{(i,m),j} - \vec{G}^{\#}(\vec{W}_{i,j},\vec{W}_{m,j})\,\{\{x_\eta\}\}_{(i,m),j}\,,\\
\tilde{\vec{G}}_{i,(m,j)} &:= -\vec{F}^{\#}(\vec{W}_{i,j},\vec{W}_{i,m})\,\{\{y_\xi\}\}_{i,(j,m)} + \vec{G}^{\#}(\vec{W}_{i,j},\vec{W}_{i,m})\,\{\{x_\xi\}\}_{i,(j,m)}\,,
\end{aligned}
\tag{5.66}
$$

with the numerical two-point fluxes defined as

$$
\begin{aligned}
\vec{F}^{\#}(\vec{W}_{i,j},\vec{W}_{m,j}) &:= \begin{pmatrix} \{\{hu\}\}_{(i,m),j} \\ \{\{hu\}\}_{(i,m),j}\,\{\{u\}\}_{(i,m),j} + g\,\{\{h\}\}^2_{(i,m),j} - \frac{1}{2}g\,\{\{h^2\}\}_{(i,m),j} \\ \{\{hu\}\}_{(i,m),j}\,\{\{v\}\}_{(i,m),j} \end{pmatrix},\\
\vec{G}^{\#}(\vec{W}_{i,j},\vec{W}_{i,m}) &:= \begin{pmatrix} \{\{hv\}\}_{i,(m,j)} \\ \{\{hv\}\}_{i,(m,j)}\,\{\{u\}\}_{i,(m,j)} \\ \{\{hv\}\}_{i,(m,j)}\,\{\{v\}\}_{i,(m,j)} + g\,\{\{h\}\}^2_{i,(m,j)} - \frac{1}{2}g\,\{\{h^2\}\}_{i,(m,j)} \end{pmatrix},
\end{aligned}
\tag{5.67}
$$

along with numerical surface fluxes (5.49) *and source term discretization* (5.41)*, then it is equivalent to the direct strong form DGSEM discretization of the split form shallow water equations* (5.34)*,* (5.37) *and* (5.38) *as given in* (5.42)*,* (5.46) *and* (5.47)*.*

Proof. We examine the spatial operators of the flux differencing formulation, e.g. the ξ operator

$$
\left(\vec{\mathcal{L}}_\xi\right)_{ij} = \frac{1}{\omega_i}\left(\delta_{iN}\left[\tilde{\vec{F}}^*\right]_{Nj} - \delta_{i0}\left[\tilde{\vec{F}}^*\right]_{0j}\right) + \sum_{m=0}^{N}\tilde{D}_{im}\tilde{\vec{F}}_{(i,m),j}. \tag{5.68}
$$

With the definition $\tilde{\mathbf{D}} = 2\mathbf{D} + \mathbf{S}$ we split off the strong form surface terms based on solely interior data. On the surface, i.e. for $i = 0$ or $i = N$, the numerical two-point fluxes are identical to the physical contravariant fluxes. Thus, we find

$$
\begin{aligned}
\sum_{m=0}^{N}\tilde{D}_{im}\tilde{\vec{F}}_{(i,m),j} &= \sum_{m=0}^{N}2D_{im}\tilde{\vec{F}}_{(i,m),j} - \frac{1}{\omega_i}\left(\delta_{iN}\left[\tilde{\vec{F}}_{(i,N),j}\right]_{Nj} + \delta_{i0}\left[\tilde{\vec{F}}_{(i,0),j}\right]_{0j}\right)\\
&= \sum_{m=0}^{N}2D_{im}\tilde{\vec{F}}_{(i,m),j} - \frac{1}{\omega_i}\left(\delta_{iN}\tilde{\vec{F}}_{Nj} + \delta_{i0}\tilde{\vec{F}}_{0j}\right).
\end{aligned}
\tag{5.69}
$$

Thus, we can recover the strong form DGSEM surface terms

$$
\left(\vec{\mathcal{L}}_\xi\right)_{ij} = \frac{1}{\omega_i}\left(\delta_{iN}\left[\tilde{\vec{F}}^* - \tilde{\vec{F}}\right]_{Nj} - \delta_{i0}\left[\tilde{\vec{F}}^* - \tilde{\vec{F}}\right]_{0j}\right) + \sum_{m=0}^{N}2D_{im}\tilde{\vec{F}}_{(i,m),j}. \tag{5.70}
$$

The same argument holds for the $j = 0$ and $j = N$ interfaces. It is is left to show that the remaining volume terms are equivalent to the split form DG volume terms. We extend the flux differencing volume integral into full split forms for all three equations by using the rules from Lemma 9. Starting with the continuity equation we see

$$\begin{aligned}
&\sum_{m=0}^{N} 2D_{im}\left(\tilde{F}_1\right)_{(i,m),j} + \sum_{m=0}^{N} D_{jm}\left(\tilde{G}_1\right)_{i,(m,j)} \\
&\quad= \sum_{m=0}^{N} 2D_{im}\left(\{\{y_\eta\}\}_{(i,m),j}\{\{hu\}\}_{(i,m),j} - \{\{x_\eta\}\}_{(i,m),j}\{\{hv\}\}_{(i,m),j}\right) \\
&\qquad+ \sum_{m=0}^{N} 2D_{jm}\left(-\{\{y_\xi\}\}_{i,(j,m)}\{\{hu\}\}_{i,(j,m)} + \{\{x_\xi\}\}_{i,(j,m)}\{\{hv\}\}_{i,(j,m)}\right) \\
&\quad= (D \circ y_\eta \circ hu)^{\xi}_{ij} - (D \circ x_\eta \circ hv)^{\xi}_{ij} - (D \circ y_\xi \circ hu)^{\eta}_{ij} + (D \circ x_\xi \circ hv)^{\eta}_{ij}\,.
\end{aligned} \tag{5.71}$$

These are exactly the volume terms from the DG discretization of the split form continuity equation (5.42). We continue with the straightforward expansion of terms for the momentum equations. In case of the the hu equation, we find

$$\begin{aligned}
&\sum_{m=0}^{N} 2D_{im}\left(\tilde{F}_2\right)_{(i,m),j} + \sum_{m=0}^{N} 2D_{jm}\left(\tilde{G}_2\right)_{i,(m,j)} \\
&\quad= (D \circ y_\eta \circ hu \circ u)^{\xi}_{ij} - (D \circ x_\eta \circ hv \circ u)^{\xi}_{ij} + g\,(D \circ y_\eta \circ h \circ h)^{\xi}_{ij} - \frac{g}{2}\left(D \circ y_\eta \circ h^2\right)^{\xi}_{ij} \\
&\qquad- (D \circ y_\xi \circ hu \circ u)^{\eta}_{ij} + (D \circ x_\xi \circ hv \circ u)^{\eta}_{ij} - g\,(D \circ y_\xi \circ h \circ h)^{\eta}_{ij} + \frac{g}{2}\left(D \circ y_\xi \circ h^2\right)^{\eta}_{ij}\,,
\end{aligned} \tag{5.72}$$

whereas the extension of the hv equation is given by

$$\begin{aligned}
&\sum_{m=0}^{N} 2D_{im}\left(\tilde{F}_3\right)_{(i,m),j} + \sum_{m=0}^{N} 2D_{jm}\left(\tilde{G}_3\right)_{i,(m,j)} \\
&\quad= (D \circ y_\eta \circ hu \circ v)^{\xi}_{ij} - (D \circ x_\eta \circ hv \circ v)^{\xi}_{ij} - g\,(D \circ x_\eta \circ h \circ h)^{\xi}_{ij} + \frac{g}{2}\left(D \circ x_\eta \circ h^2\right)^{\xi}_{ij} \\
&\qquad- (D \circ y_\xi \circ hu \circ v)^{\eta}_{ij} + (D \circ x_\xi \circ hv \circ v)^{\eta}_{ij} + g\,(D \circ x_\xi \circ h \circ h)^{\eta}_{ij} - \frac{g}{2}\left(D \circ x_\xi \circ h^2\right)^{\eta}_{ij}\,.
\end{aligned} \tag{5.73}$$

These expansions are not immediately identical to direct DG split form discretizations (5.46) and (5.47). Similar to the proof in one dimension in Lemma 12, specifically in (5.25), it is still left to show that the pressure terms are identical.

$$\begin{aligned}
&g\,(D \circ y_\eta \circ h \circ h)^{\xi}_{ij} - \frac{g}{2}\left(D \circ y_\eta \circ h^2\right)^{\xi}_{ij} - g\,(D \circ y_\xi \circ h \circ h)^{\eta}_{ij} + \frac{g}{2}\left(D \circ \xi \circ h^2\right)^{\eta}_{ij} \\
&\overset{!}{=} \frac{g}{2}h_{ij}\,(D \circ y_\eta \circ h)^{\xi}_{ij} - \frac{g}{2}h_{ij}\,(D \circ y_\xi \circ h)^{\eta}_{ij}
\end{aligned} \tag{5.74}$$

We start with the y_η terms and extend the notations according to (4.62) and (4.63) to find

$$
\begin{aligned}
& g\left(D \circ y_\eta \circ h \circ h\right)_{ij}^{\xi} - \frac{g}{2}\left(D \circ y_\eta \circ h^2\right)_{ij}^{\xi} \\
=& \frac{g}{4}\left(\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj} h_{mj}^2 + \left(y_\eta\right)_{ij}\sum_{m=0}^{N} D_{im} h_{mj}^2 + 2h_{ij}\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{im} h_{mj}\right. \\
& \left. + h_{ij}^2 \sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj} + 2\left(y_\eta\right)_{ij} h_{ij}\sum_{m=0}^{N} D_{im} h_{mj}\right) \\
& - \frac{g}{4}\left(\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{im} h_{mj}^2 + h_{ij}^2\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj} + \left(y_\eta\right)_{ij}\sum_{m=0}^{N} D_{im} h_{mj}^2\right) \\
=& \frac{g}{2} h_{ij}\left(\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj} h_{mj} + \left(y_\eta\right)_{ij}\sum_{m=0}^{N} D_{im} h_{mj}\right) \\
=& \frac{g}{2} h_{ij}\left(D \circ y_\eta \circ h\right)_{ij}^{\xi} - \frac{g}{2} h_{ij}^2\left(\sum_{m=0}^{N} D_{im}\left(y_\eta\right)_{mj}\right).
\end{aligned}
\tag{5.75}
$$

Analyzing the pressure terms for y_ξ we find a similar extra term $\frac{g}{2}h_{ij}^2\left(\sum_{m=0}^{N} D_{jm}\left(y_\xi\right)_{im}\right)$. Together, these terms cancel due to the metric identities, as we have seen before in (5.45). The same argument can be made for the x_ξ and x_η pressure terms. We conclude that the equations in flux differencing formulation are equivalent to the straightforward strong form DGSEM discretization of the split form equations. ■

We have seen in Lemma 15 that the specific choice of two-point fluxes given by (5.67) in the flux differencing scheme leads to an equivalency between the scheme in flux differencing form and a direct DG discretization of the split form equations. We summarize the full ECDGSEM and its properties in Theorem 3.

Theorem 3 (Curvilinear ECDGSEM). *The semi-discrete DG approximation to the two dimensional split form shallow water equations on curvilinear grids is given by*

$$
J\vec{W}_t + \vec{\mathcal{L}}_\xi + \vec{\mathcal{L}}_\eta = \vec{\tilde{S}} \tag{5.76}
$$

with spatial operators

$$
\begin{aligned}
\left(\vec{\mathcal{L}}_\xi\right)_{ij} &= \frac{1}{\omega_i}\left(\delta_{iN}\left[\vec{\tilde{F}}^{*,ec}\right]_{Nj} - \delta_{i0}\left[\vec{\tilde{F}}^{*,ec}\right]_{0j}\right) + \sum_{m=0}^{N} \tilde{D}_{im}\vec{\tilde{F}}_{(i,m),j}, \\
\left(\vec{\mathcal{L}}_\eta\right)_{ij} &= \frac{1}{\omega_j}\left(\delta_{Nj}\left[\vec{\tilde{G}}^{*,ec}\right]_{iN} - \delta_{0j}\left[\vec{\tilde{G}}^{*,ec}\right]_{i0}\right) + \sum_{m=0}^{N} \tilde{D}_{jm}\vec{\tilde{G}}_{i,(m,j)},
\end{aligned}
\tag{5.77}
$$

and curvilinear numerical volume fluxes

$$
\begin{aligned}
\vec{\tilde{F}}_{(i,m),j} &:= \vec{F}^{\#}(\vec{W}_{i,j}, \vec{W}_{m,j})\left\{\!\left\{y_\eta\right\}\!\right\}_{(i,m),j} - \vec{G}^{\#}(\vec{W}_{i,j}, \vec{W}_{m,j})\left\{\!\left\{x_\eta\right\}\!\right\}_{(i,m),j}, \\
\vec{\tilde{G}}_{i,(m,j)} &:= -\vec{F}^{\#}(\vec{W}_{i,j}, \vec{W}_{i,m})\left\{\!\left\{y_\xi\right\}\!\right\}_{i,(j,m)} + \vec{G}^{\#}(\vec{W}_{i,j}, \vec{W}_{i,m})\left\{\!\left\{x_\xi\right\}\!\right\}_{i,(j,m)},
\end{aligned}
\tag{5.78}
$$

where the entropy conserving numerical two-point fluxes are defined by

$$\begin{aligned}\vec{F}^{\#}(\vec{W}_{i,j},\vec{W}_{m,j}) &:= \begin{pmatrix} \{\{hu\}\}_{(i,m),j} \\ \{\{hu\}\}_{(i,m),j}\,\{\{u\}\}_{(i,m),j} + g\,\{\{h\}\}^2_{(i,m),j} - \frac{1}{2}g\,\{\{h^2\}\}_{(i,m),j} \\ \{\{hu\}\}_{(i,m),j}\,\{\{v\}\}_{(i,m),j} \end{pmatrix}, \\ \vec{G}^{\#}(\vec{W}_{i,j},\vec{W}_{i,m}) &:= \begin{pmatrix} \{\{hv\}\}_{i,(m,j)} \\ \{\{hv\}\}_{i,(m,j)}\,\{\{u\}\}_{i,(m,j)} \\ \{\{hv\}\}_{i,(m,j)}\,\{\{v\}\}_{i,(m,j)} + g\,\{\{h\}\}^2_{i,(m,j)} - \frac{1}{2}g\,\{\{h^2\}\}_{i,(m,j)} \end{pmatrix}.\end{aligned} \tag{5.79}$$

The numerical interface fluxes are given by

$$\begin{aligned}\vec{\tilde{F}}^{*,ec} &= y_\eta \vec{F}^{*,ec} - x_\eta \vec{G}^{*,ec}. \\ \vec{\tilde{G}}^{*,ec} &= -y_\xi \vec{F}^{*,ec} + x_\xi \vec{G}^{*,ec},\end{aligned} \tag{5.80}$$

and depend on the the entropy conserving fluxes in x and y direction given as

$$\begin{aligned}\vec{F}^{*,ec}(\vec{W}^+,\vec{W}^-) &:= \begin{pmatrix} \{\{h\}\}\,\{\{u\}\} \\ \{\{h\}\}\,\{\{u\}\}^2 + \frac{g}{2}\,\{\{h^2\}\} \\ \{\{h\}\}\,\{\{u\}\}\,\{\{v\}\} \end{pmatrix}, \\ \vec{G}^{*,ec}(\vec{W}^+,\vec{W}^-) &:= \begin{pmatrix} \{\{h\}\}\,\{\{v\}\} \\ \{\{h\}\}\,\{\{u\}\}\,\{\{v\}\} \\ \{\{h\}\}\,\{\{v\}\}^2 + \frac{g}{2}\,\{\{h^2\}\} \end{pmatrix}.\end{aligned} \tag{5.81}$$

The source term is discretized by

$$\begin{aligned}(\tilde{s}_2)_{ij} :=& -g h_{ij}\left((D\circ y_\eta \circ b)^\xi_{ij} - (D\circ y_\xi\circ b)^\eta_{ij}\right) \\ &- \frac{\delta_{iN}}{\omega_i}\left[\frac{g}{2}y_\eta\,\{\{h\}\}\,[\![b]\!]\right]_{Nj} - \frac{\delta_{i0}}{\omega_i}\left[\frac{g}{2}y_\eta\,\{\{h\}\}\,[\![b]\!]\right]_{0j} \\ &+ \frac{\delta_{Nj}}{\omega_j}\left[\frac{g}{2}y_\xi\,\{\{h\}\}\,[\![b]\!]\right]_{iN} + \frac{\delta_{0j}}{\omega_j}\left[\frac{g}{2}y_\xi\,\{\{h\}\}\,[\![b]\!]\right]_{i0}, \\ (\tilde{s}_3)_{ij} :=& -g h_{ij}\left(-(D\circ x_\eta\circ b)^\xi_{ij} + (D\circ x_\xi\circ b)^\eta_{ij}\right) \\ &+ \frac{\delta_{iN}}{\omega_i}\left[\frac{g}{2}x_\eta\,\{\{h\}\}\,[\![b]\!]\right]_{Nj} + \frac{\delta_{i0}}{\omega_i}\left[\frac{g}{2}x_\eta\,\{\{h\}\}\,[\![b]\!]\right]_{0j} \\ &- \frac{\delta_{Nj}}{\omega_j}\left[\frac{g}{2}x_\xi\,\{\{h\}\}\,[\![b]\!]\right]_{iN} - \frac{\delta_{0j}}{\omega_j}\left[\frac{g}{2}x_\xi\,\{\{h\}\}\,[\![b]\!]\right]_{i0}.\end{aligned} \tag{5.82}$$

The scheme described above is called the ECDGSEM and has the following properties:

15.1 *Discrete conservation of the mass and discrete conservation of the momentum if the bottom topography is constant.*

15.2 Guaranteed conservation of the total discrete energy, which is an entropy function for the shallow water equations. Hence it fulfills the discrete entropy equality (2.52)*.*

15.3 Discrete well-balanced property for arbitrary bottom topographies.

Proof. We have shown in Lemma 15 that the presented flux differencing scheme is equivalent to a discontinuous Galerkin discretization of the split form as given in (5.42), (5.46) and (5.47). Thus, results from Lemma 13 for the entropy conservation and Lemma 14 for the well-balanced property apply. The discrete conservation of the numerical scheme follows directly from the telescoping flux differencing form of the approximation [65]. ■

Remark 3 (Metric identities)*.* The proof of the entropy conservation property 3.2 requires that the metric identities (3.61) hold discretely. Specifically, this is used in (5.52) to cancel terms.

Remark 4 (Numerical Flux)*.* The entropy conserving numerical flux on the interfaces is not unique. If we used the entropy conserving two-point flux on the interfaces as well and investigate the well-balanced property we find surface terms of the form

$$\{\{h\}\}^2 - h^2 + \{\{h\}\}\,[\![b]\!]. \tag{5.83}$$

In contrast to (5.64) we then observe

$$\begin{aligned}\{\{h\}\}^2 - h^2 &= \frac{(h_i+h_o)^2}{4} - h_i^2 = \frac{h_o^2 + 2h_ih_o - h_i^2}{4} - \frac{1}{2}h_i^2\\ &= \frac{1}{4}[\![h^2]\!] + \frac{1}{2}h_i[\![h]\!] = \frac{1}{2}\left(\{\{h\}\} + h_i\right)[\![h]\!].\end{aligned} \tag{5.84}$$

To cancel all the interface terms the source term discretization must then be chosen accordingly with terms of the kind $\frac{1}{2}\left(\{\{h\}\} + h_i\right)[\![b]\!]$.

5.3. Entropy stability

In the previous sections we have developed entropy conservative schemes for the one dimensional (Theorem 2) and two dimensional (Theorem 3) shallow water equations. These methods conserve the entropy in the semi-discrete system up to machine precision, apart from changes due to any boundary conditions. However, in the presence of discontinuities (shocks), this is not a desirable behaviour and solutions of non-linear PDEs may develop such shocks in finite time even for smooth initial data. It is necessary to replace the entropy conservation law (2.52) by an entropy inequality (2.53) in the presence of shocks [233]. We accomplish this by adding numerical dissipation to the entropy conservative numerical fluxes (5.80). We do so in a controlled way, proportional to the jump in entropy variables, such that the entropy is guaranteed to be dissipated. Then it is guaranteed that the discrete entropy inequality holds.

We first note, that the physical fluxes in two dimensions

$$\vec{f} = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}, \quad \vec{g} = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{pmatrix}, \tag{5.85}$$

have the associated flux Jacobians

$$\mathbf{A}_f = \vec{f}_{\vec{w}} = \begin{pmatrix} 0 & 1 & 0 \\ gh - u^2 & 2u & 0 \\ -uv & v & u \end{pmatrix}, \tag{5.86}$$

and

$$\mathbf{A}_g = \vec{g}_{\vec{w}} = \begin{pmatrix} 0 & 0 & 1 \\ -uv & v & u \\ gh - v^2 & 0 & 2v \end{pmatrix}. \tag{5.87}$$

The eigenvalues of $\mathbf{A}_f$ are $\lambda_f^1 = u + c$, $\lambda_f^2 = u$, $\lambda_f^3 = u - c$ and the eigenvalues of $\mathbf{A}_g$ similarly are $\lambda_g^1 = v + c$, $\lambda_g^2 = v$, $\lambda_g^3 = v - c$. In these expressions, the wave celerity is denoted by $c = \sqrt{gh}$. The matrices of right eigenvectors of (5.86) and (5.87) are

$$\mathbf{R}_f = \begin{pmatrix} 1 & 0 & 1 \\ u + c & 0 & u - c \\ v & 1 & v \end{pmatrix}, \tag{5.88}$$

and

$$\mathbf{R}_g = \begin{pmatrix} 1 & 0 & 1 \\ u & 1 & u \\ v + c & 0 & v - c \end{pmatrix}, \tag{5.89}$$

respectively.

We also need the entropy Jacobian matrix $\mathbf{H} = \vec{w}_{\vec{q}}$, which is found to be

$$\mathbf{H} = \frac{1}{g} \begin{pmatrix} 1 & u & v \\ u & gh + u^2 & uv \\ v & uv & gh + v^2 \end{pmatrix}. \tag{5.90}$$

We can relate the right eigenvectors to the entropy Jacobian with an appropriate scaling [173]

$$\mathbf{H} = (\mathbf{RT})(\mathbf{RT})^T, \tag{5.91}$$

with diagonal scaling matrix

$$\mathbf{T} = \operatorname{diag}(\sqrt{s_1}, \sqrt{s_2}, \sqrt{s_3}). \tag{5.92}$$

We define $\mathbf{Z} = \mathbf{T}^2$ and have the identity in a new form

$$\mathbf{H} = \mathbf{RZR}^T. \tag{5.93}$$

For the eigenvectors of $\mathbf{A}_f$, denoted $\mathbf{R}_f$, we find

$$s_1 = \frac{1}{2g}, \qquad s_2 = h, \quad s_3 = \frac{1}{2g}. \tag{5.94}$$

A straightforward calculation shows that the same scaling applies to the eigenvectors of the flux Jacobian $\mathbf{A}_g$ as well.

We can now define the entropy stable numerical surface flux functions. We subtract the dissipation terms required for dissipation in the $x-$direction

$$\vec{F}^{*,es} = \vec{F}^{*,ec} - \frac{1}{2}\mathbf{R}_f\,|\mathbf{\Lambda}_f|\,\mathbf{Z}\,\mathbf{R}_f^T[\![\vec{q}\,]\!], \tag{5.95}$$

and the $y-$direction

$$\vec{G}^{*,es} = \vec{G}^{*,ec} - \frac{1}{2}\mathbf{R}_g\,|\mathbf{\Lambda}_g|\,\mathbf{Z}\,\mathbf{R}_g^T[\![\vec{q}\,]\!], \tag{5.96}$$

where $\mathbf{\Lambda}_f$ and $\mathbf{\Lambda}_g$ are the diagonal matrices containing the eigenvalues computed above. All these terms in (5.95) and (5.96) are evaluated as arithmetic averages at an element interface. It is important that the dissipation terms are proportional to the jumps of the entropy variables and not to the jump of the conserved quantities as, for example, in the numerical Lax-Friedrichs flux (3.80). If we compute the discrete entropy equation by contracting the scheme with the entropy variables, we obtain contributions of the form

$$-\frac{1}{2}[\![\vec{q}\,]\!]^T\mathbf{R}_f|\mathbf{\Lambda}_f|\,\mathbf{Z}\,\mathbf{R}_f^T[\![\vec{q}\,]\!] \tag{5.97}$$

at each interface. These are guaranteed to be negative due to the positivity of the matrix $\mathbf{R}_f\,|\mathbf{\Lambda}_f|\,\mathbf{Z}\,\mathbf{R}_f^T$. Thus, this choice of dissipation ensures that entropy is decreased when the jump in entropy variables across interfaces is large (e.g. shocks) and is nearly preserved when the jumps are small for well resolved smooth solutions.

We present the entropy stable DGSEM and its properties in the following Theorem.

Theorem 4 (Curvilinear ESDGSEM)**.** *The semi-discrete split DG approximation to the two dimensional shallow water equations on curvilinear grids*

$$J\vec{W}_t + \vec{\mathcal{L}}_\xi + \vec{\mathcal{L}}_\eta = \vec{\tilde{S}}, \tag{5.98}$$

with

$$\begin{aligned}\left(\vec{\mathcal{L}}_\xi\right)_{ij} &= \frac{1}{\omega_i}\left(\delta_{iN}\left[\vec{\tilde{F}}^{*,es}\right]_{Nj} - \delta_{i0}\left[\vec{\tilde{F}}^{*,es}\right]_{0j}\right) + \sum_{m=0}^{N}\tilde{D}_{im}\vec{\tilde{F}}_{(i,m),j},\\ \left(\vec{\mathcal{L}}_\eta\right)_{ij} &= \frac{1}{\omega_j}\left(\delta_{Nj}\left[\vec{\tilde{G}}^{*,es}\right]_{iN} - \delta_{0j}\left[\vec{\tilde{G}}^{*,es}\right]_{i0}\right) + \sum_{m=0}^{N}\tilde{D}_{mj}\vec{\tilde{G}}_{i,(m,j)},\end{aligned} \tag{5.99}$$

with curvilinear two-point fluxes defined as in the entropy conservative case (5.78)*. The numerical interface fluxes are given by*

$$\begin{aligned}\vec{\tilde{F}}^{*,es} &= y_\eta\vec{F}^{*,es} - x_\eta\vec{G}^{*,es},\\ \vec{\tilde{G}}^{*,es} &= -y_\xi\vec{F}^{*,es} + x_\xi\vec{G}^{*,es},\end{aligned} \tag{5.100}$$

and depend on the fluxes in x and y direction

$$\begin{aligned}\vec{F}^{*,es} &= \vec{F}^{*,ec} - \frac{1}{2}\mathbf{R}_f\,|\mathbf{\Lambda}_f|\,\mathbf{R}_f^T[\![\,\vec{q}\,]\!],\\ \vec{G}^{*,es} &= \vec{G}^{*,ec} - \frac{1}{2}\mathbf{R}_g\,|\mathbf{\Lambda}_g|\,\mathbf{R}_g^T[\![\,\vec{q}\,]\!],\end{aligned} \tag{5.101}$$

which include the entropy conserving fluxes $\vec{F}^{*,ec}$ and $\vec{G}^{*,ec}$ given in (5.81)*, as well as an entropy stable dissipation term which depends on the scaled flux eigenvalues*

$$\begin{aligned}|\mathbf{\Lambda}_f| &= \frac{1}{2g}\begin{pmatrix}|\,\{\{u\}\}+\{\{c\}\}\,| & 0 & 0\\ 0 & 2g|\,\{\{h\}\}\,\{\{u\}\}\,| & 0\\ 0 & 0 & |\,\{\{u\}\}-\{\{c\}\}\,|\end{pmatrix},\\ |\mathbf{\Lambda}_g| &= \frac{1}{2g}\begin{pmatrix}|\,\{\{v\}\}+\{\{c\}\}\,| & 0 & 0\\ 0 & 2g|\,\{\{h\}\}\,\{\{v\}\}\,| & 0\\ 0 & 0 & |\,\{\{v\}\}-\{\{c\}\}\,|\end{pmatrix},\end{aligned} \tag{5.102}$$

and eigenvectors

$$\begin{aligned}\mathbf{R}_f &= \begin{pmatrix}1 & 0 & 1\\ \{\{u\}\}+\{\{c\}\} & 0 & \{\{u\}\}-\{\{c\}\}\\ \{\{v\}\} & 1 & \{\{v\}\}\end{pmatrix},\\ \mathbf{R}_g &= \begin{pmatrix}1 & 0 & 1\\ \{\{u\}\} & 1 & \{\{u\}\}\\ \{\{v\}\}+\{\{c\}\} & 0 & \{\{v\}\}-\{\{c\}\}\end{pmatrix},\end{aligned} \tag{5.103}$$

and the jump in entropy variables $[\![\vec{q}\,]\!]$. The source term discretization $\vec{\vec{S}}$ is the same as in the entropy conserving case (5.41)*. The scheme described above is called the ESDGSEM and has the following properties:*

4.1 Discrete conservation of the mass and discrete conservation of the momentum if the bottom topography is constant.

4.2 Guaranteed dissipation of the total discrete energy, which is an entropy function for the shallow water equations. Hence it fulfills the discrete entropy inequality (2.53)

4.3 The well-balanced property for arbitrary bottom topographies, provided any discontinuities coincide with element interfaces.

Proof. The ESDGSEM follows directly from the curvilinear ECDGSEM presented in Thm. 3. To guarantee entropy stability we replace the entropy conserving numerical fluxes (5.81) at element interfaces with the entropy stable numerical fluxes (5.95), (5.96). For the “lake at rest” initial conditions the jump in entropy variables is zero, $[\![\vec{q}\,]\!] = 0$, so the additional dissipation term vanishes and does not affect the well-balanced property of the scheme. ■

The semi-discrete ESDGSEM is only complete when equipped with a suitable time integrator. We previously used a low storage Runge-Kutta scheme in [253] but have now switched to a strong stability preserving Runge-Kutta scheme (SSPRK). We will see that this choice is necessary for the positivity preserving limiter in Section 5.5. We use a SSPRK method frequently used in wet/dry shallow water schemes such as by Xing [265] and shock capturing schemes as by Shu [220]. The three stage scheme is given by

$$\begin{aligned} W^{(1)} &= W^n + \Delta t \mathcal{R}\left(W^n\right), \\ W^{(2)} &= \frac{3}{4} W^n + \frac{1}{4}\left(W^{(1)} + \Delta t \mathcal{R}\left(W^1\right)\right), \\ W^{n+1} &= \frac{1}{3} W^n + \frac{2}{3}\left(W^{(2)} + \Delta t \mathcal{R}\left(W^2\right)\right), \end{aligned} \tag{5.104}$$

where $\mathcal{R}$ denotes the spatial ESDGSEM operator

$$\mathcal{R} = -\frac{1}{J}\left(\vec{\mathcal{L}}_\xi + \vec{\mathcal{L}}_\eta - \vec{\tilde{S}}\right). \tag{5.105}$$

The time step is chosen according to the CFL condition

$$\Delta t = \frac{\min \Delta x}{\lambda_{\max}} \mathrm{CFL}, \tag{5.106}$$

where $\lambda_{\max}$ denotes the maximum eigenvalue of the system and the constant CFL < 0.5 is chosen before the start of the computation. The minimal element size $\min \Delta x$ is approximated a priori via numerical quadrature over each element.

5.4. Artificial viscosity shock capturing

In this chapter we describe how to add artificial viscosity to the ESDGSEM to reduce the amount of oscillations in the solution. Even without any limiter or artificial viscosity, the ESDGSEM demonstrates more robustness than a standard DGSEM approximation [83, 253]. However, it is not oscillation free, especially for (very) high-order simulations and in the presence of shocks. If the oscillations are too sever, they might lead to unphysical solutions and cause a simulation to break down. Since we do not want to abandon the entropy stable approach, we only consider additional smoothing or limiting, if it preserves the entropy stability of the scheme. In [81] the authors prove that artificial viscosity can be added in an entropy stable way. The key requirements are a certain connection between gradient variables and entropy variables and a suitable discretization according to Bassi and Rebay [8].

We choose to only add artificial viscosity to the momentum equations. The smoothing is weaker without direct artificial viscosity in the continuity equation. If we choose remaining gradient variables as the entropy variables, we obtain a straightforward one-to-one mapping to the entropy variables. Thus, we easily fulfill one of the requirements for entropy stability through a theorem from [81]. To be flexible with the amount of viscosity added to the scheme, we scale the gradient by the water height h and a viscosity parameter ϵ in the viscous fluxes. The viscosity parameter is chosen based on an estimation of the local smoothness of the solution and we will provide details on its computation later.

By introducing artificial viscosity to the shallow water system (2.33), we obtain a modified set of equations, which can be written out as

$$\begin{aligned} h_t + (hu)_x + (hv)_y &= 0, \\ (hu)_t + (h\,u^2 + g\,h^2/2)_x + (huv)_y &= -g\,h\,b_x + \nabla\cdot\left(h\,\epsilon\vec{\mathcal{U}}\right), \\ (hv)_t + (huv)_x + (h\,v^2 + g\,h^2/2)_y &= -g\,h\,b_y + \nabla\cdot\left(h\,\epsilon\,\vec{\mathcal{V}}\right), \\ \vec{\mathcal{U}} &= \nabla u, \\ \vec{\mathcal{V}} &= \nabla v, \end{aligned} \tag{5.107}$$

or expressed in compact flux form by

$$\begin{aligned} \vec{w}_t + \nabla\cdot(\vec{f},\vec{g})^T &= \vec{s} + \nabla\cdot(\vec{f}^v,\vec{g}^v)^T, \\ \vec{\mathcal{U}} &= \nabla u, \\ \vec{\mathcal{V}} &= \nabla v, \end{aligned} \tag{5.108}$$

with viscous fluxes $\vec{f}^v(\vec{w},\vec{\mathcal{U}},\vec{\mathcal{V}}) = h\,\epsilon\,(0,\mathcal{U}_1,\mathcal{V}_1)^T$ and $\vec{g}^v(\vec{w},\vec{\mathcal{U}},\vec{\mathcal{V}}) = h\,\epsilon\,(0,\mathcal{U}_2,\mathcal{V}_2)^T$. We include the new viscous fluxes into the flux divergence to find the system of equations in reference space as

$$\begin{aligned} \mathcal{J}\vec{w}_t &= -\hat{\nabla}\cdot(\vec{\tilde{f}} + \vec{\tilde{f}}^v, \vec{\tilde{g}} + \vec{\tilde{g}}^v)^T + \vec{\tilde{s}}, \\ \mathcal{J}\vec{\mathcal{U}} &= \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix}\hat{\nabla}u, \\ \mathcal{J}\vec{\mathcal{V}} &= \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix}\hat{\nabla}v, \end{aligned} \tag{5.109}$$

where we transform the physical gradient operator with the metrics of the element mappings.

We proceed to derive weak and strong formulations and their discretizations for the continuity and momentum equations, analogous to the steps of the standard DGSEM derivation in Sec. 3.2.2. The transformed gradient equations are multiplied by the same test function ϕ and integrated over the domain to find the weak form

$$\begin{aligned} &\int\limits_E \mathcal{J}\,\vec{\mathcal{U}}\,\phi\,\mathrm{dE} - \oint\limits_{\partial E} \phi\,U^*\,\vec{n}\,\mathrm{dS} + \int\limits_E u \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix} \hat{\nabla}\phi\,\mathrm{dE} = 0, \\ &\int\limits_E \mathcal{J}\,\vec{\mathcal{V}}\,\phi\,\mathrm{dE} - \oint\limits_{\partial E} \phi\,V^*\,\vec{n}\,\mathrm{dS} + \int\limits_E v \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix} \hat{\nabla}\phi\,\mathrm{dE} = 0, \end{aligned} \tag{5.110}$$

where U^* and V^* are the numerical interface states for the gradient equations. We discretize the weak form gradient equations (5.110) and simplify, using the Lagrange property. For example, the volume integral approximations for the $\vec{\mathcal{U}}$ equations reduce to

$$\begin{aligned}
&\int_{-1}^{1}\int_{-1}^{1} u\left(y_\eta \frac{\partial}{\partial\xi}(\ell_i(\xi)\ell_j(\eta)) - y_\xi \frac{\partial}{\partial\eta}(\ell_i(\xi)\ell_j(\eta))\right)\mathrm{d}\xi\mathrm{d}\eta \\
&\qquad\approx -\sum_{m=0}^{N}\hat{D}_{im}u_{mj}y_{\eta_{mj}}\,\omega_i\omega_j + \sum_{n=0}^{N}\hat{D}_{jn}u_{in}y_{\xi_{in}}\,\omega_i\omega_j, \\
&\int_{-1}^{1}\int_{-1}^{1} u\left(-x_\eta \frac{\partial}{\partial\xi}(\ell_i(\xi)\ell_j(\eta)) + x_\xi \frac{\partial}{\partial\eta}(\ell_i(\xi)\ell_j(\eta))\right)\mathrm{d}\xi\mathrm{d}\eta \\
&\qquad\approx \sum_{m=0}^{N}\hat{D}_{im}u_{mj}x_{\eta_{mj}}\,\omega_i\omega_j - \sum_{n=0}^{N}\hat{D}_{jn}u_{in}x_{\xi_{in}}\,\omega_i\omega_j.
\end{aligned} \tag{5.111}$$

The discrete surface integrals for $\vec{\mathcal{U}}$ are

$$\begin{aligned}
\oint_{\partial E}\phi\,U^*\,\vec{n}_1\,\mathrm{dS} &\approx -\,\delta_{i0}\,U^*_{i0}y_{\eta_{i0}}\omega_j + \delta_{iN}\,U^*_{iN}y_{\eta_{i0}}\omega_j - \delta_{j0}\,U^*_{0j}y_{\xi_{0j}}\omega_i + \delta_{jN}\,U^*_{Nj}y_{\xi_{Nj}}\omega_i, \\
\oint_{\partial E}\phi\,U^*\,\vec{n}_2\,\mathrm{dS} &\approx -\,\delta_{i0}\,U^*_{i0}x_{\eta_{i0}}\omega_j + \delta_{iN}\,U^*_{iN}x_{\eta_{i0}}\omega_j - \delta_{j0}\,U^*_{0j}x_{\xi_{0j}}\omega_i + \delta_{jN}\,U^*_{Nj}x_{\xi_{Nj}}\omega_i.
\end{aligned} \tag{5.112}$$

We summarize the ESDGSEM with artificial viscosity and define full discretizations of all the new viscous terms and gradient equations in Theorem 5. Furthermore, we proof that the resulting method is entropy stable.

Theorem 5 (Entropy stability of ESDGSEM with artificial viscosity)**.** *The ESDGSEM* (5.76) *for the shallow water equations with additional viscous terms as in* (5.107) *is given by*

$$J\vec{W}_t + \vec{\mathcal{L}}_\xi + \vec{\mathcal{L}}_\eta = \vec{\tilde{S}} + \vec{\mathcal{L}}^v_\xi + \vec{\mathcal{L}}^v_\eta. \tag{5.113}$$

The viscous terms $\vec{\mathcal{L}}^v_\xi$, $\vec{\mathcal{L}}^v_\eta$ are discretized in strong form by

$$\begin{aligned}
(\vec{\mathcal{L}}^v_\xi)_{ij} &= \sum_{m=0}^{N} D_{im}(\vec{\tilde{F}}^v)_{mj} + \frac{1}{\omega_i}\left(\delta_{iN}\vec{\tilde{F}}^{v,*}_{Nj} - \delta_{i0}\vec{\tilde{F}}^{v,*}_{0j}\right) - \frac{1}{\omega_i}\left(\delta_{iN}\vec{\tilde{F}}^{v}_{Nj} - \delta_{i0}\vec{\tilde{F}}^{v}_{0j}\right), \\
(\vec{\mathcal{L}}^v_\eta)_{ij} &= \sum_{m=0}^{N} D_{jm}(\vec{\tilde{G}}^v)_{im} + \frac{1}{\omega_j}\left(\delta_{Nj}\vec{\tilde{G}}^{v,*}_{iN} - \delta_{0j}\vec{\tilde{G}}^{v,*}_{i0}\right) - \frac{1}{\omega_j}\left(\delta_{Nj}\vec{\tilde{G}}^{v}_{iN} - \delta_{0j}\vec{\tilde{G}}^{v}_{i0}\right).
\end{aligned} \tag{5.114}$$

The curvilinear viscous fluxes are defined according to BR1 [8] as

$$\begin{aligned}
\vec{\tilde{F}}^v &= y_\eta\vec{F}^v - x_\eta\vec{G}^v, \\
\vec{\tilde{G}}^v &= -y_\xi\vec{F}^v + x_\xi\vec{G}^v,
\end{aligned} \tag{5.115}$$

with viscous fluxes

$$\begin{aligned}
\vec{F}^v &= h\,\epsilon\,(\mathcal{U}_1,\,\mathcal{V}_1)^T, \\
\vec{G}^v &= h\,\epsilon\,(\mathcal{U}_2,\,\mathcal{V}_2)^T.
\end{aligned} \tag{5.116}$$

The viscous flux interface coupling is computed by the arithmetic mean,

$$\begin{aligned}\vec{\vec{F}}^{v,*} &= \left\{\left\{\vec{\vec{F}}^{v}\right\}\right\},\\ \vec{\vec{G}}^{v,*} &= \left\{\left\{\vec{\vec{G}}^{v}\right\}\right\}.\end{aligned} \tag{5.117}$$

The explicit formulas for the discretization of the gradients $\vec{\mathcal{U}} = \nabla u$ and $\vec{\mathcal{V}} = \nabla v$ are given by

$$\begin{aligned}(\mathcal{U}_1)_{ij} =&(y_\eta)_{ij}\sum_{m=0}^{N}\hat{D}_{im}u_{mj} - (y_\xi)_{ij}\sum_{m=0}^{N}\hat{D}_{jm}u_{im}\\ &+\frac{(y_\eta)_{ij}}{\omega_i}\left(\delta_{iN}U^*_{Nj} - \delta_{i0}U^*_{0j}\right) + \frac{(y_\xi)_{ij}}{\omega_j}\left(\delta_{Nj}U^*_{iN} - \delta_{0j}U^*_{i0}\right)\\ (\mathcal{U}_2)_{ij} =&-(x_\eta)_{ij}\sum_{m=0}^{N}\hat{D}_{im}u_{mj} + (x_\xi)_{ij}\sum_{m=0}^{N}\hat{D}_{jm}u_{im}\\ &+\frac{(x_\eta)_{ij}}{\omega_i}\left(\delta_{iN}U^*_{Nj} - \delta_{i0}U^*_{0j}\right) + \frac{(x_\xi)_{ij}}{\omega_j}\left(\delta_{Nj}U^*_{iN} - \delta_{0j}U^*_{i0}\right)\\ (\mathcal{V}_1)_{ij} =&(y_\eta)_{ij}\sum_{m=0}^{N}\hat{D}_{im}v_{mj} - (y_\xi)_{ij}\sum_{m=0}^{N}\hat{D}_{jm}v_{im}\\ &+\frac{(y_\eta)_{ij}}{\omega_i}\left(\delta_{iN}V^*_{Nj} - \delta_{i0}V^*_{0j}\right) + \frac{(y_\xi)_{ij}}{\omega_j}\left(\delta_{Nj}V^*_{iN} - \delta_{0j}V^*_{i0}\right)\\ (\mathcal{V}_2)_{ij} =&-(x_\eta)_{ij}\sum_{m=0}^{N}\hat{D}_{im}v_{mj} + (x_\xi)_{ij}\sum_{m=0}^{N}\hat{D}_{jm}v_{im}\\ &+\frac{(x_\eta)_{ij}}{\omega_i}\left(\delta_{iN}V^*_{Nj} - \delta_{i0}V^*_{0j}\right) + \frac{(x_\xi)_{ij}}{\omega_j}\left(\delta_{Nj}V^*_{iN} - \delta_{0j}V^*_{i0}\right),\end{aligned} \tag{5.118}$$

where the numerical states U^* and V^* are chosen according to BR1 as the average values

$$\begin{aligned}U^* &= \{\{u\}\},\\ V^* &= \{\{v\}\}.\end{aligned} \tag{5.119}$$

The ESDGSEM with artificial viscosity (5.113) *and viscous terms discretized as in* (5.114) *and* (5.118) *is entropy stable.*

Proof. In [81] the authors show that viscous terms according to Bassi and Rebay [8] are entropy stable if the viscous fluxes can be rewritten as the product of a symmetric, positive definite block matrix $\mathcal{B}^\epsilon$ and the gradient of the entropy variables

$$\overset{\leftrightarrow}{f}{}^{v}(\vec{w}, \nabla\vec{w}) = \mathcal{B}^\epsilon\nabla\vec{q}, \tag{5.120}$$

with state vectors $\overset{\leftrightarrow}{f^v} = \begin{pmatrix} \vec{f}^v \\ \vec{g}^v \end{pmatrix} \in \mathbb{R}^6$ and $\nabla\vec{q} = \begin{pmatrix} \vec{q}_x \\ \vec{q}_y \end{pmatrix} \in \mathbb{R}^6$. In the case of the shallow water equations, the block matrix $\mathcal{B}^\epsilon$ can be expressed as

$$\mathcal{B}^\epsilon = \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{pmatrix} \in \mathbb{R}^{6\times 6}. \tag{5.121}$$

The entropy stability requirement for the block matrix $\mathcal{B}^\epsilon$ is then that each block $\mathbf{B}_{ij}$ is symmetric

$$\mathbf{B}^\epsilon_{ij} = \left(\mathbf{B}^\epsilon_{ji}\right)^T, \tag{5.122}$$

and positive (semi-)definite

$$\sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial \vec{q}}{\partial x_i}^T \mathbf{B}^\epsilon_{ij} \frac{\partial \vec{q}}{\partial x_j} \geq 0, \qquad \forall \vec{q}. \tag{5.123}$$

The choice of gradient variables and fluxes in (5.107) make the block matrix here very simple

$$\mathcal{B}^\epsilon = \epsilon h \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5.124}$$

We verify requirement (5.123) by noting

$$\begin{aligned} \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{\partial \vec{q}}{\partial x_i}^T \mathbf{B}^\epsilon_{ij} \frac{\partial \vec{q}}{\partial x_j} &= (\vec{q}_x)^T \mathbf{B}^\epsilon_{11}\, \vec{q}_x + (\vec{q}_y)^T \mathbf{B}^\epsilon_{22}\, \vec{q}_y \\ &= \epsilon\, h \left(u_x^2 + v_x^2 + u_y^2 + v_y^2\right) \geq 0. \end{aligned} \tag{5.125}$$

■

Remark 5. Introducing artificial viscosity to the system, as in Theorem 5, requires an additional time step condition for the diffusive part [79, 162]. We denote the original advective time step by Δt^a and the new viscous time step by Δt^v. We also introduce a new parameter, DFL, that works analogous to the CFL parameter for the advective time step. The two time steps are then computed by

$$\begin{aligned} \Delta t^v &= \frac{\text{DFL}}{\lambda^v_{\max}} \left(\frac{\min \Delta x}{N+1}\right)^2, \\ \Delta t^a &= \frac{\text{CFL}}{\lambda_{\max}} \frac{\min \Delta x}{N+1}, \end{aligned} \tag{5.126}$$

where $\lambda_{\max}$ and $\lambda^v_{\max}$ denote the maximum eigenvalues of the Jacobians of the advective and diffusion fluxes, respectively. Then, the time step for the computation is taken as the minimum of viscous and advective time steps:

$$\Delta t := \min\left(\Delta t^v, \Delta t^a\right). \tag{5.127}$$

The viscosity parameter ϵ in (5.107) is dynamically computed for each element similar to the works of Persson and Peraire [202]. It is based on a smoothness measure of an element local indicator quantity. We usually choose the indicator based on our nodal DG approximation by $Q_{ij} = h_{ij}$ but other choices such as $Q_{ij} = h^2_{ij}$ work as well and the optimal choice may well be problem dependent. We transform the nodal indicator Q to modal space $\hat{Q}$ by

$$\hat{Q}_{ij} = \sum_{i=0}^{N}\sum_{j=0}^{N} V^{-1}_{ij} Q_{ij} V^{-1}_{ji}, \tag{5.128}$$

with Vandermonde matrix $\mathbf{V}$ defined by

$$\begin{aligned} V_{ij} &= L_j(\xi^{GL}_i)\sqrt{j+0.5}, \\ V^{-1}_{ij} &= \left(\ell_j, \tilde{L}_i\right)_{L^2} \approx \sum_{l=0}^{N} L_i(\xi^G_l)\,\ell^{GL}_j(x^G_i)\,\omega^G_l\,\sqrt{i+0.5}, \end{aligned} \tag{5.129}$$

where L_i is the i-the Legendre polynomial, ℓ^{GL}_i the i-th Lagrange polynomial based on Legendre-Gauss-Lobatto nodes and ξ^G_i the Legendre-Gauss nodes. The scaled Legendre polynomials are $\tilde{L}_i = L_i\sqrt{i+0.5}$. The ω^G are the Legendre-Gauss quadrature weights. We compute shock indicators similar to [202] by

$$\sigma_{dof} = \log_{10}\left(\max\left(\frac{(Q-\tilde{Q}, Q-\tilde{Q})_{L^2}}{(Q,Q)_{L^2}}, \frac{(\tilde{Q}-\tilde{\tilde{Q}}, \tilde{Q}-\tilde{\tilde{Q}})_{L^2}}{(\tilde{Q},\tilde{Q})_{L^2}}\right)\right), \tag{5.130}$$

with

$$\begin{aligned} \tilde{Q} &:= \sum_{i,j=0}^{N-1} \hat{Q}_{ij}\tilde{L}_i\tilde{L}_j, \\ \tilde{\tilde{Q}} &:= \sum_{i,j=0}^{N-2} \hat{Q}_{ij}\tilde{L}_i\tilde{L}_j. \end{aligned} \tag{5.131}$$

With these definitions (5.130) can be simplified to

$$\sigma_{dof} = \log_{10}\left(\max\left(I_1, I_2\right)\right), \tag{5.132}$$

with simplified smoothness measurements

$$I_1 := \frac{\sum_{i=0}^{N-1}\left(\hat{Q}^2_{iN} + \hat{Q}^2_{Ni}\right) + \hat{Q}^2_{NN}}{\sum_{i,j=0}^{N}\hat{Q}^2_{ij}}, \tag{5.133}$$

and

$$I_2 := \frac{\sum_{i=0}^{N-2}\left(\hat{Q}^2_{i(N-1)} + \hat{Q}^2_{(N-1)i}\right) + \hat{Q}^2_{(N-1)(N-1)}}{\sum_{i,j=0}^{N-1}\hat{Q}^2_{ij}}. \tag{5.134}$$

The smoothness indicator σ_{dof} is used to determine the amount of viscosity applied in every element individually by setting

$$\epsilon = \begin{cases} 0, & \text{if } \sigma_{dof} \leq \sigma_{min}, \\ \frac{1}{2}\epsilon_0\Delta^\epsilon, & \text{if } \sigma_{dof} \leq \sigma_{min} \ , \\ \epsilon_0, & \text{else.} \end{cases} \tag{5.135}$$

with

$$\Delta^\epsilon := 1.0 + \sin\left(\frac{\pi(\sigma_{dof} - \frac{1}{2}(\sigma_{max} + \sigma_{min}))}{\sigma_{max} - \sigma_{min}}\right). \tag{5.136}$$

We note that good choices for the parameters e_0, σ_{max} and σ_{min} are problem dependent and subject to some experimentation. However, we did find that the setting

$$\begin{aligned} e_0 &= 0.1, \\ \sigma_{max} &= -3.5, \\ \sigma_{min} &= -8.0, \end{aligned} \tag{5.137}$$

often leads to good results. For more information on detecting troubled elements we refer to [130].

5.5. Wet/dry treatment

The added artificial viscosity from Section 5.4 greatly reduces numerical oscillations. However, in nearly-dry regions, even the smallest oscillations may render a numerical scheme unstable. We need an additional safety net, that strictly enforces the positivity of the water height h without destroying accuracy, conservation, well-balancedness or entropy-stability of the ESDGSEM.

A mechanism to prevent unphysical values are the positivity preserving limiters proposed in the literature, e.g., [204, 271, 265, 264]. The widely used positivity limiter for the shallow water equations by Xing [265] is based on a linear scaling around element averages. As such, it is dependent on non-negative mean water heights in all elements at each time step. For certain numerical fluxes, it can then be proven that non-negative mean water heights are preserved for one Euler time step. This proof relies on the numerical flux to preserve non-negative water heights for finite volume schemes. Numerical fluxes that have this property are called positivity preserving [271]. For Cartesian meshes, this property translates immediately to DG methods [265]. In [271] it is proven that the Lax-Friedrichs numerical flux [204] is positivity preserving and the same is noted for the Godunov flux [57], Boltzmann type flux [203] and the Harten-Lax-van-Leer flux [105]. The positivity preserving property is shown for Euler time integration but extends to

SSPRK methods, as given by (5.104), as these are convex combinations of Euler time steps [219].

The contribution of this work is the proof that the entropy stable numerical flux of the ESDGSEM given in (5.100) is positivity preserving in the sense that non-negative mean water heights are preserved for one Euler time step. On Cartesian meshes, it is possible to prove this similarly to [271], where the update of the mean water height is written in finite volume form. Due to generally different Jacobians of the curvilinear mappings and different normals on opposing sides of an element, this is not as straightforward on curved meshes. We directly prove the positivity preservation for curved quadrilateral elements in Lemma 16. Also, we verify that the positivity preserving limiter is entropy stable in Lemma 17. Both results are then summarized in Theorem 6.

We now consider one Euler time step of the ESDGSEM and show that non-negative water heights are preserved if the time step is sufficiently small. For notational convenience, we denote values on the interfaces by $W_{j,s}$ (as opposed to W_{ij} for internal nodal values) to denote the value of quantity W at node $j \in \{0,\dots,N\}$ on interface $s \in \{1,\dots,4\}$, where s denotes the element local side number. We also introduce the surface Jacobian $\mathcal{J}^{\text{surf}}$ on $\xi = \pm 1$ ($s = 2,4$) and $\eta = \pm 1$ ($s = 1,3$) interfaces by

$$
\begin{aligned}
\mathcal{J}^{\text{surf}} &:= \sqrt{y_\xi y_\xi + x_\xi x_\xi}, \qquad \text{for } \eta = \pm 1\ ,\\
\mathcal{J}^{\text{surf}} &:= \sqrt{y_\eta y_\eta + x_\eta x_\eta}, \qquad \text{for } \xi = \pm 1\ .
\end{aligned}
\tag{5.138}
$$

Lemma 16 (Preservation of non-negative mean water heights in ESDGSEM)**.** *If the water height h is non-negative for all LGL nodes in all elements, then the average water height in the next time step is non-negative for all elements under the additional time step restrictions*

$$
\begin{aligned}
\Delta t &\le \frac{\omega_0\, a_{j,s}}{\left(A_{j,s} + 2\,\{\{\tilde{u}\}\}_{j,s}\right)},\\
\Delta t &\le \left|\frac{\omega_0\, a_{j,s}\, g\, h_{j,s}}{\{\{c\}\}_{j,s}\, B_{j,s}[\![\tilde{u}]\!]_{j,s}}\right|, \qquad \textit{only if } h_{j,s} > 0,
\end{aligned}
\tag{5.139}
$$

where we introduce the rotated normal velocity

$$
\tilde{u} := n_x u + n_y v \tag{5.140}
$$

for all edge nodes $j = 0,\dots,N$ on all element sides $s = 1,\dots,4$, and

$$
\begin{aligned}
A &:= |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,| + |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,|,\\
B &:= |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,| - |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,|.
\end{aligned}
\tag{5.141}
$$

For the Legendre Gauss Lobatto nodes, the quadrature weight ω_0 is given by

$$
\omega_0 = \frac{1}{2}N(N-1). \tag{5.142}
$$

We also have geometric scaling factors on the interfaces given by

$$
a_{j,s} := \frac{\mathcal{J}_{j,s}}{\mathcal{J}^{surf}_{j,s}}, \tag{5.143}
$$

with volume and surface Jacobians defined in (3.49) *and* (5.138).

Proof. Since the shallow water equations are rotationally invariant, we can rotate the velocities in normal direction and evaluate the numerical flux in x-direction with rotated velocities

$$\begin{aligned} \tilde{u} &:= n_x u + n_y v, \\ \tilde{v} &:= t_x u + t_y v = -n_y u + n_x v. \end{aligned} \tag{5.144}$$

Afterwards, we rotate back to obtain the numerical fluxes in normal direction. Denoting the rotated conservative variables by $\vec{\tilde{W}}$, the formulas for computing the numerical fluxes in normal direction are

$$\begin{aligned} \tilde{F}_1^{*,es}(\vec{W}^+, \vec{W}^-, \vec{n}) &= F_1^{*,es}(\vec{\tilde{W}}^+, \vec{\tilde{W}}^-), \\ \tilde{F}_2^{*,es}(\vec{W}^+, \vec{W}^-, \vec{n}) &= n_x F_2^{*,es}(\vec{\tilde{W}}^+, \vec{\tilde{W}}^-) + t_x F_3^{*,es}(\vec{\tilde{W}}^+, \vec{\tilde{W}}^-), \\ \tilde{F}_3^{*,es}(\vec{W}^+, \vec{W}^-, \vec{n}) &= n_y F_2^{*,es}(\vec{\tilde{W}}^+, \vec{\tilde{W}}^-) + t_y F_3^{*,es}(\vec{\tilde{W}}^+, \vec{\tilde{W}}^-). \end{aligned} \tag{5.145}$$

We want to guarantee positive water heights and, thus, only consider the numerical flux contributions for the continuity equation. Thus, we aim to find a compact expression for the first entry of the numerical flux in normal direction, $\vec{\tilde{F}}^{*,es}$, given by

$$\tilde{F}_1^{*,es} = \left(F_1^{*,es}, G_1^{*,es}\right) \cdot \vec{n} = F_1^{*,es}(\vec{\tilde{W}}^+, \vec{\tilde{W}}^-), \tag{5.146}$$

where the numerical flux in physical x direction is defined as

$$\vec{F}^{*,es} = \vec{F}^{*,ec} - \frac{1}{2}\mathbf{R}_f \, |\mathbf{\Lambda}| \, \mathbf{R}_f^T [\![\, \vec{q}\,]\!]. \tag{5.147}$$

We evaluate the numerical flux in x-direction with rotated conservative variables, $\vec{\tilde{W}}$, which yields

$$\vec{F}^{*,es}(\vec{\tilde{W}}^+, \vec{\tilde{W}}^-) = \begin{pmatrix} \{\{h\}\}\,\{\{\tilde{u}\}\} \\ \{\{h\}\}\,\{\{\tilde{u}\}\}^2 + \frac{1}{2}\,g\,\{\{h^2\}\} \\ \{\{h\}\}\,\{\{\tilde{u}\}\}\,\{\{\tilde{v}\}\} \end{pmatrix} - \frac{1}{2}\tilde{\mathbf{R}}\,|\tilde{\mathbf{\Lambda}}|\,\tilde{\mathbf{R}}^T [\![\, \vec{\tilde{q}}\,]\!], \tag{5.148}$$

with matrix of right eigenvectors

$$\tilde{\mathbf{R}} = \begin{pmatrix} 1 & 0 & 1 \\ \{\{\tilde{u}\}\} + \{\{c\}\} & 0 & \{\{\tilde{u}\}\} - \{\{c\}\} \\ \{\{\tilde{v}\}\} & 1 & \{\{\tilde{v}\}\} \end{pmatrix}, \tag{5.149}$$

and scaled diagonal eigenvalue matrix

$$|\tilde{\mathbf{\Lambda}}| = \frac{1}{2g} \begin{pmatrix} |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,| & 0 & 0 \\ 0 & 2g|\,\{\{h\}\}\,\{\{\tilde{u}\}\}\,| & 0 \\ 0 & 0 & |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,| \end{pmatrix}, \tag{5.150}$$

with wave celerity $c = \sqrt{gh}$. We compute the first row of the matrix product of the dissipation term by multiplying with just the first row $\tilde{\mathbf{R}}_1$ to find

$$\begin{aligned}
&2g\tilde{\mathbf{R}}_1\,|\tilde{\mathbf{\Lambda}}|\,\tilde{\mathbf{R}}^T\\
&= \begin{pmatrix}1, & 0, & 1\end{pmatrix}\begin{pmatrix}|\,\{\{\tilde{u}\}\}+\{\{c\}\}\,| & 0 & 0\\ 0 & 2g|\,\{\{h\}\}\,\{\{\tilde{u}\}\}\,| & 0\\ 0 & 0 & |\,\{\{\tilde{u}\}\}-\{\{c\}\}\,|\end{pmatrix}\begin{pmatrix}1 & \{\{\tilde{u}\}\}+\{\{c\}\} & \{\{\tilde{v}\}\}\\ 0 & 0 & 1\\ 1 & \{\{\tilde{u}\}\}-\{\{c\}\} & \{\{\tilde{v}\}\}\end{pmatrix}\\
&= \begin{pmatrix}|\,\{\{\tilde{u}\}\}+\{\{c\}\}\,|, & 0, & |\,\{\{\tilde{u}\}\}-\{\{c\}\}\,|\end{pmatrix}\begin{pmatrix}1 & \{\{\tilde{u}\}\}+\{\{c\}\} & \{\{\tilde{v}\}\}\\ 0 & 0 & 1\\ 1 & \{\{\tilde{u}\}\}-\{\{c\}\} & \{\{\tilde{v}\}\}\end{pmatrix}\\
&= \begin{pmatrix}A, & \{\{\tilde{u}\}\}\,A+\{\{c\}\}\,B, & \{\{\tilde{v}\}\}\,A\end{pmatrix},
\end{aligned} \tag{5.151}$$

with

$$\begin{aligned}
A &:= |\,\{\{\tilde{u}\}\}+\{\{c\}\}\,| + |\,\{\{\tilde{u}\}\}-\{\{c\}\}\,|,\\
B &:= |\,\{\{\tilde{u}\}\}+\{\{c\}\}\,| - |\,\{\{\tilde{u}\}\}-\{\{c\}\}\,|.
\end{aligned} \tag{5.152}$$

Multiplying by the jump in rotated entropy variables, $[\![\,\vec{\tilde{q}}\,]\!]$, we find the first entry of $\frac{1}{2}\tilde{\mathbf{R}}\,|\tilde{\mathbf{\Lambda}}|\,\tilde{\mathbf{R}}^T[\![\,\vec{\tilde{q}}\,]\!]$

$$\begin{aligned}
2g\left(\tilde{\mathbf{R}}\,|\tilde{\mathbf{\Lambda}}|\,\tilde{\mathbf{R}}^T[\![\,\vec{\tilde{q}}\,]\!]\right)_1 &= \begin{pmatrix}A, & \{\{\tilde{u}\}\}\,A+\{\{c\}\}\,B, & \{\{\tilde{v}\}\}\,A\end{pmatrix}\begin{pmatrix}g[\![h+b]\!]-\frac{1}{2}[\![\tilde{u}^2]\!]-\frac{1}{2}[\![\tilde{v}^2]\!]\\ [\![\tilde{u}]\!]\\ [\![\tilde{v}]\!]\end{pmatrix}\\
&= A\left(g[\![h+b]\!]-\{\{\tilde{u}\}\}\,[\![\tilde{u}]\!]-\{\{\tilde{v}\}\}\,[\![\tilde{v}]\!]\right)+\left(\{\{\tilde{u}\}\}\,A+\{\{c\}\}\,B\right)[\![\tilde{u}]\!]+\{\{\tilde{v}\}\}\,A[\![\tilde{v}]\!]\\
&= gA[\![h+b]\!]+\{\{c\}\}\,B[\![\tilde{u}]\!].
\end{aligned} \tag{5.153}$$

Thus, we can express the first entry of the entropy stable numerical flux in terms of the rotated velocities

$$F_1^{*,es}(\vec{\tilde{W}}^+,\vec{\tilde{W}}^-) = \{\{h\}\}\,\{\{\tilde{u}\}\} - \frac{1}{4g}\left(A[\![gh+gb]\!]+\{\{c\}\}\,B[\![\tilde{u}]\!]\right). \tag{5.154}$$

where $\tilde{u} = n_xu + n_yv$. As the water height is a conserved quantity in the ESDGSEM, we can write the update of the element average water height in one Euler time step as

$$\begin{aligned}
\overline{h}^{t_{n+1}} &= \overline{h}^{t_n} - \frac{\Delta t}{|E|}\sum_{s=1}^{4}\sum_{j=0}^{N}\omega_j\mathcal{J}_{j,s}^{\text{surf}}\tilde{F}_1^{*,es}\left(W_{j,s}^{\text{int}},W_{j,s}^{\text{ext}},n_{j,s}\right)\\
&= \overline{h}^{t_n} - \frac{\Delta t}{|E|}\sum_{s=1}^{4}\sum_{j=0}^{N}\omega_j\mathcal{J}_{j,s}^{\text{surf}}F_1^{*,es}\left(\tilde{W}_{j,s}^{\text{int}},\tilde{W}_{j,s}^{\text{ext}}\right),
\end{aligned} \tag{5.155}$$

where $\mathcal{J}_{j,s}^{\text{surf}}$ is the surface Jacobian at node j on interface s defined in (5.138). We can also write the average water height $\overline{h}^{t_n}$ as

$$\begin{aligned}
\overline{h}^{t_n} &= \frac{1}{|E|}\sum_{j=0}^{N}\sum_{i=0}^{N} h_{ij}\mathcal{J}_{ij}\omega_i\omega_j \\
&= \frac{1}{|E|}\sum_{i=1}^{N-1}\sum_{j=1}^{N-1} h_{ij}\mathcal{J}_{ij}\omega_i\omega_j + \frac{1}{2}\frac{1}{|E|}\sum_{s=1}^{4}\sum_{j=1}^{N-1} h_{j,s}\mathcal{J}_{j,s}\omega_0\omega_j + \frac{1}{2}\frac{1}{|E|}\sum_{s=1}^{4}\sum_{j=0}^{N} h_{j,s}\mathcal{J}_{j,s}\omega_0\omega_j \\
&= \frac{1}{|E|}\sum_{i=1}^{N-1}\sum_{j=1}^{N-1} h_{ij}\mathcal{J}_{ij}\omega_i\omega_j + \frac{1}{2}\frac{1}{|E|}\sum_{s=1}^{4}\sum_{j=1}^{N-1} h_{j,s}\mathcal{J}_{j,s}\omega_0\omega_j \qquad (5.156)\\
&\quad + \frac{1}{2|E|}\sum_{j=0}^{N} h_{j,3}\mathcal{J}_{j,3}\omega_0\omega_j + \frac{1}{2|E|}\sum_{j=0}^{N} h_{j,1}\mathcal{J}_{j,1}\omega_0\omega_j \\
&\quad + \frac{1}{2|E|}\sum_{i=0}^{N} h_{j,4}\mathcal{J}_{j,4}\omega_0\omega_j + \frac{1}{2|E|}\sum_{i=0}^{N} h_{j,2}\mathcal{J}_{j,2}\omega_0\omega_j.
\end{aligned}$$

Inserting the new expression for the average water height (5.156) into the update scheme (5.155) we find

$$\begin{aligned}
\overline{h}^{t_{n+1}} &= \frac{1}{|E|}\sum_{i=1}^{N-1}\sum_{j=1}^{N-1} h_{ij}\mathcal{J}_{ij}\omega_i\omega_j + \frac{1}{2}\frac{1}{|E|}\sum_{s=1}^{4}\sum_{j=1}^{N-1} h_{j,s}\mathcal{J}_{j,s}\omega_0\omega_j \\
&+ \frac{1}{|E|}\sum_{j=0}^{N}\mathcal{J}_{j,1}\omega_0\omega_j\left[\frac{1}{2}h_{j,1} - \frac{\Delta t}{\omega_0\, a_{j,1}}F_1^{*,es}\left(\tilde{W}_{j,1}^{\text{int}}, \tilde{W}_{j,1}^{\text{ext}}\right)\right] \\
&+ \frac{1}{|E|}\sum_{j=0}^{N}\mathcal{J}_{j,3}\omega_0\omega_j\left[\frac{1}{2}h_{j,3} - \frac{\Delta t}{\omega_0\, a_{j,3}}F_1^{*,es}\left(\tilde{W}_{j,3}^{\text{int}}, \tilde{W}_{j,3}^{\text{ext}}\right)\right] \qquad (5.157)\\
&+ \frac{1}{|E|}\sum_{j=0}^{N}\mathcal{J}_{j,2}\omega_0\omega_j\left[\frac{1}{2}h_{j,2} - \frac{\Delta t}{\omega_0\, a_{j,2}}F_1^{*,es}\left(\tilde{W}_{j,2}^{\text{int}}, \tilde{W}_{j,2}^{\text{ext}}\right)\right] \\
&+ \frac{1}{|E|}\sum_{j=0}^{N}\mathcal{J}_{j,4}\omega_0\omega_j\left[\frac{1}{2}h_{j,4} - \frac{\Delta t}{\omega_0\, a_{j,4}}F_1^{*,es}\left(\tilde{W}_{j,4}^{\text{int}}, \tilde{W}_{j,4}^{\text{ext}}\right)\right],
\end{aligned}$$

with $a_{j,s} := \frac{\mathcal{J}_{j,s}}{\mathcal{J}_{j,s}^{\text{surf}}}$. We note that for the special case of uniform Cartesian meshes this factor is simply $a_{j,s} = \Delta y$ for $s = 1, 3$ and $a_{j,s} = \Delta x$ for $s = 2, 4$. The first two sums are clearly non-negative for meshes with positive Jacobians. We proceed to examine the interface terms

$$\frac{1}{2}h_{j,s} - \frac{\Delta t}{\omega_0\, a_{j,s}}F_1^{*,es}\left(\tilde{W}_{j,s}^{\text{int}}, \tilde{W}_{j,s}^{\text{ext}}\right), \qquad s = 1, \dots, 4 \quad . \qquad (5.158)$$

We can do this for an arbitrary node j, side s, and only need to distinguish between internal values W^- and external values W^+ on the interface. We thus omit the indices j and s in the following steps. By inserting the compact expression for $F_1^{*,es}$ from (5.154) and assuming continuous bottom topographies across element interfaces we find

$$\begin{aligned}
&\frac{1}{2}h^- - \frac{1}{4g}\frac{\Delta t}{\omega_0\, a}\left(4g\,\{\{h\}\}\,\{\{\tilde{u}\}\} - gA[\![h]\!] - \{\{c\}\}\,B[\![\tilde{u}]\!]\right)\\
&= \frac{1}{2}h^- - \frac{1}{4g}\frac{\Delta t}{\omega_0\, a}\Big(gh^+\left(2\,\{\{\tilde{u}\}\} - A\right) + gh^-\left(2\,\{\{\tilde{u}\}\} + A\right) - \{\{c\}\}\,B[\![\tilde{u}]\!]\Big)\\
&= \frac{1}{4}\frac{\Delta t}{\omega_0\, a}h^+\left(A - 2\,\{\{\tilde{u}\}\}\right) + \frac{1}{4}h^-\left(1 - \frac{\Delta t}{\omega_0\, a}\left(A + 2\,\{\{\tilde{u}\}\}\right)\right) + \frac{1}{4}\left(h^- + \frac{\Delta t}{g\,\omega_0\, a}\,\{\{c\}\}\,B[\![\tilde{u}]\!]\right)
\end{aligned} \tag{5.159}$$

We examine the three terms in (5.159) individually. The first term is always non-negative, since

$$A = |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,| + |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,| \geq 2|\,\{\{\tilde{u}\}\}\,|. \tag{5.160}$$

For the second term we require an additional time step condition. From

$$\left(1 - \frac{\Delta t}{\omega_0\, a}\left(A + 2\,\{\{\tilde{u}\}\}\right)\right) \overset{!}{\geq} 0, \tag{5.161}$$

we find

$$\Delta t \overset{!}{\leq} \frac{\omega_0\, a}{\left(A + 2\,\{\{\tilde{u}\}\}\right)}. \tag{5.162}$$

For the third term, we cannot generally factor out h^-, so we need to treat this carefully. We need to show that the last term is not negative in the cases $h^- = 0$ and $h^+ = 0$ as well as in the fully wet case where both water heights are positive. In the case $h^- = 0$, we also have $\tilde{u}^- = 0$ and thus we see $[\![\tilde{u}]\!] = \tilde{u}^+$, and the whole term becomes

$$\frac{\Delta t}{g\,\omega_0\, a}\,\{\{c\}\}\,B\tilde{u}^+. \tag{5.163}$$

We note that $\{\{c\}\}$ is always non-negative, so we must examine the signs of B and $\tilde{u}^+$. The velocity can have an arbitrary sign but from the definition of B

$$B = |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,| - |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,| = \left|\frac{1}{2}\tilde{u}^+ + \{\{c\}\}\right| - \left|\frac{1}{2}\tilde{u}^+ - \{\{c\}\}\right|, \tag{5.164}$$

we see that the sign of B matches the sign of $\tilde{u}^+$ and thus the whole term is guaranteed non-negative for $h^- = 0$ and $h^+ \geq 0$. If $h^- > 0$ (and thus in general $\tilde{u}^- \neq 0$), we are allowed to factor out h^-. Then we require

$$\frac{\Delta t}{g\,\omega_0\, a}\frac{\{\{c\}\}\,B[\![\tilde{u}]\!]}{h^-} \overset{!}{\geq} -1. \tag{5.165}$$

This condition guarantees non negativity for the third term in (5.159) and can only be violated if $B[\![\tilde{u}]\!] < 0$. There are two different states where this is the case. Either we have $\tilde{u}^- > \tilde{u}^+$ and $\{\{\tilde{u}\}\} > 0$, which implies $\tilde{u}^- > 0$. Or, alternatively, we have $B < 0$

and $[\![\tilde{u}]\!] > 0$, which implies $0 > \tilde{u}^+ > \tilde{u}^-$. In either way, for $B[\![\tilde{u}]\!] < 0$, we can guarantee non negativity by enforcing the additional time step restriction

$$\Delta t \overset{!}{\leq} \left| \frac{g\,\omega_0\,a\,h^-}{\{\{c\}\}\,B[\![\tilde{u}]\!]} \right|. \tag{5.166}$$

This proof for one Euler time step extends to SSPRK methods as used in the ESDGSEM as seen in, e.g., [265]. ■

Lemma 16 guarantees a non-negative average water height in the next time step, but we still need to ensure point-wise non-negativity. This can be achieved with the positivity limiter applied to the shallow water equations by Xing et al. [265] and developed in [204, 271, 264] which is a linear scaling around the element average

$$\widehat{\vec{W}}_{ij} = \theta\left(\vec{W}_{ij} - \overline{\vec{W}}_E\right) + \overline{\vec{W}}_E, \tag{5.167}$$

where θ is computed by

$$\theta = \min\left(1, \frac{\overline{h}_E}{\overline{h}_E - m_E}\right), \tag{5.168}$$

and m_E is the minimum and $\overline{h}_E$ is the average water height in element E. The scaling is applied to the water height h and the discharges hu and hv with the same parameter θ based on the water height. Since we have shown that the average water height is guaranteed positive in Lemma 16, this limiter ensures positive water heights for all computation nodes. It can be shown that this limiter maintains high-order accuracy and is conservative [270]. We show that the positivity preserving limiter is also entropy stable in Lemma 17. As entropy stability is attained on a global level, it is sufficient to show that the positivity limiter applied to an element E does not increase the entropy for that element. Global entropy stability then follows immediately.

Lemma 17 (Entropy Stability of Positivity Preservation)**.** *An entropy stable method coupled with the positivity preserving limiter* (5.167) *is entropy stable.*

Proof. We prove this result in a similar fashion to Ranocha [206]. Let $\mathscr{E}(\vec{W})$ denote the discrete entropy within an element with solution polynomial $\vec{W}$. The limited value of the solution polynomial is denoted by $\widehat{\vec{W}}$ (5.167). Then

$$\begin{aligned}
\overline{\mathscr{E}\left(\widehat{\vec{W}}\right)} &= \frac{1}{|E|}\sum_{i=0}^{N}\sum_{j=0}^{N}\mathscr{E}\left(\widehat{\vec{W}}_{ij}\right) J_{ij}\,\omega_i\omega_j \\
&\overset{\mathscr{E}\text{ convex}}{\leq} \frac{1}{|E|}\theta\sum_{i=0}^{N}\sum_{j=0}^{N}\mathscr{E}(\vec{W}_{ij})J_{ij}\,\omega_i\omega_j + \frac{1}{|E|}(1-\theta)\sum_{i=0}^{N}\sum_{j=0}^{N}\mathscr{E}\left(\overline{\vec{W}}\right) J_{ij}\,\omega_i\omega_j \\
&= \frac{1}{|E|}\theta\sum_{i=0}^{N}\sum_{j=0}^{N}\mathscr{E}(\vec{W}_{ij})J_{ij}\,\omega_i\omega_j + (1-\theta)\mathscr{E}\left(\overline{\vec{W}}\right) \\
&\overset{\text{Jensen's inequality}}{\leq} \frac{1}{|E|}\theta\sum_{i=0}^{N}\sum_{j=0}^{N}\mathscr{E}(\vec{W}_{ij})J_{ij}\,\omega_i\omega_j + (1-\theta)\overline{\mathscr{E}(\vec{W})} \\
&= \theta\overline{\mathscr{E}(\vec{W})} + (1-\theta)\overline{\mathscr{E}(\vec{W})} \\
&= \overline{\mathscr{E}(\vec{W})}.
\end{aligned} \tag{5.169}$$

So, the entropy of the limited solution polynomial $\hat{W}$ is less or equal to the entropy of the original solution. It follows that the positivity limiter does not increase the entropy of the system and the method remains entropy stable. ■

We summarize the results of this section in Theorem 6.

Theorem 6 (ESDGSEM with positivity limiter)**.** *The ESDGSEM* (5.76) *combined with the positivity preserving limiter* (5.167) *and the additional time step restrictions* (5.139) *fulfills all the properties from Theorem 4 and also guarantees non-negative water heights for all LGL-nodes.*

Proof. We proved in Lemma 16 that preservation of non-negative water mean height is guaranteed if the water height in the previous time step is non-negative for all LGL nodes. The positivity limiter (5.167) then guarantees non-negative water height at all LGL nodes. Finally, in Lemma 17 it is shown that the positivity limiter is entropy stable. We note that other properties from Theorem 4 such as mass conservation and the well-balancedness are unaffected by the positivity limiting procedure. ■

Remark 6. Theorem 6 also holds when combining the positivity preserving limiter with the artificial viscosity shock capturing from Sec. 5.4 since no artificial viscosity is added to the continuity equation.

Remark 7. The positivity limiter does not affect the well-balanced property of the ES-DGSEM since the scaling will not be applied for the "lake at rest" test case. However, the capability of handling dry areas leads to a generalization called the "dry lake," defined by

$$\begin{aligned}
h &= \max\{H_{\text{const}} - b, 0\} \\
u &= v = 0,
\end{aligned} \tag{5.170}$$

where the bottom topography surpasses the constant water level, creating dry areas. If this leads to partially dry elements, the well-balanced property of the scheme is lost, as the proof relies strongly on the property $H = h + b =$ const and the consistency of the derivative operator $\mathbf{D}$. Retaining the well-balanced property for partly dry elements is a difficult challenge and subject to ongoing research. Strategies include adaptive mesh refinement or the development of different local derivative operators that account for the dry nodes within the element, e.g. [20].

6. GPU acceleration

6.1. Introduction to GPU computing

In the field of high performance computing (HPC), there is always a demand for as much computational processing power as possible in order to run large, well-resolved simulations in acceptable time frames. When the performance gains from increasing the CPU clock rates were saturated, intentions turned towards parallel computing on many cores in modern supercomputers. In recent years, GPUs have emerged as a competing, or complementing, hardware architecture due to their high performance and their cost and energy efficiency. Nowadays GPUs are frequently deployed in modern supercomputers as part of heterogeneous systems, combining CPUs and GPUs in each node. The number one ranking supercomputer is Summit at the Oak Ridge National Laboratory (ORNL). It features 4,356 nodes, where each one is equipped with two 22-core Power9 CPUs, and six NVIDIA Tesla V100 GPUs. Overall, four out of the five best performing supercomputers feature GPUs, with the exception being the Sunway TaihuLight in Jiangsu [239]. Other top supercomputers featuring GPUs in addition to their CPU structure are the Titan at Oak Ridge National Laboratory [144] or the Chinese Tianhe-1A [266] as well as the Sierra and the AI Bridging Cloud Infrastructure (ABCI). The GPU of choice in these supercomputers is typically the high-end scientific computing card Tesla V100. GPUs have evolved into the massively parallel processing platforms they are today due to their origin as renderers for 3D graphics in video games. In this context it was required to recalculate a large amount of pixels in real time. Thus, a single-thread program that refreshes one individual pixel was executed many times in parallel. The demand for ever faster graphics processors from the video game industry is still a primary driver behind the development of increasingly parallel GPUs [179]. The first unified graphics and computing GPU architecture was the GeForce 8800 in 2006 allowing the efficient execution of $12,288$ parallel threads on 128 processor cores [179]. The increased interest from the scientific computing community led to the development of dedicated general-purpose GPUs (GP-GPU), specifically with the launch of the NVIDIA Tesla series in 2007. Since then, NVIDIA has been releasing new GPUs for both demands, video games and scientific computing, on a regular basis. The line most suited for graphics rendering is named GTX whereas the scientific computing cards are the Tesla series. However, even the cards primarily developed for video game graphics are now very suitable for scientific computing purposes due to the increased video memory (VRAM) and floating point performance in single and double precision. This makes GPUs widely accessible to everyone for a relatively cheap entry price compared to buying competitively many CPU cores. Thus, GP-GPU computing has been widely adopted by researchers from varying fields [77]. The first Tesla GPUs were primarily constructed to perform single precision floating point operations and not even able to perform double precision operations at all. In 2009, the GT200 was given this capability but lacked competitive performance. Only about 12.5% of the single precision peak performance was achievable for double precision operations. The Fermi architecture from 2011 came with increased double precision performance at 50% of the single precision performance. The following two generations Kepler in 2012 and Maxwell in 2015 prioritized single precision performance once more, with double precision performances ranging from just 3% to 30%. The final breakthrough of double precision computing on GPUs finally arrived with the P100 GPUs of the Pascal generation in 2016 which delivered a steady 50% of single precision performance for double precision calculations. This level was retained for the V100 GPUs of 2017's Volta series. Performance of graphics cards are determined by several factors. Each GPU engine has a certain amount of CUDA cores with a certain clock frequency. Another crucial component determining the overall performance is the memory which again comes with a specific speed and bandwidth. For the Pascal generation's NVIDIA GTX 1080 there are 2560 CUDA cores with a base clock of 1607 MHz which can go up to 1733 MHz in boost mode. It is equipped with 8 GB of memory at bandwidth of 320 GB/s. The other card used in this work is the NVIDIA Tesla V100 and is a far more powerful GPU for scientific computing purposes. While the base clock is lower at just 1230 MHz (1380 in boost mode), it doubles the amount of single precision CUDA cores to 5120 and also offers 2560 double precision CUDA cores. It has 32 GB of video memory at a bandwidth of 900.0 GB/s, which is more than double the bandwidth of the GTX 1080's memory. Since memory bandwidth is often the bottleneck in computational performance, this is a huge improvement. The decreased base clock is, while important for gaming performance, not a huge factor for computational science purposes. The manufacturer also offers theoretical peak performance numbers for floating point operations. The gaming GPU GTX 1080 offers 8873 GFLOPS for single precision operators and 277.3 GFLOPS for double precision operations, while the scientific computing card Tesla V100 almost doubles the single precision performance with 14 TFLOPS. The double precision performance is at a whole other level with exactly half of the single precision performance. Thus, the Tesla V100's double precision performance is almost thirty times as high as the double precision performance of the GTX 1080. The Tesla V100 is arguably the best GPU for scientific computing to this date and even exceeds the Titan V in terms of available VRAM and memory bandwidth. This being said, the Tesla V100 costs more than three times as much, with a release price of $\$10,000$ compared to the \$3000 price of the Titan V, making the Titan V an attractive option as well. We show the development of CPU and GPU performance in terms of GFLOP/s at base clock in Figure 6.1 and the theoretical peak memory bandwidth in Figure 6.2.

GP-GPU programming is typically performed in either OpenCL or CUDA. While OpenCL is vendor independent and works with AMD cards as well, CUDA is exclusive to NVIDIA hardware. OpenCL is also a more general framework applicable to either CPU or GPU architectures. For this work, we will focus on CUDA and use its nomenclature. We point out that many of the concepts and key words have an OpenCL analogy. For

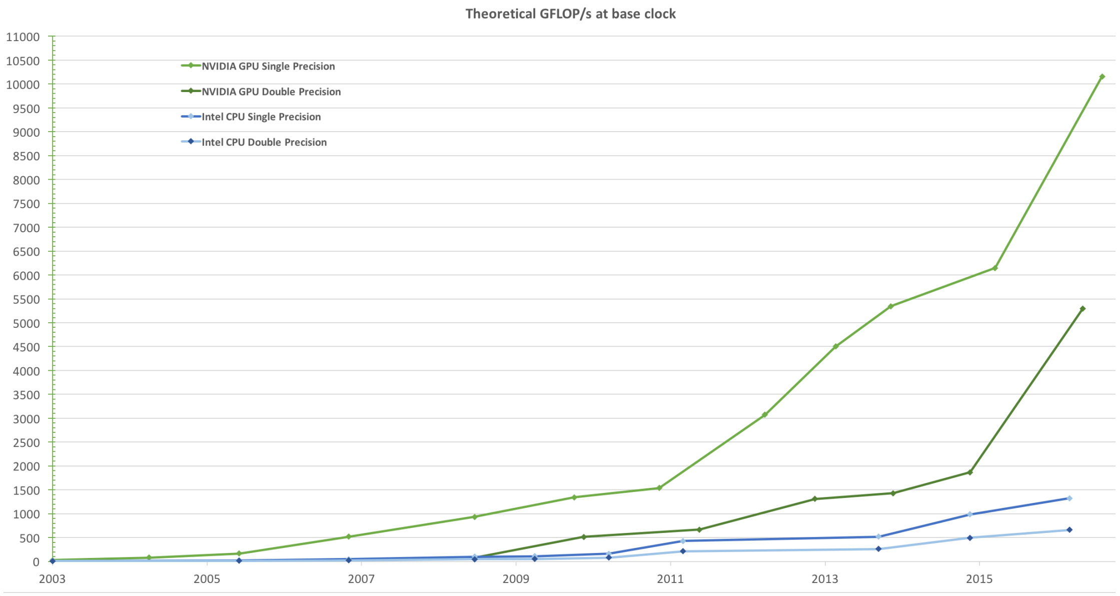


Figure 6.1.: Development of NVIDIA GPU flop performance over the different architectures. Graphic taken from the NVIDIA Programming Guide [188].

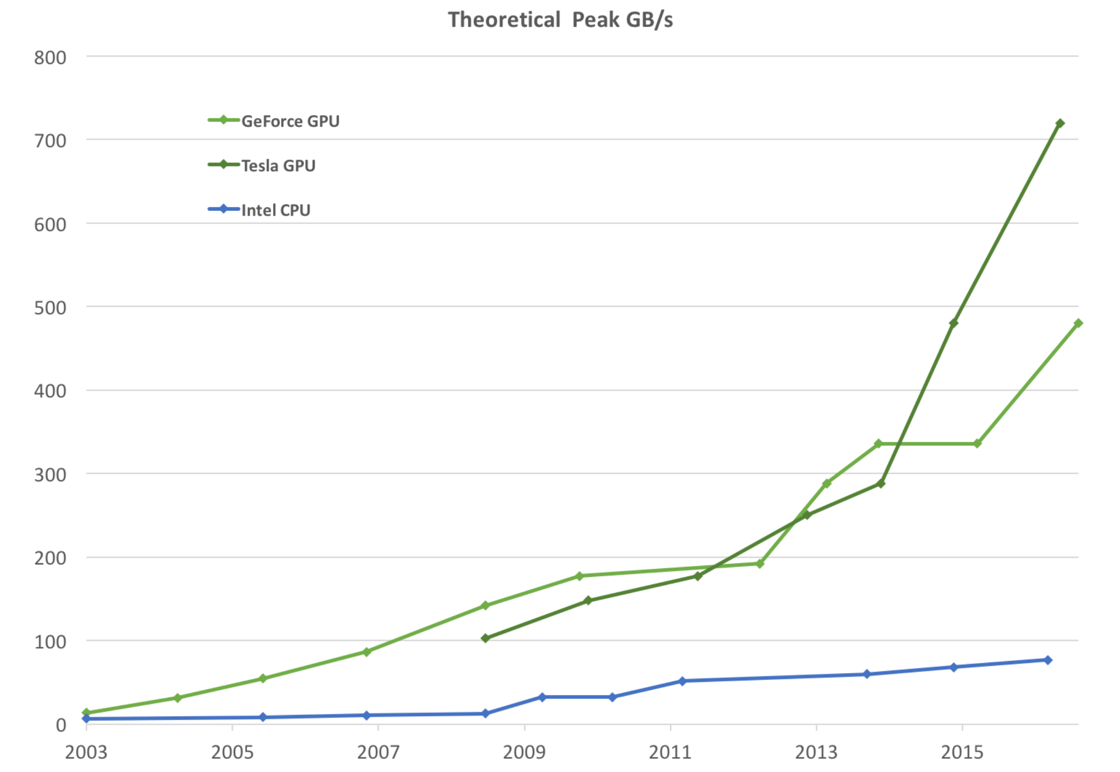


Figure 6.2.: Development of NVIDIA GPU memory bandwidth over the different architectures. Graphic taken from the NVIDIA Programming Guide [188].

instance, individual threads are called work-items and thread blocks are referred to as workgroups.

The architecture of a modern GPU consists of a scalable array of multithreaded streaming multiprocessors. To execute hundreds of threads concurrently, a Single-Instruction, Multiple-Thread (SIMT) architecture is used on each streaming multiprocessor. It combines instruction-level parallelism within each thread with thread-level parallelism through simultaneous *hardware multithreading*. Each streaming multiprocessor partitions its thread blocks into groups of 32 threads which are then scheduled to be executed in parallel within a *warp*. Each multiprocessor partitions a set of 32-bit registers among the warps as well as shared memory among the thread blocks. Apart from general limits on the number of thread blocks and warps on a streaming multiprocessor, memory heavy kernels can further reduce the maximum number of warps and/or thread blocks available for its execution. Within a warp, one common instruction is executed at a time. Thus, maximum efficiency is achieved if all threads within a warp execute the same command. If there is a conditional branch, such as an if-statement, threads that are not on the same path are disabled and executed later. To write efficient code, this path branching should be avoided if possible. For additional information on GPU architectures we defer to the official NVIDIA programming guide [188].

Due to the great performance and easy availability of GPUs, GPU computing is a very active field of research. Many researchers have developed GPU implementations of their numerical methods in an effort to create faster and more efficient codes. We will use this opportunity to present a small and incomplete overview of the work in the field of GPU acceleration of CFD codes.

Early attempts at GPU acceleration in scientific computing date back to before the adaptation of graphics hardware towards general purpose programming, which really first started with the CUDA compatible cards GTX 8800 or Tesla C870 in 2006. For particle based Lattice Boltzmann methods, first GPU accelerations were carried out as early as 2003, relying on OpenGL to treat graphics hardware as a general single instruction, multiple data (SIMD) computer [158]. First GPU acceleration for finite difference time domain methods were introduced in 2004, reporting performance increases by a factor of 7 when compared to a CPU of the same generation [139]. Finite element solvers were considered in this context in 2005 by Göddecke et al., although the achieved performance increases were only in the range of factors of 2-3 [86].

In [58, 49], the authors assess GPU performance for finite difference WENO methods applied to several hyperbolic systems including the Euler and Navier-Stokes equations in detail. Reported speed ups compared to single CPU performance range from a factor of 4 for low-order WENO schemes and coarse grids to up to 100 for high-order schemes on fine meshes. The speed ups they report depend strongly on the GPU used and are much larger for the GTX 480 then for the GTX 550. This is due to their varying performance of 1,345 GFLOPS for the GTX 480 compared to the 691.2 GFLOPS of the GTX 550. Similar comparisons for finite volume WENO type schemes applied to the Euler equations report speed up factors of 10-20 when compared to two different AMD and Intel CPUs [100]. In [5], the authors GPU-accelerate a finite difference WENO scheme with domain decomposition applied to the compressible Favre-averaged Navier-Stokes equations in an effort to enable large-scale DNS and LES computations. Further effort has been invested towards using GPU acceleration of large-scale simulation of turbulent flows in [216]. They performed a DNS of a spatially evolving compressible mixing layer. Comparing the performance on a NVIDIA Tesla S2070 with the AMD Opteron 2352 (MPI-parallelized for 32 cores), the authors find a performance increase by a factor of 22. Measuring power consumption, they report savings of 240 W vs 575 W. They do, however, note that the applicability of GPU acceleration is harshly limited, in their case, by the video memory of the GPU. This issue can be overcome by using multiple GPUs or increasing the video memory of a single GPU. As we have described above, the amount of video memory is still increasing with each new hardware generation, but the memory available today on one GPU is still insufficient for (very) large problems. We give an overview of problem sizes and their usage of video memory for the numerical scheme presented in this work in Section 6.3.1.

In this light, the extension of GPU-accelerated codes towards multiple GPUs, clusters of GPUs and hybrid CPU-GPU clusters is a logical progression. Fortunately, this is relatively straightforward via MPI parallelization, as done in [27, 238, 118, 261]. Another option is the integration of multiple GPUs into an OpenMP implementation, as the authors in [122] propose. For this work however, we restrict ourselves to MPI parallelization if multiple GPUs are used.

Thibault and Senocak presented one of the first multi-GPU incompressible Navier-Stokes solvers using a simple second order finite difference scheme and first order Euler in time in [238]. They report large speed up factors of 13 and 33 depending on the CPU for one GPU and up to 100 for multi-GPU performance when compared to a serial CPU execution, which, of course, is not quite a fair comparison.

In [96, 268], Zaspel and Griebel developed a multi-GPU implementation of a two-phase incompressible Navier-Stokes solver with Jacobi-preconditioned conjugate gradient solver for the pressure Poisson equation and a finite difference WENO level set method for all transport terms and the level-set gradient. They also offer an interesting study of performance per Dollar-investment and performance per Watt. Their results show a speed-up of the order of three by comparing equally priced GPUs and CPUs and a factor of two in energy efficiency in favor of GPUs.

In a series of papers [22, 24, 23, 215, 214], Brodtkorb, Saetra et al. develop a multi-GPU finite volume solver for the shallow water equations, the Kurganov-Petrova scheme. They use their scheme to simulate the 1959 Malpasset dam break case and report that the first 4000 seconds can be accurately approximated in 27 seconds. They also report that activating their real-time visualization merely increases the runtime by 11%.

GPU accelerated codes have also been proposed for a wide range of methods apart from the typical FD, FV and DG methods that we consider in this work. For example, GPU acceleration has also been applied and studied for the Lattice Boltzmann equations [261, 189], Monte-Carlo methods [17], preconditioned conjugate gradient solvers [4, 37] or smoothed particle hydrodynamics (SPH) methods [55].

Numerical schemes that lend themselves to a very high level of parallelization are particularly well suited for GPU architectures. Discontinuous Galerkin methods with explicit time integration are a prime example of numerical schemes that benefit immensely from a GPU implementation, as most of the computational work is element local and elements are only coupled through interface exchanges for the computation of the surface integrals. Thus, GPU acceleration for various types of DG methods are widely studied and applied with great success [72, 221, 222, 87, 76, 40, 41, 88, 90, 248]. One of the first works in the GPU acceleration of DG methods is by Klöckner, Warburton, et al. who implemented a nodal DG scheme on GPUs in 2009 [129]. They report a performance gain of 50 times the single CPU performance, depending on the polynomial order, and note, that the computation is memory bound in many cases and utilizes the full memory bandwidth during the simulation. This work has been continued by these authors in [131] with the application to the wave equation and focusing on overcoming software challenges to make the code more efficient. By their own words, the first paper is more technical, while the second is supposed to serve as a introductory piece towards a wider, non-technical audience [132]. With the third paper of the series, the authors focus their attention towards the use of a Python-based metaprogramming infrastructure, introducing the concept of run-time code generation [132]. The idea is, that coding is done in an abstract scripting language and then ported towards GPU code by the machine at run-time. This lead to the development of the abstract scripting languages PyCUDA and PyOpenCL [127, 126, 133, 128].

While the development of the PyCUDA and PyOpenCL frameworks started out for discontinuous Galerkin methods, they are easily applicable to other methods as well. In the open-source, Python based PyFR framework the authors provide a GPU accelerated implementation for solving advection-diffusion type problems on streaming architectures such as GPUs and general heterogeneous systems [259, 260]. The scheme is based on the flux reconstruction scheme by Huynh [115] and extensions by other authors such as Vincent, Castonguay, Jameson and others [245, 35, 252, 246]. One appeal of this scheme is, that under certain restrictions, one can recover various collocation based nodal methods such as DG, spectral volume, or spectral difference by choosing the appropriate correction function [244]. In a comparison study to industry standard CPU code STAR-CCM+, a GPU accelerated fluid solver based on PyFR shows advantageous results in terms of performance and achieved accuracy at a given cost [244].

In this work, we use a similar abstract framework for the GPU acceleration called OCCA. It is similar to PyCUDA and PyOpenCL but is based on C instead of Python and works with both, CUDA and OpenCL, in the same framework. We will present OCCA in more detail in Section 6.2.

6.2. OCCA

One problem in the development and implementation of programs is that they are typically tailored towards a specific hardware architecture and programming language. Executing the same code on a different platform may be challenging if not impossible. Also, it is not guaranteed that the underlying hardware architecture endures and will not be replaced by a different (superior) technology. An example of this is the cell architecture of IBM which was used for the Sony Playstation 3. While the cell architecture was ahead of its competition in terms of performance due to its parallel structure, it was deemed too complex and challenging to program for by the gaming industry. Thus, this technology was not utilized in the next generation's Playstation 4 console. This uncertainty raises the question of code longevity if it is developed for one specific hardware architecture.

In an effort to enable portable programming for a wide range of hardware architectures and programming languages, the unified framework implementation OCCA has been developed by Medina et al. [172, 170]. The framework introduces an intermediary programming language called the OCCA kernel language (okl) that is an extension to C. Kernels written in okl are then translated and compiled at run-time in a specified programming language on a specified hardware device. Supported programming languages include parallel options such as OpenMP, OpenCL and CUDA as well as a straightforward serial version. The supported hardware devices similarly range, among others, from simple one core CPUs in serial mode to multiple cores via OpenMP or OpenCL and massively parallel GPU architectures via OpenCL and CUDA.

Applications are set up to distinguish between a **host**, which is typically a single CPU core, and a **device**, which can, e.g., be a multi-core CPU or a GPU. The initial set up of the program and the device is performed on the host. Necessary data is allocated, initialized and transferred to the device. Then the device performs the desired, usually computationally expensive, operations and passes the result back to the host for further processing or output purposes. To increase the parallelism of this approach even more, MPI can be used to have multiple hosts each handle a device. This makes it straightforward to combine multiple GPUs for the computation of larger problems. The simplest way to do this is to have one host per device but it is possible to have one host handle multiple devices in the OCCA framework.

Many applications and numerical schemes have been implemented on GPUs using the OCCA framework. A performance comparison of high-order finite difference codes written in the abstract language OCCA versus the native implementation in OpenCL or CUDA show that the overhead of OCCA is minimal [170, 171]. Furthermore, OCCA implementations have been developed for DG methods, among others, for the shallow water equations [76, 254], the incompressible Navier-Stokes equations [121] and acoustic and elastic models [177]. Another application using CUDA is atmospheric modelling as discussed in [251]. Also, different types of DG methods have been investigated as to their applicability on GPUs via OCCA. For example, different types of meshes [39, 40] and basis functions [41] have been studied in terms of performance and applicability for GPU implementations. A discussion of high-order absorbing boundary conditions and corner/edge compatibility can be found in [175]. While even straightforward implementations on GPU architectures usually provide decent performance, only optimizing the kernels can unleash the full power of modern GPU computing capability. Optimization strategies for kernels using tensor-product operations can be found in [231]. Finally, the implementations on clusters of GPUs is discussed for the reversed-time migration problem in [176].

The easier implementation in combination with the increased portability across platforms and minimal computational overhead make implementing code in OCCA a very attractive option. For these reasons, we choose the OCCA framework as the basis for the ESDGSEM GPU implementation.

6.3. The ESDGSEM implementation

As typical for discontinuous Galerkin methods, the entropy stable version presented in Chapter 5 is highly parallelizable and, thus, a very good candidate for GPU acceleration. Most of the computational work of the scheme is performed locally on each element. Communication between elements happens only through the exchange of element interface information, which is used for the computation of the numerical fluxes. Additionally, global data comparison is needed to determine the stable explicit time step according to the CFL condition (5.106).

The OCCA implementation of the ESDGSEM distinguishes between a host on a single CPU core and the OCCA device, in this case a GPU, that is handled by the host. Open MPI is used to launch several host instances which are then able to each host a GPU device. The calculation is set up by a single rank zero host and distributed evenly among the hosts. Each host then initializes its device, allocates the memory necessary for the calculation and transfers the needed data onto the device. After this initial setup, data transfers between host and device only occur for MPI communication when using multiple devices (GPUs). As host to device data transfer is rather slow, this is an important aspect for the efficiency of the code. When using multiple devices, an exchange of boundary information on MPI-boundaries is necessary for each Runge-Kutta step of the calculation. Also, once for each time step, the maximum eigenvalue must be collected from each rank such that a global maximum eigenvalue can be determined. In practise, data transfer time can be hidden behind the computations by using non-blocking MPI-sends.

On each MPI rank, the local computations of the ESDGSEM are divided into multiple kernels. This matches the different kinds of parallelism that can be utilized between the different steps. Primarily, there is a distinction between kernels that run in parallel for each element interface and kernels that parallelize the elements. Kernels computing interface values in parallel include a kernel for the computation of the numerical interface fluxes and a kernel that gathers the *left* and *right* state data on each interface. Element based parallelization is used in the kernel for the computation of the volume integrals as well as in the kernel for the computation of the surface integrals. Also in this category are the kernels from the Runge-Kutta time integrator and the solution update kernel. Finally, the kernel computing the maximum eigenvalue and the shock capturing and positivity preservation kernels handle elements in parallel. The kernels from the artificial viscosity steps are structured in a very similar way, introducing new surface and volume kernels to the computation.

While the computation of the four surface integrals of each individual element only needs the interface data, special care must be taken when updating the time derivative.

This is due to the fact that the corner nodes are included as part of two interfaces. Thus, two threads will try to update that specific data point simultaneously, leading to a race condition and unpredictable behaviour. Even if the parallelization is based on elements, it must be ensured that this does not happen. In this implementation, only opposing interfaces are updated at the same time with a barrier separating the two calculations.

We also point out that this structure of kernels may leave some room for optimization. In particular, some kernels might be merged into one kernel, saving kernel launch times as well as precious data loads and stores. There is, however, a precious balance between acceptable occupancy and register pressure and good performance. Putting too much work into one kernel may result in a decline in performance and it is difficult to predict a priori how much is optimal.

None of the modifications of the split form based ESDGSEM change the strong parallelizability of standard discontinuous Galerkin methods. The ESDGSEM is, however, more computationally expensive as the split form volume integral requires additional floating point operations compared to a standard volume integral. To see this, we restate the algorithm for the volume integrals at a node i, j:

$$\begin{aligned}(\text{Split Form Volume})_{ij} &= \sum_{l=0}^{N} \tilde{D}_{il}\tilde{F}^{\#}_{(l,i),j} + \sum_{l=0}^{N} \tilde{D}_{jl}\tilde{G}^{\#}_{i,(j,l)},\\ (\text{Standard Volume})_{ij} &= \sum_{l=0}^{N} D_{il}\tilde{F}_{l,j} + \sum_{l=0}^{N} D_{jl}\tilde{G}_{i,l}.\end{aligned} \tag{6.1}$$

The computation of the fluxes $\tilde{F}^{\#}$ and $\tilde{G}^{\#}$ each consists of computing $N+1$ flux evaluations for each node i, j, totaling $2(N+1)^3$ flux evaluations. In the standard DG formulation the fluxes require only one evaluation at each node i, j, resulting in $2(N+1)^2$ flux evaluations. Additionally, each individual flux evaluation is more expensive for the ESDGSEM as it essentially consists of averaging two flux evaluations at i, j and l, j for $\tilde{F}^{\#}$ or i, j and i, l for $\tilde{G}^{\#}$.

One strategy to increase the performance of the split form volume kernel is to use the symmetry $\tilde{F}^{\#}_{(l,i),j} = \tilde{F}^{\#}_{(i,l),j}$ to drastically reduce the number of floating point operations. This effectively halves the number of operations at the cost of storing more data, yielding a cost factor of about 2.5 between standard and ESDGSEM. On GPUs, this trick is not possible due to the limited shared memory space. Precomputing and storing the fluxes $\tilde{F}^{\#}_{l,i,j}$ and $\tilde{G}^{\#}_{l,i,j}$ for $i, j, l = 0 \ldots, N$ would exceed the shared memory space even for medium sized problems. These architectural limitations raise the question of how to optimize GPU kernels and which end performance is actually satisfactory.

Our main test system features a NVIDIA GTX 1080 which is a high end Pascal generation consumer grade video card at the time this work is written. We compile the kernels in CUDA and run the code in single precision to investigate the effect of kernel modifications on the performance. As discussed in Section 6.1, the GTX 1080 lacks double precision processing power, with only about $\frac{1}{32}$ as many double precision CUDA cores compared to the single precision cores. Thus, we use the NVIDIA Tesla V100, to verify our findings in double precision. The Tesla V100 is a state of the art scientific

Elements	TotalElements	N	SolutionDofs	VRAMGB
2,000	$4 \cdot 10^6$	1	$1.2 \cdot 10^7$	7.6
1,500	$2.25 \cdot 10^6$	2	$2.7 \cdot 10^7$	7.63
1,200	$1.44 \cdot 10^6$	3	$3.89 \cdot 10^7$	7.63
1,000	$1 \cdot 10^6$	4	$4.8 \cdot 10^7$	7.62
850	$7.23 \cdot 10^5$	5	$5.42 \cdot 10^7$	7.49
750	$5.63 \cdot 10^5$	6	$6.08 \cdot 10^7$	7.61
670	$4.49 \cdot 10^5$	7	$6.6 \cdot 10^7$	7.68
600	$3.6 \cdot 10^5$	8	$6.91 \cdot 10^7$	7.6
550	$3.03 \cdot 10^5$	9	$7.35 \cdot 10^7$	7.72
500	$2.5 \cdot 10^5$	10	$7.5 \cdot 10^7$	7.59
460	$2.12 \cdot 10^5$	11	$7.68 \cdot 10^7$	7.53
430	$1.85 \cdot 10^5$	12	$7.99 \cdot 10^7$	7.63
400	$1.6 \cdot 10^5$	13	$8.11 \cdot 10^7$	7.57
375	$1.41 \cdot 10^5$	14	$8.27 \cdot 10^7$	7.57
350	$1.23 \cdot 10^5$	15	$8.27 \cdot 10^7$	7.44

Figure 6.3.: Various problem sizes for varying polynomial degree while keeping close to the GPU video memory capacity of 8 GB.

computing card, that features in many modern super computers. We show the double precision results in Subsection 6.3.3.

Our results show that the immense processing power of modern GPU hardware alleviates most of this increased computational complexity. We even observe that for polynomial degrees $N \leq 7$ the increased computational complexity of the split form is completely mitigated by the GPU processing power.

6.3.1. Memory capacity

A disadvantage of GPU computing is that the problem size is limited by the GPU memory. Modern consumer grade GPUs typically offer up to 12 GB of Video Random Access Memory (VRAM). The NVIDIA GTX 1080 used for the single precision computations in this work offers 8 GB, while the scientific computing card of choice, the NVIDIA Tesla V100, comes with either 16 GB or 32 GB of video memory. Comparing the new Volta generation cards to their Pascal predecessors highlights the trend that large memory capacities are in high demand and further increased memory sizes are to be expected in the future.

We give an overview on problem sizes and their memory usage on one NVIDIA GTX 1080 in Table 6.3. It becomes apparent that high-order schemes are very efficient in terms of memory usage, as more degrees of freedom can be used for higher polynomial degrees while keeping the memory requirements at a similar level.

6.3.2. Practical performance metrics

Performance of graphics cards are determined by several factors. Each GPU engine has a certain amount of CUDA cores with a certain clock frequency. Another crucial component determining the overall performance is the memory which, again, comes with a specific speed and bandwidth. For the NVIDIA GTX 1080 there are 2560 CUDA cores with a base clock of of 1607 MHz which, in boost mode, can go up to 1733 MHz. The 8GB of memory have a speed of 10 GBps and feature a 256-bit memory interface with a bandwidth of 320 GB/s. The theoretical peak performance numbers are given by 8873 GFLOPS for single precision operators and 277.3 for double precision operations.

While these are the performance numbers provided by the manufacturer, in practice these numbers are not achievable. In an effort to find a more realistic upper limit for the kernel performance, we estimate an empirical bound by investigating the effective bandwidth (BW) of a kernel by

$$\text{effective BW} = \frac{\text{Bytes Read} + \text{Bytes Written}}{t}. \tag{6.2}$$

A natural upper limit on the performance of any kernel is the time it takes to copy the same amount of memory that is read and written by the kernel. As a memory copy reads and writes each data, we measure the time it takes to copy a buffer of size $\frac{\text{Bytes Read} + \text{Bytes Written}}{2}$. This memory copy is performed by the *cudaMemcpyDevice-ToDevice* function provided by the hardware manufacturer. The resulting bandwidth is called the MemCopy bandwidth.

We compute an empirical upper limit on the kernel performance, the **MemCopy roofline**, by assuming that the achieved GFLOPS/s of the kernel were run at full MemCopy bandwidth. Thus, the MemCopy roofline is computed by scaling the GFLOPS/s achieved by the kernel with the factor of MemCopy-bandwidth over effective kernel bandwidth

$$\text{MemCopy roofline} = \frac{\text{GFLOPS/s} \times \text{MemCopy BW}}{\text{effective Kernel BW}}. \tag{6.3}$$

If the effective kernel bandwidth is optimal, the kernel performance graph lies exactly on the MemCopy roofline. In this case, the kernel is limited by the memory bandwidth or **memory-bound**. If a kernel performance is limited by the computations, the kernel bandwidth will deviate from the MemCopy-bandwidth. The resulting performance graph will be below the MemCopy roofline and the kernel is called **compute-bound**.

While the MemCopy roofline is a good upper bound on kernel performance whenever a kernel is memory-bound, the upper bound is not very strict for compute-bound kernels. Another approach to find a limiting factor for the achievable performance, is the shared memory bandwidth (SM BW) estimated by

$$\text{SM BW} = \#\text{cores} \times \#\text{SIMD Lanes} \times \|\text{word in bytes}\| \times \text{clock frequency}. \tag{6.4}$$

For the specifications of the GTX 1080 we find the shared memory bandwidth to be

$$\text{SM Bandwidth} = 20 \times 32 \times 4\text{bytes} \times 1.607\text{GHz} = 4113.92\text{GB/s}. \tag{6.5}$$

We scale the shared memory bandwidth by the performed work per memory access to find the shared memory roofline

$$\text{shared memory roofline} = \text{SM BW} \times \frac{\text{flops per block}}{\text{SM bytes loaded+stored per block}}. \tag{6.6}$$

Both, the MemCopy roofline and the shared memory roofline yield upper bounds on the kernel performance. We take the minimum of both rooflines to obtain the stricter bound everywhere,

$$\text{combined roofline} = \min\left(\text{SM roofline}, \text{MemCopy roofline}\right). \tag{6.7}$$

These performance bounds provide a sharper hardware limit on the performance than the manufacturer's performance numbers. Whenever the actual kernel performance is close to the roofline we can be satisfied with the kernel efficiency. In compute bound cases it is usually still impossible to reach the performance bound provided by the roofline presented here.

6.3.3. Optimization of the split form volume kernel

We aim for an efficient GPU acceleration of the volume kernel computing the split form volume integral as shown in (6.1). To get a grasp on the impact of different optimization techniques, we start with a fresh, naive implementation. Sequentially, we introduce optimization steps and document the impact on the kernel performance. The average kernel runtime is measured over 1000 kernel executions for a sample test case. The performance measure GFLOPS/s is obtained as the number of floating point operations performed during each second of runtime. The optimization steps in each kernel version are described later in this section. The optimization techniques are similar to previous works, e.g. in [76] and subject to current research, e.g. [41, 231, 121].

To investigate the kernel efficiency for different configurations, we follow two strategies. In one strategy, we aim for a similar amount of degrees of freedom. As observed in Section 6.3.1, this leads to a smaller memory footprint for higher orders. Thus, the second strategy aims to use a similar amount of GPU memory. In both cases, we increase the polynomial order N and decrease the number of elements K to increase the computational complexity of the volume integral. While the first strategy keeps the algebraic problem size the same and the memory problem size decreases, the second strategy increases the algebraic problem size to keep the memory problem size roughly constant, relatively close the GPU memory of 8 GB for the GTX 1080. The exact combinations of polynomial orders and number of elements can be found in Figure 6.6 for strategy 1 and in Figure 6.7 for strategy 2.

Optimization steps

- Version 1: Minimizing Global Loads / Utilizing Shared Memory

  The first step in improving the kernel performance is to reduce the number of costly loads from global GPU memory. To do this, we load all the necessary data only once. If the value is needed by several nodes of the element, we store it in shared memory, which is fast but limited. If it is only needed by an individual node, and thus only in one thread, we store it in a thread-local register. Also, data from shared memory that is used multiple times is loaded to register memory only once.

- Version 2: Declaring variables constant and pointers restricted

  As an additional step to improve data storage and transfer, we declare all variables that do not change their value during the computation as **constant**. We also set all pointers passed to the kernel as **restricted**. We store constant values such as $\frac{1}{2}g$ and $\frac{1}{4}g$ as kernel infos. We also introduce an additional shared memory array that stores the inverse of the water height $1/h$ for the flux calculations.

- Version 3: Multiple Elements per Block

  One GPU thread block is typically able to handle multiple DG elements in parallel, depending on the polynomial degree. To reduce the number of idle threads, we introduce a parameter N_E that sets the number of elements handled per thread block. This number depends on the polynomial order and typically decreases with N. Changes in this parameter can drastically impact the performance of the kernel, so this is a parameter open to optimization.

- Version 4: Optimizing the Loops

  To align memory access, adjacent threads should access adjacent global device memory. This is ensured by accessing index i, j as $i \times (N+1)+j$ if j is the innermost loop index. Also shared memory is accessed fastest, if the innermost loop is over the outermost index. So if the shared variable is accessed as $s\_Q[\text{ieLoc}][i][j]$ the inner loops should be over ieLoc, i, j in exactly this order. Also, loop unrolling is added to the serial inner loops to give the compiler room for further optimization.

- Version 5: Avoid Bank Conflicts, add Padding

  To avoid bank conflicts when accessing memory, we increase the size of the shared memory arrays by one, if $N+1$ is a multiple of 4. This is done by a variable $nglPad$ which is 1 or 0 depending on the polynomial degree and added to the last entry of the shared memory arrays.

- Version 6: Split inner loop & Hide shared memory loads

  We split the inner loop in two and compute the F and G fluxes and their contribution to the volume integral separately. This potentially opens up room for the compiler to optimize register loads and ease register pressure. We also change the order of operations such that variables needed for the update of the time integral such as J, b_x or b_y are loaded before the flux derivatives are computed. This could potentially hide load times behind the flux computations. We also introduce separate variables for the flux derivatives for F and G and then add them together in the end in the update step.

Finding the right number of elements per block

Kernel optimization step 3 has introduced the capability of assigning multiple DG elements to one thread block. Each thread block is able to handle 512 concurrent threads, however it is not always optimal to use all of these. Due to limited register pressure per work group, computation heavy kernels may be faster if there is room left for the compiler to optimize the kernel. We introduce a parameter N_E to denote the number of DG elements assigned to each work group and optimize it by timing the volume kernel for a sample test case. We want to note that in the full code there are several such parameters. For instance, the surface integral has its own parameter of how many interfaces are handled per work group. We show the runtimes of the volume kernel of both the ESDGSEM as well as the standard DG volume kernel for varying numbers of elements per thread block and different polynomial degrees in Tables 6.1, 6.2 and 6.3. In many cases there are several sweet spots for the N_E parameter. In the $N = 5$ case, these values are multiples of 4, e.g. $N_E = 4$, $N_E = 8$ and $N_E = 12$. One case where too many elements in one thread block leads to suboptimal performance can be seen, e.g., for $N = 7$ in Table 6.2, where performance levels drop for $N_E > 6$. In the $N = 7$ case, any parameter choice $N_E < 6$ leads to similar run time results. The higher the polynomial degree, the less options for choosing the N_E parameter are available. For $N = 12$, the best choice is to use three elements per thread block. Choosing more elements is impossible since four elements per thread block would result in $12^2 \times 4 = 576$ threads which is more than the 512 the GPU can handle. For the best performance results an optimal parameter N_E must be found for each polynomial degree, as we have done for the results shown in the following sections.

Single precision results

We show the achieved performance of each kernel version against the empirical roofline (6.7) derived in Section 6.3.2. This highlights the impact of the different optimization techniques applied to both, the standard DG volume kernel and the split form volume kernel from (6.1). We, again, distinguish between the constant memory load and the constant degrees of freedom strategies described above. The results for the different kernel versions of the standard DGSEM volume kernel can be found in Figure 6.4, whereas the results for the split form ESDGSEM volume kernel are shown in Figure 6.5. The performance results show that both strategies lead to similar results. This should be expected unless the different problem sizes lead the kernels to be compute bound for one configuration but not the other. The notable exception is the performance of kernel version V3 for the standard DGSEM, which shows a significant performance drop in case the degrees of freedom are kept constant.

We also compare the final kernel versions for each volume integral and show operation counts and runtimes for $N = 1, \ldots, 15$ with similar degrees of freedom in Figure 6.6 and for similar memory loads in Figure 6.7. For polynomial degrees $N \leq 7$ the optimized ESDGSEM volume kernel is memory bound and shows optimal performance. Increasing the polynomial order further leads to stagnating performance as the kernel becomes

N_E	ESDGSEM	DGSEM
1	8.46	8.06
2	7.97	7.99
3	7.84	7.92
4	7.45	7.44
5	7.8	7.84
6	7.83	7.84
7	7.79	7.84
8	7.42	7.44
9	7.88	7.81
10	7.81	7.81
11	8.06	7.81
12	7.49	7.43
13	7.86	7.81
14	7.79	7.8

Table 6.1.: Split form and standard DG volume kernel runtimes for varying number of elements per thread block N_E at polynomial order of $N = 5$.

N_E	ESDGSEM	DGSEM
1	7.19	7.2
2	7.18	7.19
3	7.19	7.18
4	7.18	7.18
5	7.2	7.17
6	7.2	7.18
7	7.33	7.18
8	7.23	7.18

Table 6.2.: Split form and standard DG volume kernel runtimes for varying number of elements per thread block N_E at polynomial order of $N = 7$.

N_E	ESDGSEM	DGSEM
1	10.58	8.46
2	9.76	7.79
3	9.63	7.8

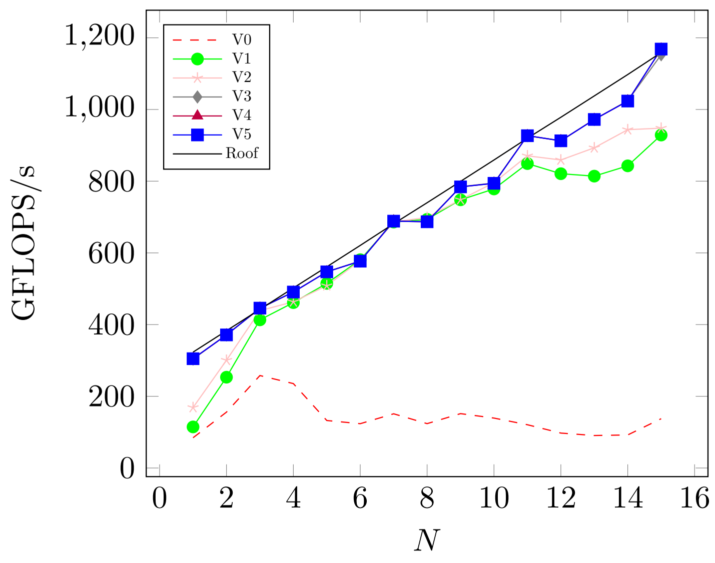
Table 6.3.: Split form and standard DG volume kernel runtimes for varying number of elements per thread block N_E at polynomial order of $N = 12$.

compute bound. In contrast, the standard DGSEM volume kernel is memory bound even for $N = 15$ and the achieved performances are close to the roofline for all polynomial orders. As split form and standard kernels require the same amount of data reads and writes, the runtime is basically identical in the memory bound region. For $N > 7$, the split form volume kernel is compute bound and the runtimes deviate. The diverging number of floating point operations becomes clearly visible in the required runtime for the execution of the kernels. However, even though there is a difference by a factor of 6 in terms of floating point operations for the ESDGSEM compared to standard DG, we only observe a runtime difference of a factor of 1.5 in the most computationally expensive $N = 15$ case. Thus, the additional runtime requirement is not as harsh as it may first seem.

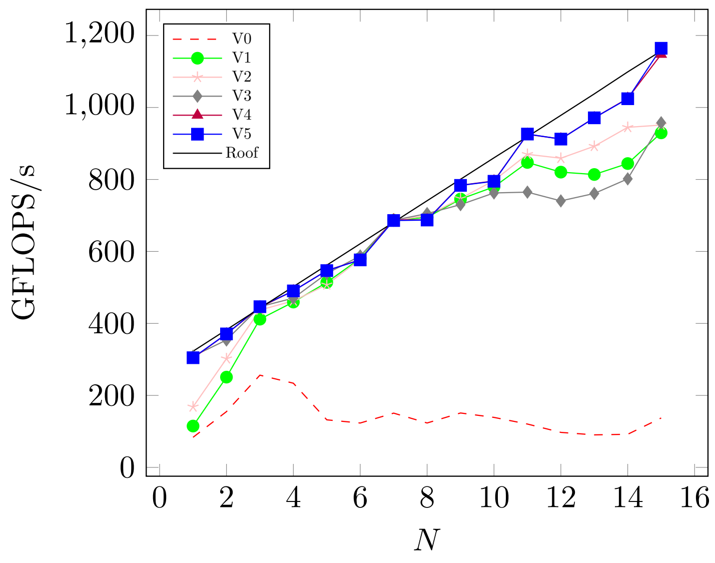
Looking at the raw numbers of achieved performances in terms of GFLOPS/s, we see that the split form kernel outperforms the standard volume kernel by factor of 3 to 5.5. This suggests that the ESDGSEM volume kernel profits immensely from the computational capabilities of the GPU and is indeed a very good fit to that architecture. For the very relevant cases of moderate polynomial orders $N \leq 7$, the additional computational complexity is completely mitigated by GPU processing power and there is virtually no cost attached to all the benefits of the entropy stable DG variant.

Double precision results

We use a different test system with a Tesla V100 to run the ESDGSEM volume kernel in double precision, using the same kernel versions as in the single precision case described in Section 6.3.3. The performance of the different kernel versions is shown in Figure 6.8 for the standard volume integral kernel as well as for the entropy stable split form kernel. Additional comparisons between standard and entropy stable volume kernels are provided for the most optimized kernels in Figure 6.9. The elements per work block were found by optimization for the GTX 1080 so there might be some room for improvement for the Tesla V100. This could explain why the most optimized kernel is already compute bound for $N = 7$ and why there is a significant drop in performance for $N = 9$. For lower order $N \leq 7$, the split form kernel performance is close to optimal.

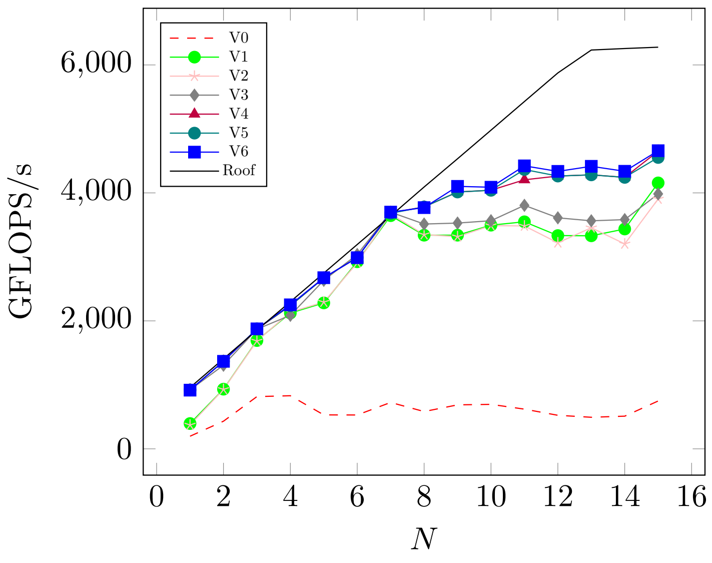


(a) Similar memory loads.

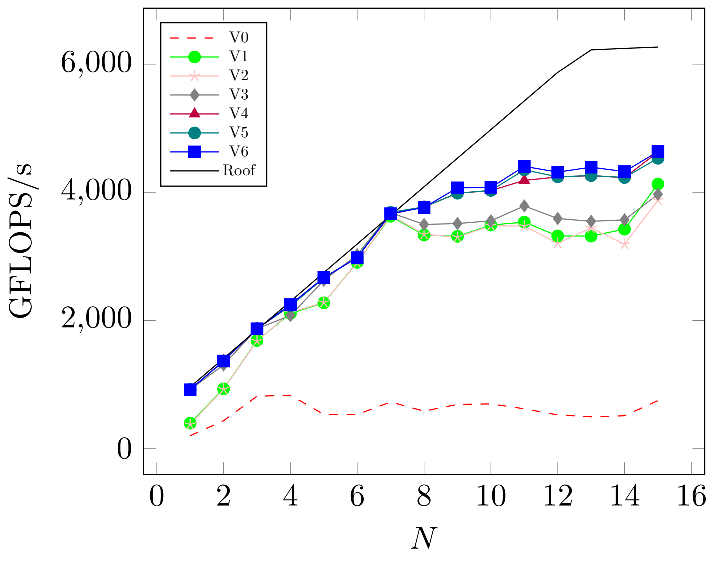


(b) Same number of degrees of freedom.

Figure 6.4.: Comparison of all standard DGSEM volume kernel versions and the memory roofline on a NVIDIA GTX 1080 in single precision for similar memory loads and same number of degrees of freedom. A remarkable difference can be observed for V3, which performs significantly better for the similar memory loads case. These results were previously published by Wintermeyer et al. in [254].



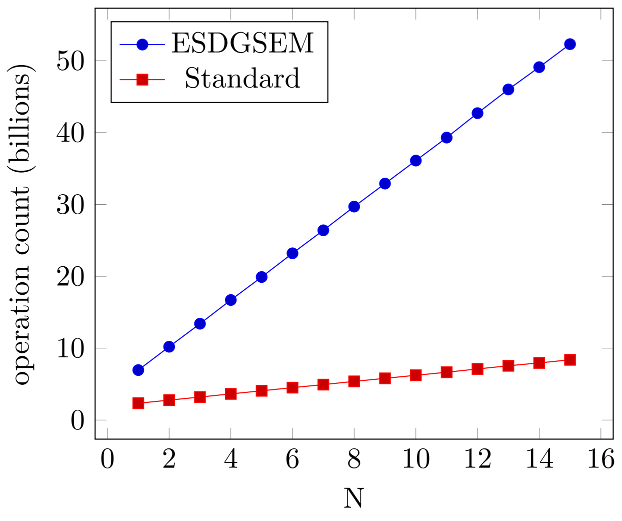
(a) Similar memory loads.



(b) Same number of degrees of freedom.

Figure 6.5.: Comparison of all ESDGSEM volume kernel versions and the memory roofline on a NVIDIA GTX 1080 in single precision for similar memory loads and same number of degrees of freedom. The kernels show similar performance levels regardless of the type of test. These results were previously published by Wintermeyer et al. in [254].

N	K
1	3,000
2	2,000
3	1,500
4	1,200
5	1,000
6	858
7	750
8	667
9	600
10	546
11	500
12	462
13	429
14	400
15	375



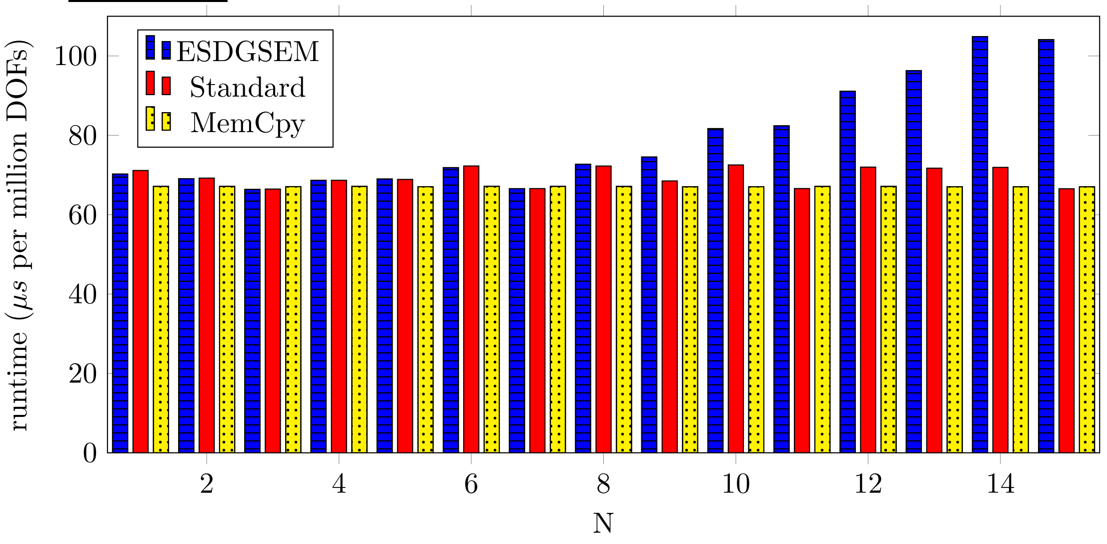
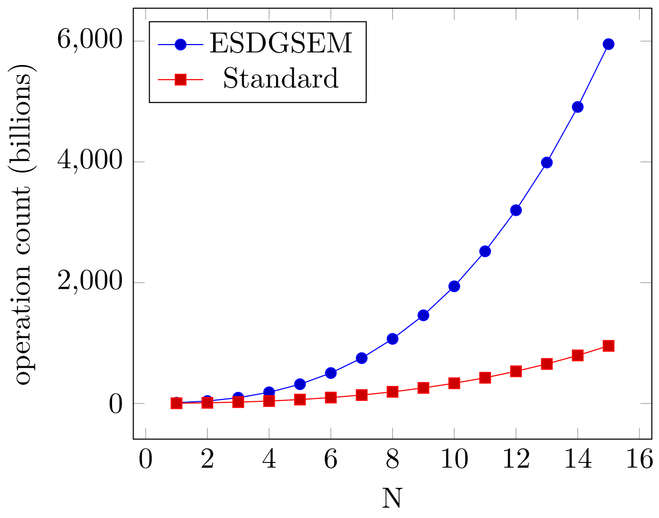


Figure 6.6.: Number of operations and runtime comparison for one kernel execution between the split form volume integral computation of the ESDGSEM and the volume integral of the standard DG method for similar number of degrees of freedom on a NVIDIA GTX 1080 in single precisions. Polynomial order N and number of elements K per spatial direction are listed in the top left table. This result was previously published by Wintermeyer et al. in [254].

N	K
1	4,000
2	2,800
3	2,000
4	1,600
5	1,400
6	1,200
7	1,000
8	900
9	800
10	750
11	700
12	650
13	600
14	550
15	500



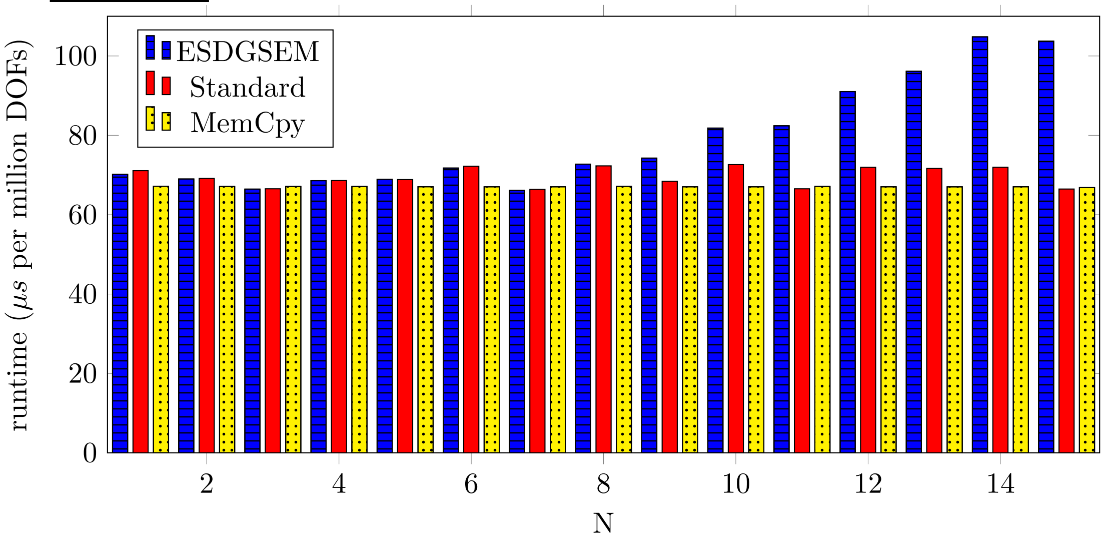
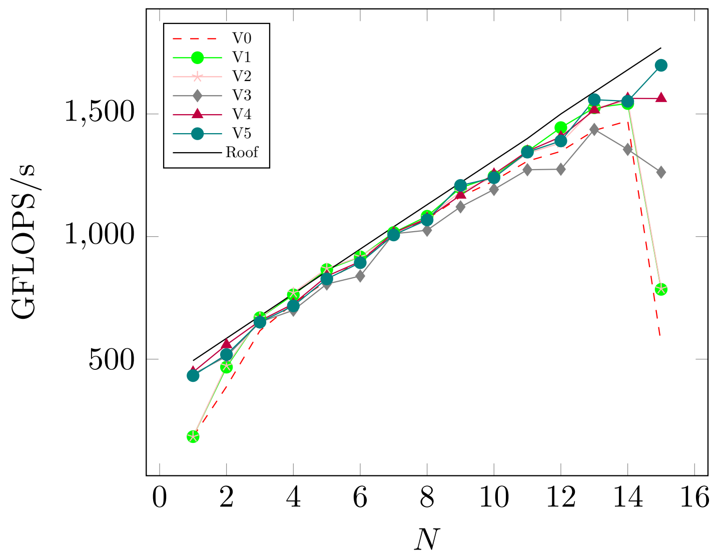
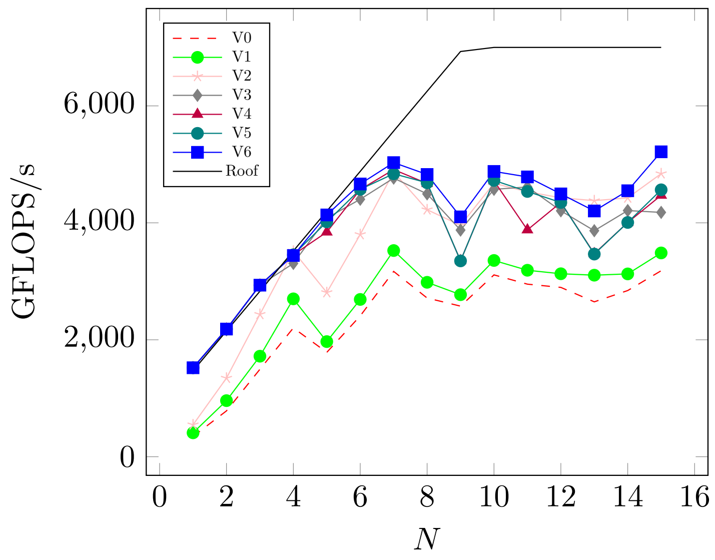


Figure 6.7.: Number of operations and runtime comparison for one kernel execution between the split form volume integral computation of the ESDGSEM and the volume integral of the standard DG method for increasing computational complexity on a NVIDIA GTX 1080 in single precision. Polynomial order N and number of elements K per spatial direction are listed in the top left table. This result was previously published by Wintermeyer et al. in [254].



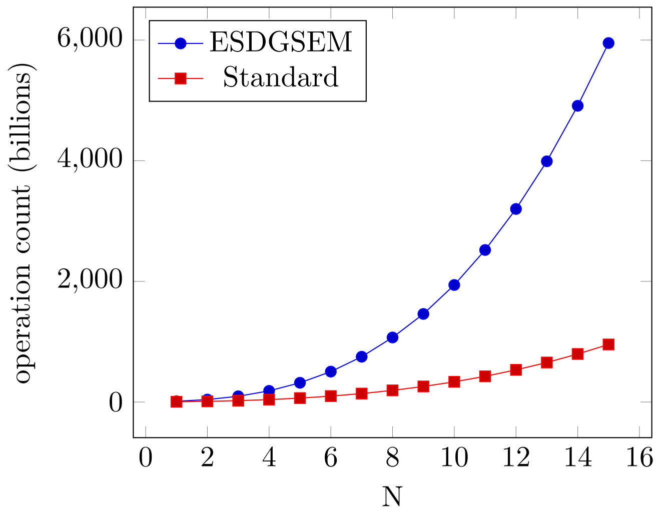
(a) Standard DGSEM volume kernels.



(b) ESDGSEM volume kernels.

Figure 6.8.: Comparison of all standard DG and ESDGSEM volume kernel versions and the memory roofline on a Tesla V100 in double precision for the similar memory load case. These results were previously published by Wintermeyer et al. in [254].

N	K
1	4,000
2	2,800
3	2,000
4	1,600
5	1,400
6	1,200
7	1,000
8	900
9	800
10	750
11	700
12	650
13	600
14	550
15	500



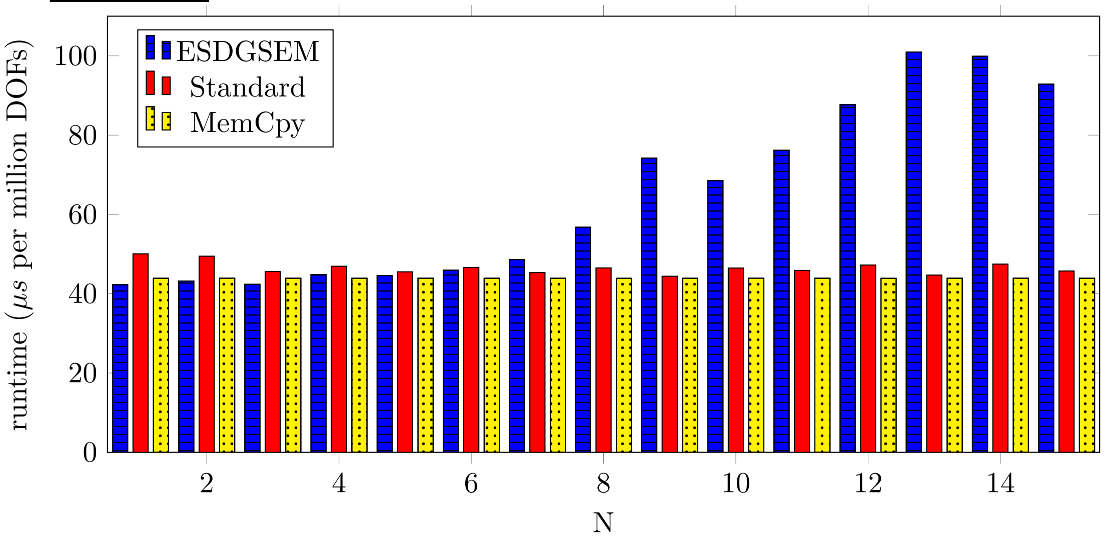


Figure 6.9.: Number of operations and runtime comparison for one kernel execution between the split form volume integral computation of the ESDGSEM and the volume integral of the standard DG method for increasing computational complexity in double precision on the NVIDIA Tesla V100. Polynomial order N and number of elements K per spatial direction are listed in the top left table. This result was previously published by Wintermeyer et al. in [254].

6.3.4. Kernel overview

To get a better grasp on the runtimes of the various kernels of the ESDGSEM implementation, we use the NVIDIA profiler nvprof. With nvprof it is possible to measure the runtimes of all the kernels. We use a dam break problem on a 200×200 element mesh for the polynomial degrees $N = 5, 10, 15$ and list the runtimes in Table 6.4. We note that while the volume kernel is the most thoroughly optimized, similar steps as listed in Section 6.3.3 have been performed for all the kernels. Without any optimization at all, the kernel runtimes are completely dominated by the amount of reads and writes to and from global GPU memory. Positivity preservation is turned off here as a special optimization is required for an efficient implementation, especially regarding the evaluation of minima, maxima and averages of element-wise quantities. The artificial viscosity computed according to BR1 is essentially another DG scheme for the gradient. Thus, the same concepts and optimization techniques can be applied for the viscous volume and surface kernels that compute the gradient and the artificial viscosity.

Kernel	N=5	N=10	N=15	Average
VolumeKernelSplitForm	8.99%	11.33%	12.92%	11.08%
SurfaceKernelGradient	11.54%	10.37%	10.57%	10.83%
UpdateKernel	8.44%	11.73%	12.02%	10.73%
VolumeKernelViscous	9.56%	11.13%	10.37%	10.35%
calcGradient	8.50%	9.64%	9.00%	9.05%
SurfaceKernel	8.57%	7.98%	8.23%	8.26%
scaleGradient	6.41%	7.25%	6.78%	6.81%
ShockCapturing	3.68%	5.83%	10.22%	6.58%
CollectEdgeDataGradient	8.31%	5.17%	4.63%	6.04%
CollectEdgeData	6.75%	5.57%	3.71%	5.34%
SurfaceKernelViscous	5.59%	5.07%	5.11%	5.26%
calcNumFluxes	3.60%	2.10%	1.34%	2.35%
calcNumFluxesViscous	3.54%	2.16%	1.31%	2.34%
calcNumFluxesGradient	3.44%	1.99%	1.26%	2.23%

Table 6.4.: Runtimes in relation to the total code runtime for a dam break problem on a 200×200 element mesh for polynomial degrees $N = 5, 10, 15$. This result was previously published by Wintermeyer et al. in [254].

7. Numerical Tests

In this chapter we thoroughly validate the proclaimed theoretical properties of the entropy conservative and entropy stable discontinuous Galerkin methods presented in chapter 5. The numerical tests cover the high-order convergence, the conservation with respect to conservative variables and entropy as well as the well-balanced property. Additionally, we subject the scheme to a series of challenging test cases to study the robustness and efficiency of the shock capturing capabilities and the positivity limiter.

We start by numerically verifying the theoretical properties in Section 7.1. Specifically, we investigate the high-order convergence as well as the conservation of water height and momentum (if the source term is zero). Furthermore, we show that the scheme is well-balanced for a discontinuous bottom topography. Finally, we conclude the numerical verification by showing that the scheme is entropy conservative or entropy stable depending on the numerical flux.

In the following section, we present a test case from [163] that demonstrates that non-entropy stable schemes such as the standard DGSEM can violate the entropy inequality and produce incorrect solutions for certain configurations. This *entropy glitch* further motivates the use of entropy stable discretizations.

To further explore the solution quality and robustness of the ESDGSEM with and without artificial viscosity, we show results for several well-known test cases. Some of these are quite challenging and require shock capturing and/or the positivity limiter to succesfully finish the simulations. The test cases considered in this work are the oscillating lake [263, 74, 36, 167, 242], the three mound dam break on a closed channel [74, 36, 167, 25, 264] and the solitary wave run-up [19, 167, 205]. As all these tests are designed on Cartesian meshes, we propose a parabolic dam break problem to test the ESDGSEM on a curved mesh as well. We set up the geometry such that the mesh is aligned with the curved dam. Then, we let the dam break partially in the center at $t = 0$. To make the test even more challenging, we also run this test with a dry area on the downstream side of the dam.

The time step in the computations is based on the CFL condition as in (5.106). If artificial viscosity is turned on, the time step is computed according to (5.127). The additional time step restrictions of the positivity limiter (5.139) are sufficient but not necessary. Thus, we do not enforce them rigorously but rather adjust the time step if needed. A more detailed discussion on choosing an appropriate time step for positivity preserving schemes is given by Xing et al. [264].

All the examples have been computed on one or two NVIDIA GTX 1080 GPUs. If multiple GPUs are used, they are parallelized by MPI such that each MPI rank hosts one GPU via OCCA [172, 170].

7.1. Numerical validation

We now validate the properties from Theorem 3 and Theorem 4 of the entropy conservative (5.76) and entropy stable (5.98) schemes numerically. To verify the convergence, conservation and the well-balanced property of the approximations we use a structured curvilinear mesh depicted in Figure 7.1. As the theoretical findings are for the semi-

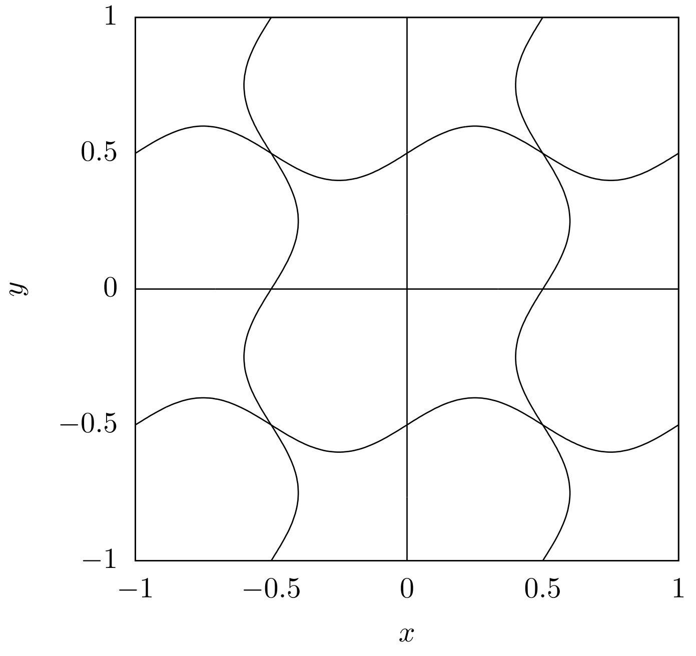


Figure 7.1.: The curvilinear mesh used for verification of convergence, conservation, and the well-balanced property.

discrete scheme, we use a fourth order low storage Runge-Kutta time integrator [31] for these tests, to reduce the impact of the time integration.

7.1.1. Convergence

We first verify the high order convergence of the ECDGSEM and ESDGSEM. We use the method of manufactured solutions to create an analytic solution

$$\begin{aligned} H(x,y,t) &= h(x,y,t) + b(x,y) = 8 + \cos(x)\sin(y)\cos(t), \\ u(x,y,t) &= 0.5, \\ v(x,y,t) &= 1.5. \end{aligned} \tag{7.1}$$

The additional source terms from the manufactured solution (7.1) are given by

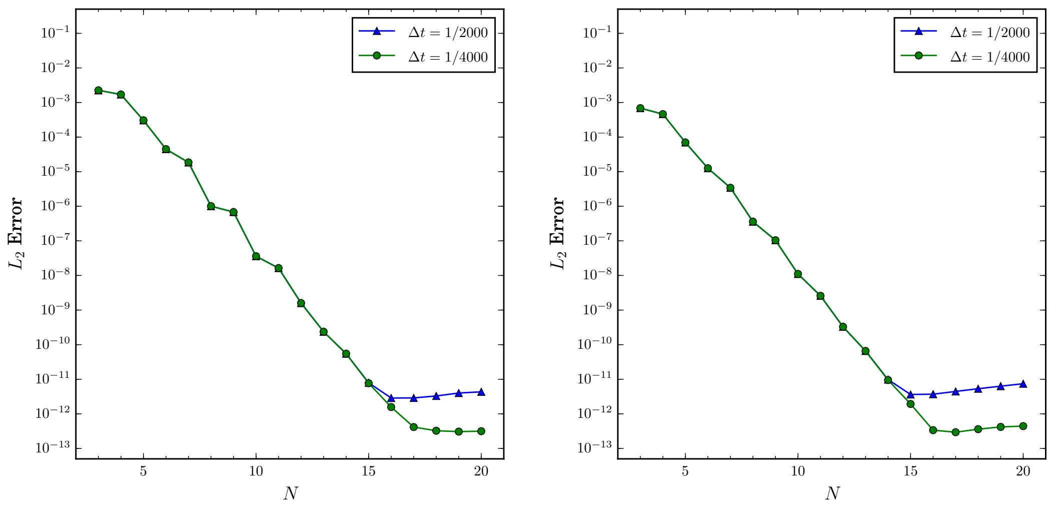
$$\begin{aligned} m_1 &:= H_t + u(H_x - b_x) + v(H_y - b_y), \\ m_2 &:= uH_t + u^2(H_x - b_x) + uv(H_y - b_y) + H_x(H - b), \\ m_3 &:= vH_t + uv(H_x - b_x) + v^2(H_y - b_y) + H_y(H - b). \end{aligned} \tag{7.2}$$

We solve this problem on the domain $[-1, 1]^2$ with the smooth bottom topography

$$b(x, y) = 2 + 0.5 \sin(2\pi x) + 0.5 \cos(2\pi y)\,. \tag{7.3}$$

The gravitational constant is set to $g = 1$. The derivatives of h and b are computed analytically from the given solutions (7.1) and (7.3).

Varying the polynomial degree on the mesh given in Fig. 7.1, we observe exponential convergence up to $N = 16$ ($N = 15$ for ESDGSEM) for $\Delta t = 1/2000$ and $N = 17$ ($N = 16$ for ESDGSEM) for $\Delta t = 1/4000$. At this point, the errors introduced by the time integrator become dominant and the accuracy stagnates. We present semi-log plots in Fig. 7.2. The order of converge is sub-optimal for odd polynomial degrees for the entropy conserving scheme, a phenomenon previously observed in the literature, e.g. [61, 78, 80, 83]. Both schemes demonstrate spectral accuracy for this smooth test problem, however.



(a) ECDGSEM.

(b) ESDGSEM.

Figure 7.2.: Semi-log plot shows the spectral convergence in space and fourth order accuracy in time for the ECDGSEM and ESDGSEM schemes applied to a smooth solution in Sec. 7.1.1. This result was previously published by Wintermeyer et al. in [253].

7.1.2. Conservation of mass, momentum and entropy

We now verify that the scheme is conservative for the water height and for the momentum, if the source term is zero. We also show that the entropy is a conserved quantity as well if the non-dissipative numerical interface fluxes (5.49) are used. Since entropy conservation is a semi-discrete property here, we expect to observe dissipative effects due to the time integrator. This dissipative effect is of the order of the time integration and strongly dependent on the time step size. For general non-constant bottom topographies the momentum equations become balance laws. In this case, we show that mass and entropy are still conserved discretely, even for discontinuous bottom topographies that align with element boundaries.

We demonstrate the described properties on a dam break problem on the domain $\Omega = [-1, 1]^2$. The dam break is initialised along the vertical line $x = 0$ on the curvilinear mesh shown in Fig. 7.1 with periodic boundary conditions and a polynomial degree of $N = 5$. The gravitational constant is set to $g = 1$. The dam break problem uses the initial conditions

$$h(x,y,0) = \begin{cases} 5 - b(x,y) & \text{if } x < 0 \\ 4 - b(x,y) & \text{if } x > 0 \end{cases}, \quad u(x,y,0) = v(x,y,0) = 0, \tag{7.4}$$

where b is defined in the following subsections.

Dam break over a flat bottom

We first consider the dam break (7.4) with a flat bottom topography, $b \equiv 0$, to verify conservation for water height, momentum and total energy. The changes in these quantities after a simulation up to $T = 1$ are listed in Table 7.1. The error in the discrete energy is proportional to the chosen time step and reflects the dissipative influence of the time integrator. The differences in water height and momentum are on the order of machine precision for all time step values. Shrinking the time step decreases the difference in the discrete total energy, converging at the fourth order accuracy of the time integrator.

Δt	Δh	Δhu	Δhv	ΔEnergy	Temporal Order
1/1000	3.55E-14	2.66E-13	4.32E-17	4.79E-08	–
1/2000	2.49E-14	2.66E-13	9.95E-16	3.01E-09	3.99
1/4000	3.20E-14	2.66E-13	1.71E-15	1.89E-10	3.99
1/8000	3.20E-14	2.66E-13	1.46E-15	1.18E-11	4.00

Table 7.1.: Differences in water height, momentum, and total energy accumulated over the entire domain at $T = 1$ compared to the initial condition for the ECDGSEM approximation for a constant bottom topography. In the entropy conservation results we observe the temporal accuracy of the time integrator. This result was previously published by Wintermeyer et al. in [253].

Dam break over a discontinuous non-flat bottom

Now, we examine the scheme for a discontinuous non-constant bottom topography defined by

$$b(x,y) = \begin{cases} 2 + 0.5\sin(2\pi x) + 0.5\cos(2\pi y)\,, & \text{for element } 2\times 2 \\ 0, \quad \text{otherwise} \end{cases}, \tag{7.5}$$

which is only non-zero in a single element. The momentum is no longer a physically conserved quantity, but the water height and total energy should still exhibit the same behaviour as for the flat bottom in the previous example. We show in Table 7.2 that the error in the total energy shrinks with the fourth order accuracy of the time integrator as the time step is refined. Also, we see that h is conserved to machine precision for all time steps.

Δt	Δh	ΔEnergy	Temporal Order
1/1000	5.33E-14	2.16E-08	–
1/2000	1.78E-14	1.35E-09	4.00
1/4000	2.84E-14	8.48E-11	3.99
1/8000	3.55E-15	5.32E-12	3.99

Table 7.2.: Differences in water height and total energy accumulated over the entire domain at $T = 1$ compared to the initial condition for the ECDGSEM approximation for the discontinuous bottom topography (7.5). We observe the temporal accuracy of the time integrator in the changes in total energy. This result was previously published by Wintermeyer et al. in [253].

7.1.3. Well-balanced property over a discontinuous bottom

We use a similar set up to show that both, the ECDGSEM and the ESDGSEM, are well-balanced on curvilinear meshes and for discontinuous bottom topographies. We configure a "lake at rest" test problem on the mesh from Figure 7.1 as in (2.37),

$$\begin{aligned} h + b(x,y) &= 5, \\ u = v &= 0, \end{aligned} \tag{7.6}$$

with the discontinuous bottom topography given in (7.5). The boundary conditions are set to be periodic. The time step is fixed at $\Delta t = 1/1000$. Table 7.3 shows that the L_2-error of the approximate total water height, $H = h + b$, is of the magnitude of round-off errors for both the ECDGSEM and the ESDGSEM for polynomial degrees $3, 4, 5$.

N	L_2-error ECDGSEM	L_2-error ESDGSEM
3	8.84E-15	5.37E-15
4	8.75E-15	5.02E-15
5	1.85E-14	1.55E-14

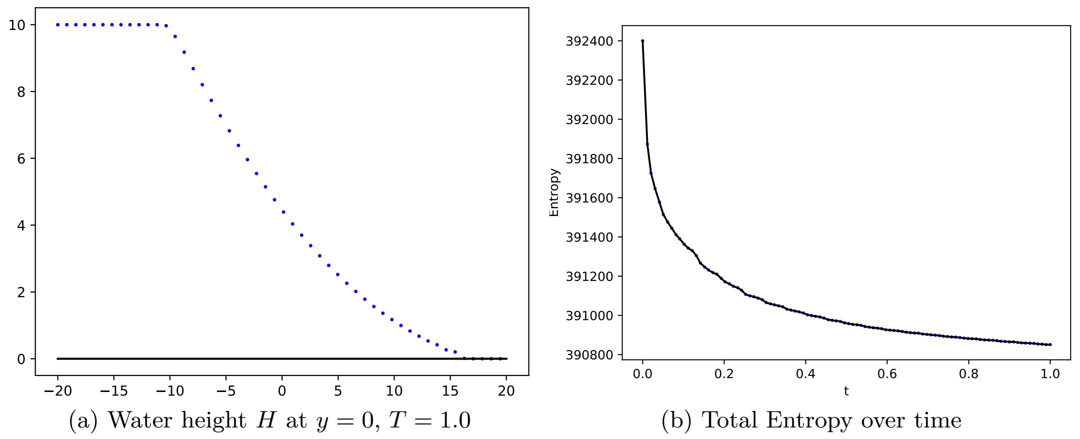
Table 7.3.: L_2-error of the approximate total water height, $H = h + b$, to the "lake at rest" problem on the curvilinear mesh shown in Fig. 7.1 at $T = 1$. This result was previously published by Wintermeyer et al. in [253].

7.1.4. Conservation and entropy stability of the positivity limiter

We also verify that the positivity limiter and the artificial viscosity maintain mass conservation and entropy stability of the ESDGSEM. To guarantee that the positivity limiter is activated, we use a dam break problem with periodic boundaries at $y = \pm 20$ and solid walls at $x = \pm 20$ on the domain $\Omega = [-20, 20]^2$. The initial conditions are given by

$$\begin{aligned} h(x,y,0) &= \begin{cases} 10.0, & \text{if } x < 0 \\ 0.0, & \text{otherwise} \end{cases}, \\ u = v &= 0. \end{aligned} \tag{7.7}$$

We run the test at a polynomial order of $N = 3$ on a uniform Cartesian mesh with 50×50 elements. The viscosity coefficient is set to be $\epsilon_0 = 0.1$ and we use the usual gravitational constant of $g = 9.81$. We observe a monotonically decreasing total entropy in Figure 7.3. Also, the mass is conserved up to machine precision. This test cannot be run without the positivity limiter, as it crashes immediately.



(a) Water height H at $y = 0$, $T = 1.0$ (b) Total Entropy over time

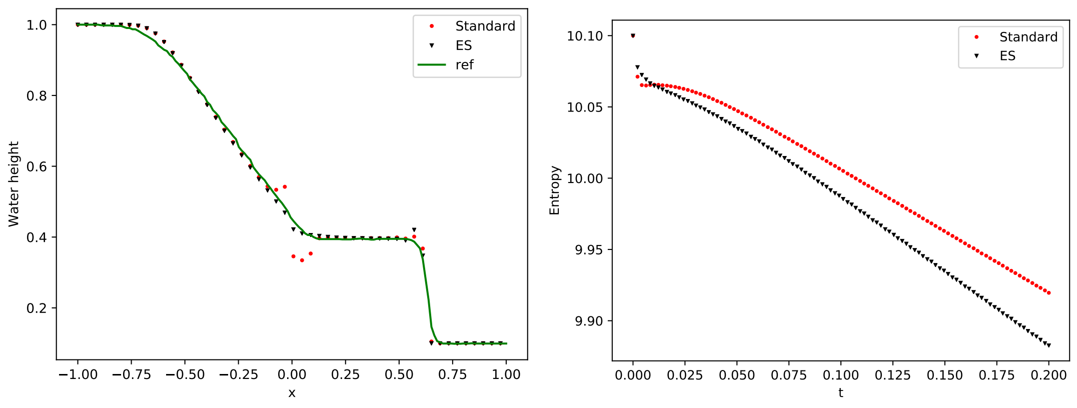
Figure 7.3.: Slice of total water height at final time and entropy evolution over time for a dam break problem with a dry zone approximated with the ESDGSEM with artificial viscosity and positivity limiter at polynomial order $N = 3$. This result was previously published by Wintermeyer et al. in [254].

7.2. Entropy glitch

We demonstrate the necessity of entropy stability to guarantee correct solutions on a specific dam break problem. We set up a two-dimensional version of the one-dimensional test case from [163] on a uniform Cartesian mesh, discretizing the domain $\Omega = [-1, 1]^2$ with 100×100 elements. The initial conditions contain a shock at $x = 0$ with water heights $h_L = 1$ and $h_R = 0.1$ and zero velocities, i.e.,

$$\begin{aligned} h(x, y, 0) &= \begin{cases} 1.0, & \text{if } x < 0 \\ 0.1, & \text{otherwise} \end{cases}, \\ u &= v = 0. \end{aligned} \tag{7.8}$$

For simplicity, the gravity constant is set to $g = 10$. We show results at $T = 0.2$ for $N = 1$ as the entropy glitch even occurs for this most robust case. The standard DGSEM quickly becomes unstable for higher polynomial orders due to the oscillations caused by the entropy glitch. The positivity limiter is enabled to catch minor overshoots from the oscillations. We compare slices of the solution at $y = 0$ with the solution provided in [163] for the one dimensional case. The standard DGSEM solutions develop an unphysical discontinuity, an “entropy glitch,” at $x = 0$ as shown in Figure 7.4. This glitch does not appear in the entropy stable solutions of the ESDGSEM. Small oscillations are visible at the shock front for both schemes. Plotting the evolution of total entropy for the both schemes, we observe a build up for the standard DGSEM at around $t = 0.02$. This unphysical build up does not occur for the ESDGSEM, which exhibits a strictly decreasing total entropy. We conclude that the entropy build up is the cause of the entropy glitch and, thus, the reason for the unphysical phenomena in the solution.



(a) Water height H, $N = 1$ (b) Total Entropy, $N = 1$

Figure 7.4.: Entropy glitch test case for ESDGSEM and standard DGSEM for $N = 1$ at $T = 0.2$ sliced at $y = 0$ compared to a 1D reference solution from [163]. The standard DGSEM introduces an incorrect shock caused by unphysical entropy production. This result was previously published by Wintermeyer et al. in [254].

7.3. Oscillating lake

A very useful test case for verifying the correctness of positivity preserving schemes is the oscillating lake. It features a parabolic bowl partly covered with moving water. Despite the presence of large dry areas, it allows for an analytic solution. The solution is initialized on the domain $\Omega = [-2, 2]^2$ by

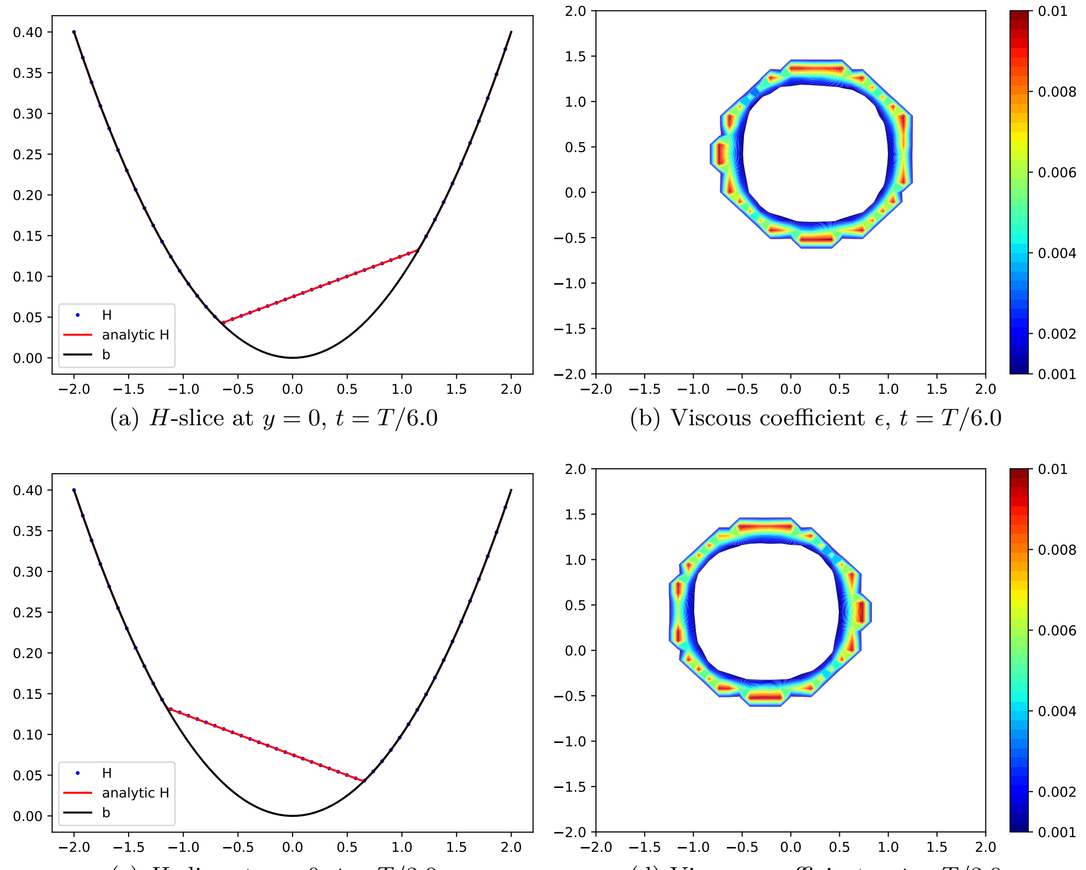
$$\begin{aligned} h(x,y,0) &= \max\left(0, \sigma\frac{h_0}{a^2}\left(2x\cos\omega t + 2y\sin\omega t - \sigma\right) + h_0 - b\right), \\ u(x,y,0) &= -\sigma\omega\sin\omega t, \\ v(x,y,0) &= \sigma\omega\cos\omega t. \end{aligned} \tag{7.9}$$

with parabolic bottom topography

$$b(x,y) = h_0\frac{x^2+y^2}{a^2}, \tag{7.10}$$

with parameters $h_0 = 0.1$, $a = 1$ $\sigma = 0.5$ and $\omega = \frac{\sqrt{2gh_0}}{a}$. Since the boundaries are never reached during the simulation, we simply set them to solid walls. Since there are no shocks, we set the viscosity parameter rather low at $\epsilon_0 = 0.01$. The gravitational constant is set to $g = 9.81$. We use a uniform Cartesian mesh with 200×200 elements for the computation.

This test case demonstrates the capabilities of a numerical method to accurately handle wetting and drying as the fluid evolves. In theory, the lake should travel smoothly around the center of the domain. We show a slice through $y = 0$ and the dynamically set viscosity coefficient at various times in Figure 7.5 and Figure 7.6. The viscosity aligns well with the edges of the lake and the approximated solution matches the analytic one.



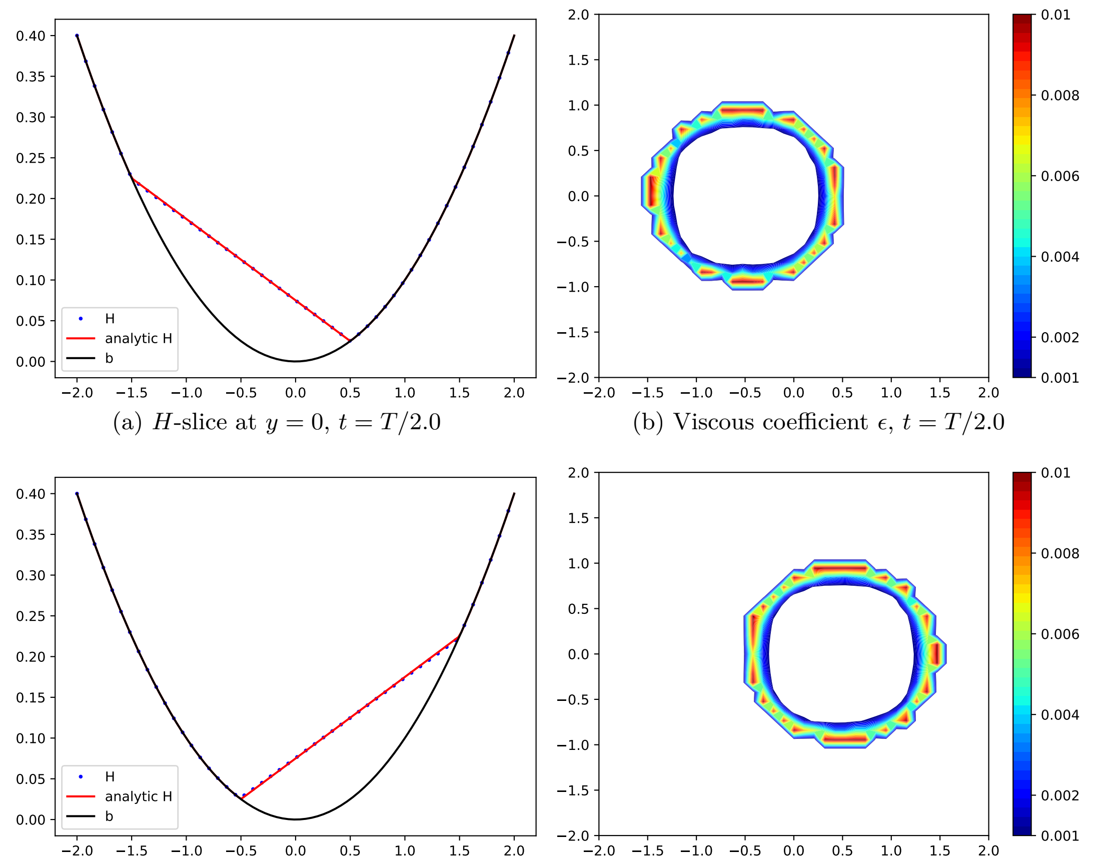
(a) H-slice at $y = 0$, $t = T/6.0$

(b) Viscous coefficient ϵ, $t = T/6.0$

(c) H-slice at $y = 0$, $t = T/3.0$

(d) Viscous coefficient ϵ, $t = T/3.0$

Figure 7.5.: ESDGSEM approximation with artificial viscosity and positivity limiter for the 2D oscillating lake at times $t = T/6$ and $t = T/3$ for $N = 3$. This result was previously published by Wintermeyer et al. in [254].



(a) H-slice at $y = 0$, $t = T/2.0$
(b) Viscous coefficient ϵ, $t = T/2.0$
(c) H-slice at $y = 0$, $t = 2T$
(d) Viscous coefficient ϵ, $t = 2T$

Figure 7.6.: ESDGSEM approximation with artificial viscosity and positivity limiter for the 2D oscillating lake at times $t = T/2$ and $t = 2T$ for $N = 3$. This result was previously published by Wintermeyer et al. in [254].

7.4. Solitary wave around a conical island

We test the entropy stable scheme for a propagating wave around a partly dry conical island in the center of the domain $\Omega = [0, 25] \times [0, 30]$. The wave partly floods the island in the center, flows around it and is reflected at the far end. This process is repeated on the way back. Previous numerical studies of this test case are published in [167, 232] and experimental results are shown in [21]. The initial wave η is defined by

$$\eta(x, y, 0) = \frac{A}{h_0}\operatorname{sech}^2(\gamma(x - x_c)), \tag{7.11}$$

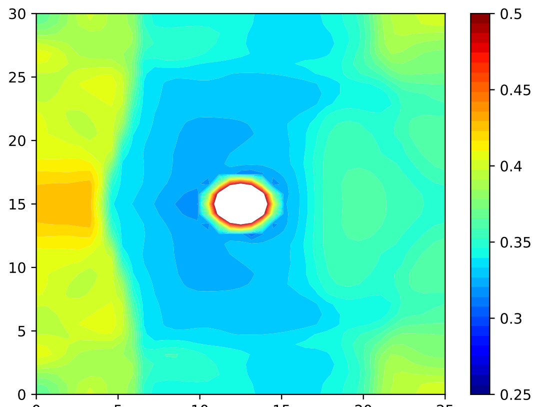
and set on top of a flat water level of $h_0 = 0.32$. The initial conditions can be summarized as

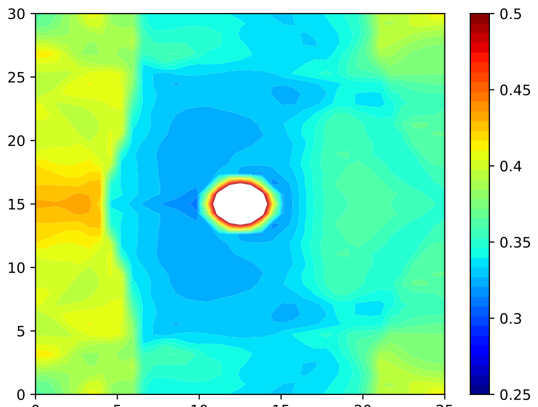
$$\begin{aligned} h(x,y,0) &= \max\left(0, h_0 + \eta(x,y,0) - b(x,y)\right), \\ u(x,y,0) &= \eta(x,y,0)\sqrt{\frac{g}{h_0}}, \\ v &= 0, \end{aligned} \tag{7.12}$$

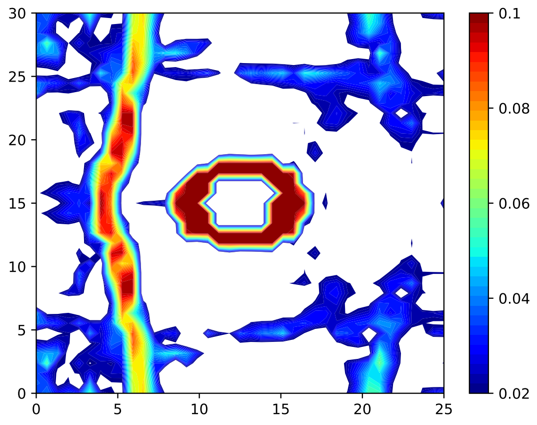
where the parameters are set to $A = 0.064m$, $x_c = 2.5m$, $\gamma = \sqrt{\frac{3A}{4h_0}}$. The conical bottom topography is defined by

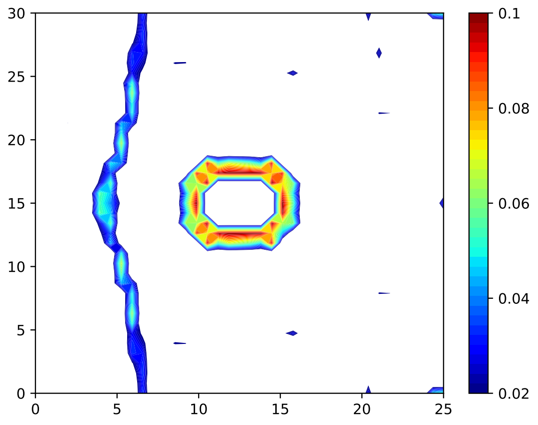
$$b(x,y) = 0.93\left(1 - \frac{r}{r_c}\right) \qquad \text{if } r \leq r_c, \tag{7.13}$$

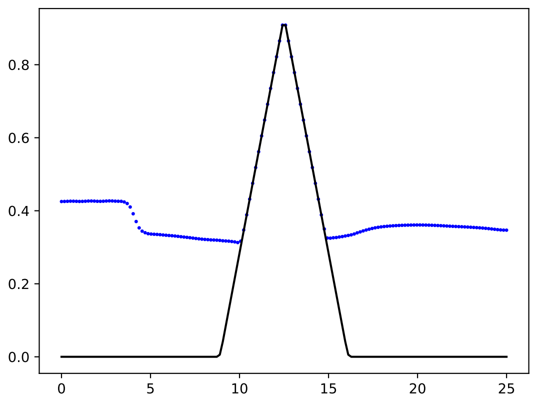
with $r = \sqrt{(x-x_c)^2 + (y-y_c)}$, $r_c = 3.6m$ and $(x_c, y_c) = (12.5, 15)$. The boundaries are set to solid walls. We use a uniform Cartesian mesh with varying spatial resolutions to run the test up to a final time of $T = 50$ and show the results in Figure 7.7. The base viscosity parameter is set to be $\epsilon_0 = 0.1$.


(a) Total water height H, 50×50 Elements


(b) Total water height H, 200×200 Elements


(c) Viscous coefficient ϵ, 50×50 Elements


(d) Viscous coefficient ϵ, 200×200 Elements


(e) H-Slice at $y = 15$, 50×50 Elements

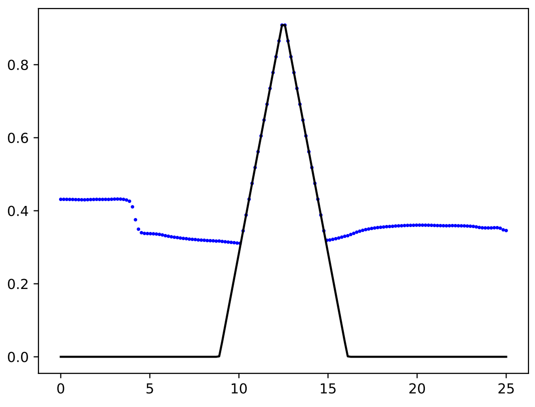

(f) H-Slice at $y = 15$, 200×200 Elements

Figure 7.7.: ESDGSEM approximation with artificial viscosity and positivity limiter for the solitary wave runup for different grid resolutions with $N = 3$ at $T = 50$. This result was previously published by Wintermeyer et al. in [254].

7.5. Three mound dam break

The following test combines the flooding of dry areas with the strong shock of a dam break. It is a very challenging example that thoroughly tests the shock capturing as well as the positivity limiter. The dam break is set up at $x = 16$ on the domain $\Omega = [0, 75] \times [0, 45]$ with a water height of $h = 1.875$ to the left, and no water everywhere else. There are also three mountains on the downstream side which will be (partially) flooded during the simulation. Formally, the initial state is given by

$$\begin{aligned} h(x, y, 0) &= \begin{cases} 1.875, & \text{if } x < 16 \\ 0, & \text{otherwise} \end{cases}, \\ u &= v = 0. \end{aligned} \tag{7.14}$$

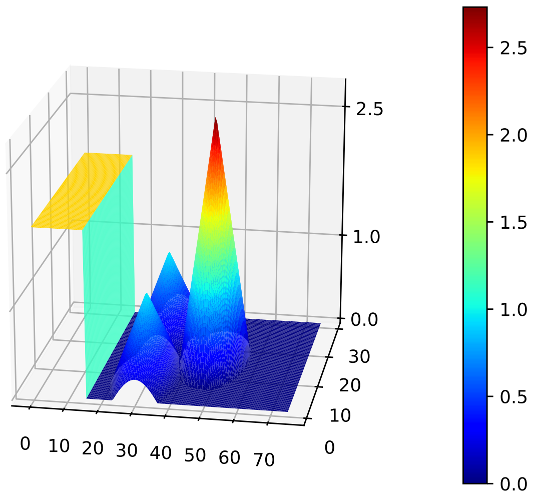
The three mounds on the down hill side are given by

$$\begin{aligned} M_1(x, y) &:= 1 - 0.1\sqrt{(x-30)^2 + (y-22.5)^2}, \\ M_2(x, y) &:= 1 - 0.1\sqrt{(x-30)^2 + (y-7.5)^2}, \\ M_3(x, y) &:= 2.8 - 0.28\sqrt{(x-47.5)^2 + (y-15)^2}, \end{aligned} \tag{7.15}$$

and the bottom topography is taken as the maximum elevation level of the three mounds

$$b(x, y) = \max(0, M_1(x, y), M_2(x, y), M_3(x, y)) \tag{7.16}$$

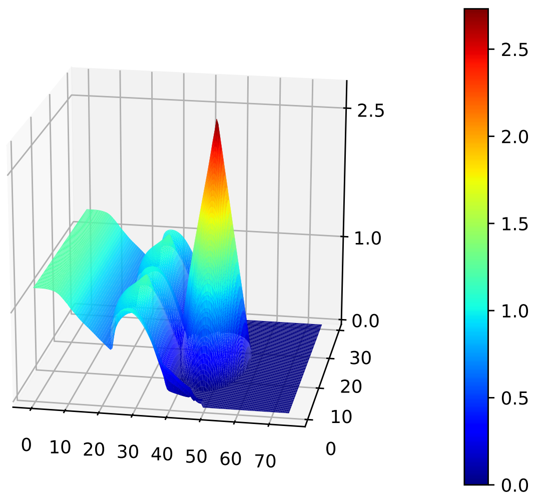
The domain is closed off by solid wall boundaries on all four sides. The domain is

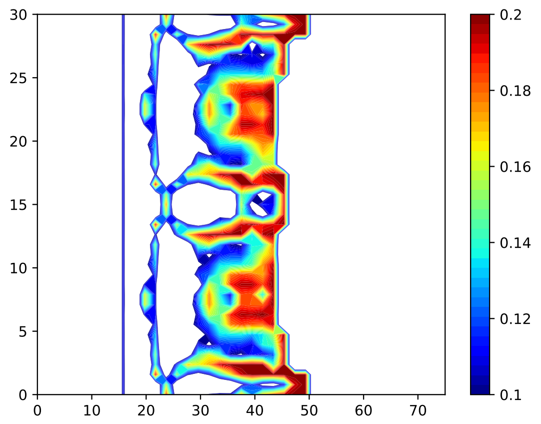


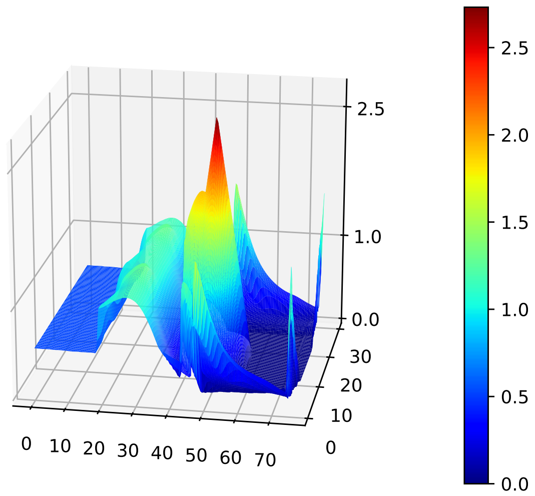
(a) H, $T = 0.0$

Figure 7.8.: Initial condition for the dam break over three mounds $N = 3$.

discretized by a Cartesian mesh with 150×100 elements. The combination of a strong shock and a dry area with steep bathymetry gradients requires a relatively high base viscosity parameter at $\epsilon_0 = 0.2$. The usual gravitational constant $g = 9.81$ is used in this simulation. The total water height at various times is shown in Figure 7.8 and Figure 7.9. The dynamic viscous coefficient ϵ is displayed as well. These plots show that the shock capturing mechanism accurately tracks the shock fronts across the domain and around the three mounds.


(a) H, $T = 5.0$


(b) Viscous coefficient ϵ, $T = 5.0$


(c) H, $T = 10.0$

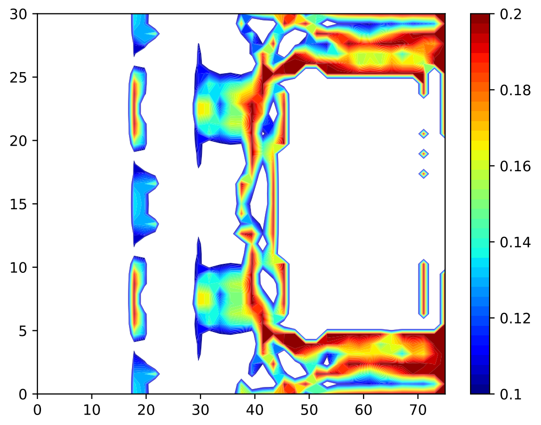
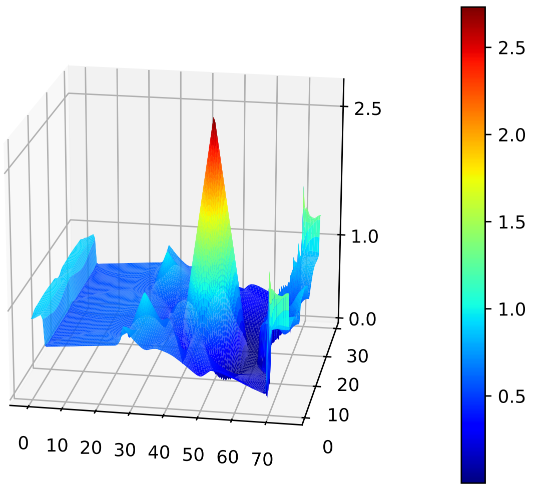
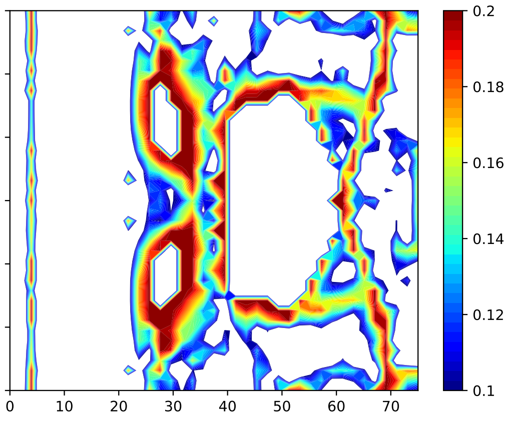

(d) Viscous coefficient ϵ, $T = 10.0$

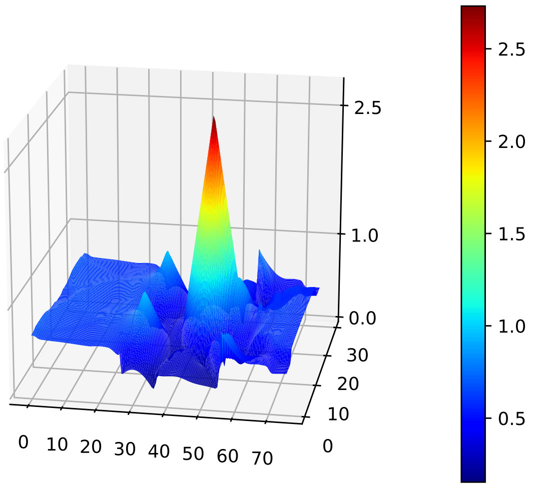
Figure 7.9.: ESDGSEM approximation with artificial viscosity and positivity limiter for the dam break over three mounds $N = 3$. This result was previously published by Wintermeyer et al. in [254].



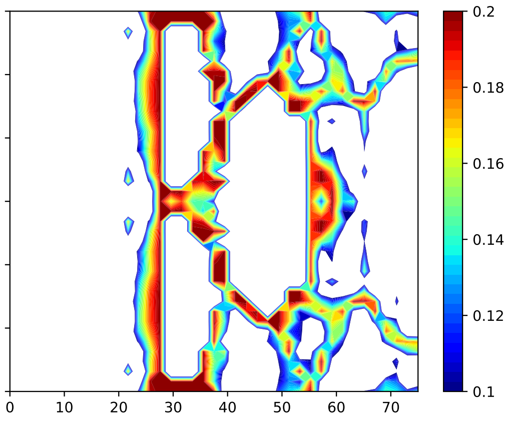
(a) H, $T = 20.0$



(b) Viscous coefficient ϵ, $T = 20.0$

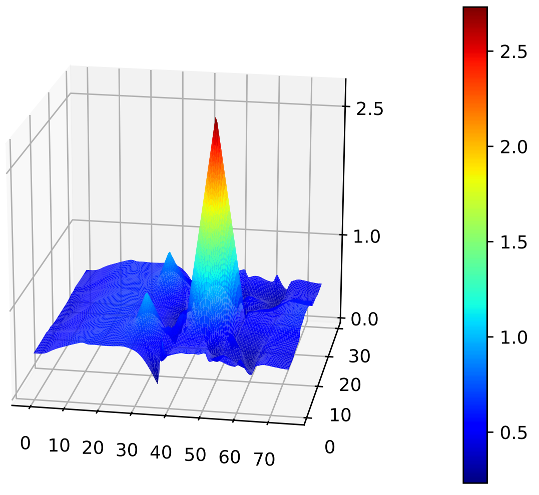


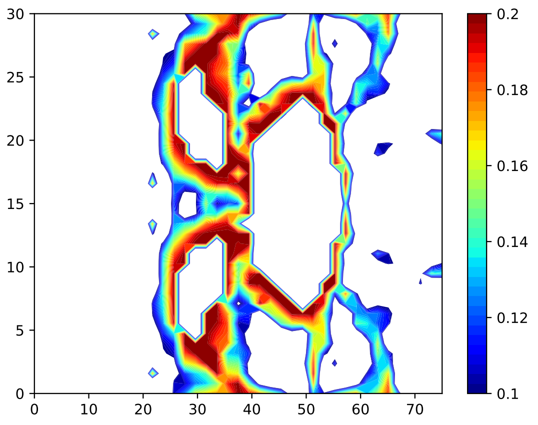
(c) H, $T = 30.0$

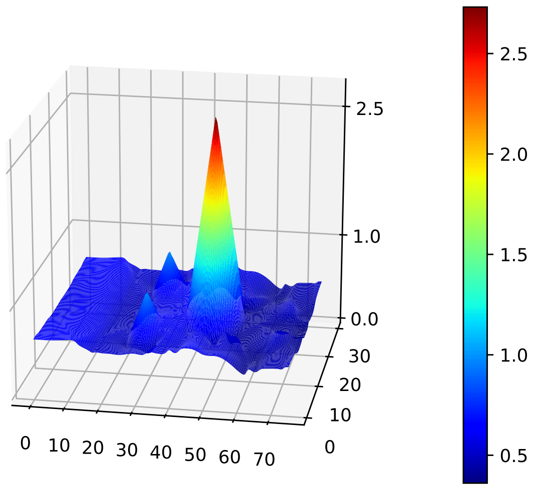


(d) Viscous coefficient ϵ, $T = 30.0$

Figure 7.10.: ESDGSEM approximation with artificial viscosity and positivity limiter for the dam break over three mounds $N = 3$. This result was previously published by Wintermeyer et al. in [254].


(a) H, $T = 40.0$


(b) Viscous coefficient ϵ, $T = 40.0$


(c) H, $T = 50.0$

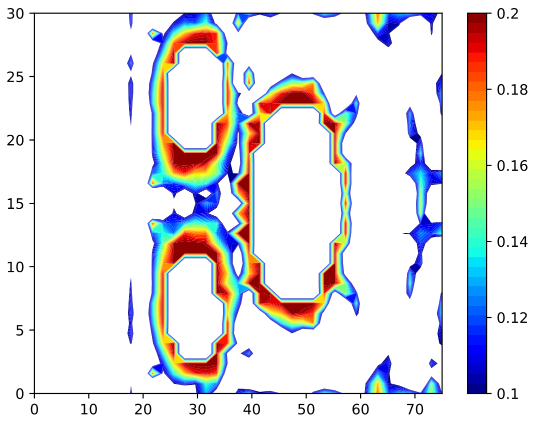

(d) Viscous coefficient ϵ, $T = 50.0$

Figure 7.11.: ESDGSEM approximation with artificial viscosity and positivity limiter for the dam break over three mounds $N = 3$. This result was previously published by Wintermeyer et al. in [254].

7.6. Parabolic partial dam break

To test the ESDGSEM for challenging problems on a curved mesh as well, we set up a parabolic dam break problem as in [253, 254]. The 40 by 40 element mesh is shaped such that it aligns with a parabolic dam in the center of the domain. The initial condition and the mesh are shown in Figure 7.12. The entropy stable scheme without shock

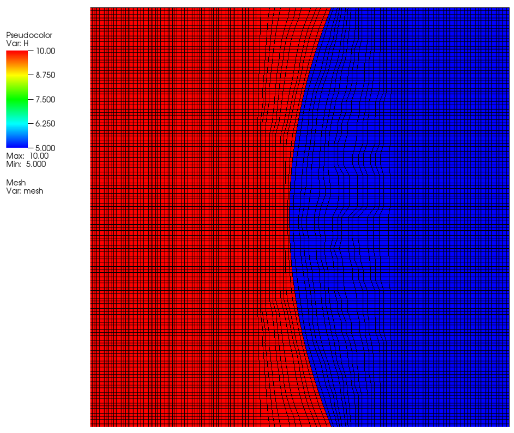
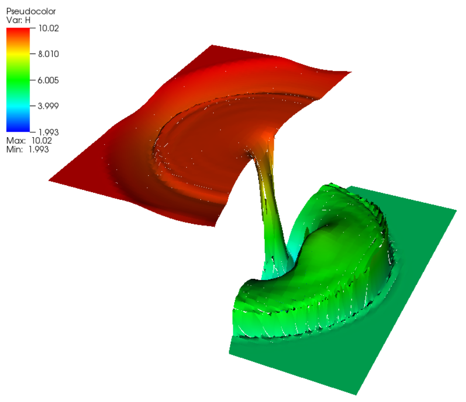


Figure 7.12.: Initial condition and mesh for the parabolic dam break test case.

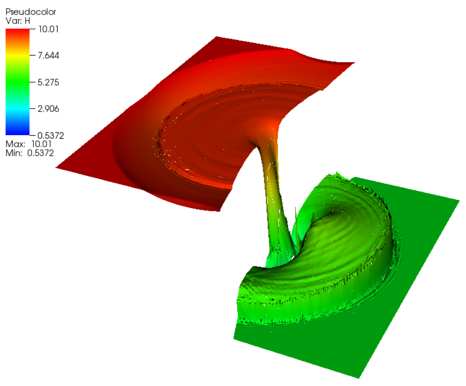
capturing demonstrated increased robustness compared to a standard DGSEM, however, it also contained oscillations in the shock region [253]. We demonstrate how the dynamic artificial viscosity reduces the oscillations on the shock front. The initial setup is given by

$$\begin{aligned} h &= \begin{cases} 10, & \text{if } x < \dfrac{1}{25}y^2 - 0.25 \\ 5, & \text{otherwise} \end{cases}, \\ u &= v = 0. \end{aligned} \tag{7.17}$$

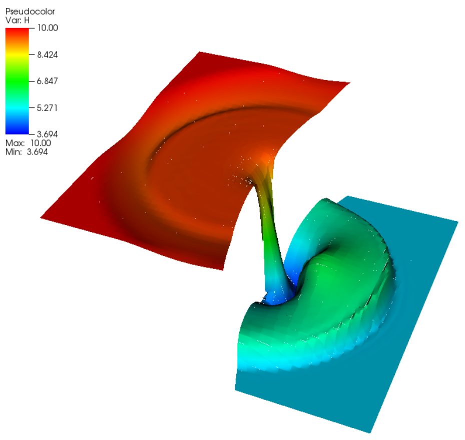
The gravitational constant is set to $g = 1.0$ here. To remove any other influences, we use a flat bottom topography for this test case. While the scheme still works for discontinuous bottoms, the multitude of different effects makes it hard to observe the impact of the artificial viscosity alone. We compare the results for $N = 3$ and $N = 7$ with and without added stabilization and show the the calculated dynamic viscosity coefficients in Figure 7.13. The base viscosity coefficient is set relatively low at $\epsilon_0 = 0.025$. The stabilizing impact of the artificial viscosity is clearly visible. Oscillations have dramatically reduced at the shock front and also at the waves on the top side of the dam. The overshoot spikes close to the center of the dam break are significantly smaller. From the dynamic viscosity coefficient plots we can see that the shock front as well as the back waves at the top are smoothed by viscosity, whereas other smooth regions are not impacted. It is also visible that the viscosity works more accurately for finer resolutions. For $N = 7$, not only less elements are affected, but less viscosity is needed overall.



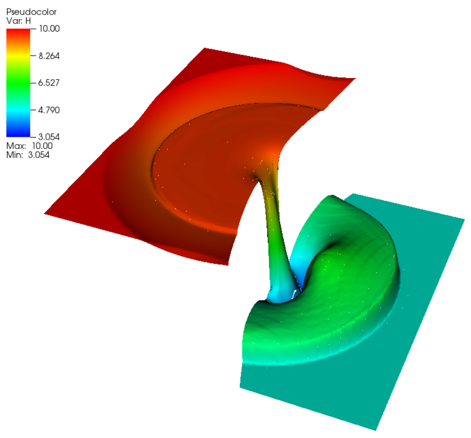
(a) Total water height H without AV, $N = 3$



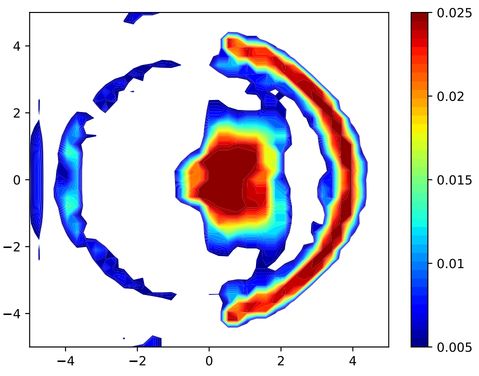
(b) Total water height H without AV, $N = 7$



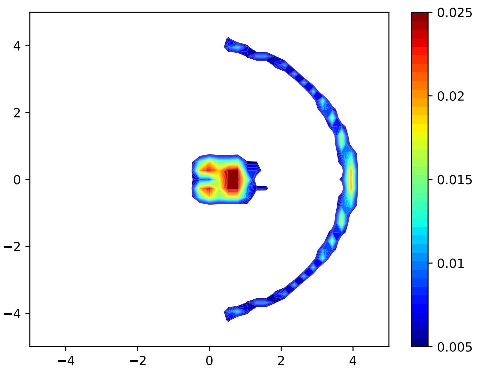
(c) Total water height H with AV, $N = 3$



(d) Total water height H with AV, $N = 7$



(e) Viscous coefficient ϵ, $N = 3$



(f) Viscous coefficient ϵ, $N = 7$

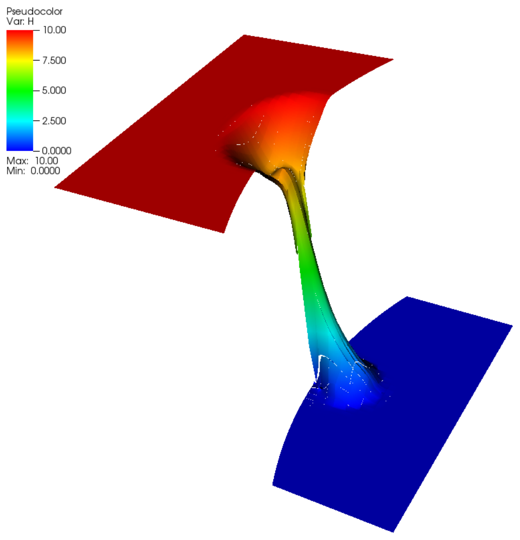
Figure 7.13.: ESDGSEM approximation for the curved dam break with and without artificial viscosity (AV) at $N = 3$ and $N = 7$ at $T = 1.5$. This result was previously published by Wintermeyer et al. in [254].

7.7. Wet/dry parabolic partial dam break

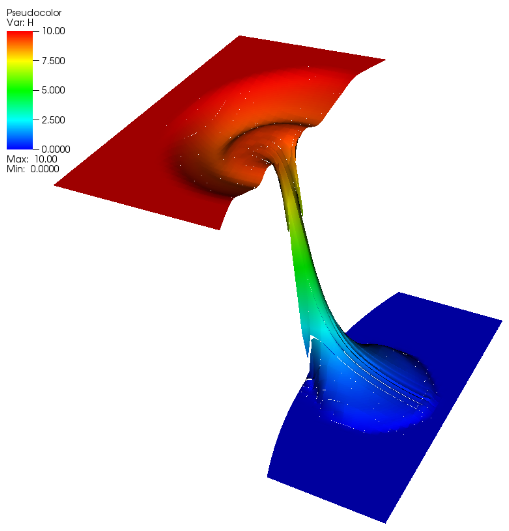
We modify the parabolic dam break problem from Section 7.6 to feature a dry area on the downstream side. This modification makes it very challenging as it is a massive shock hitting a dry area and requires the artificial viscosity as well as the positivity preserving limiter. The initial setup is given by

$$\begin{aligned} h &= \begin{cases} 10, & \text{if } x < \dfrac{1}{25}y^2 - 0.25 \\ 0, & \text{otherwise} \end{cases}, \\ u &= v = 0. \end{aligned} \tag{7.18}$$

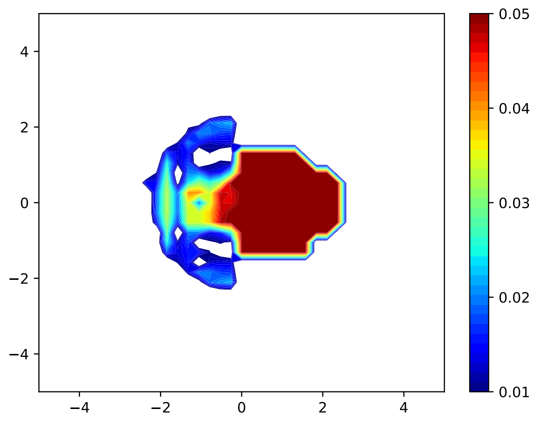
The same mesh shown in Figure 7.12 is used for this test as well. The gravitational constant is kept at $g = 1.0$ but a slightly higher viscosity parameter is used with $\epsilon_0 = 0.05$ for $N = 3$. For $N = 7$ the same viscosity parameter as in Section 7.6 is used at $\epsilon_0 = 0.025$. The solution as well as the dynamic viscosity parameter are depicted for $N = 3$ in Figure 7.14 and for $N = 7$ in Figure 7.15. As the reduced water level on the downstream side leads to a steeper dam break than in the completely wet case from Section 7.6, the final time is reduced to $T = 1.0$ such that the water does not hit the back wall. The results show that the dynamic viscosity is only applied to the critical regions.



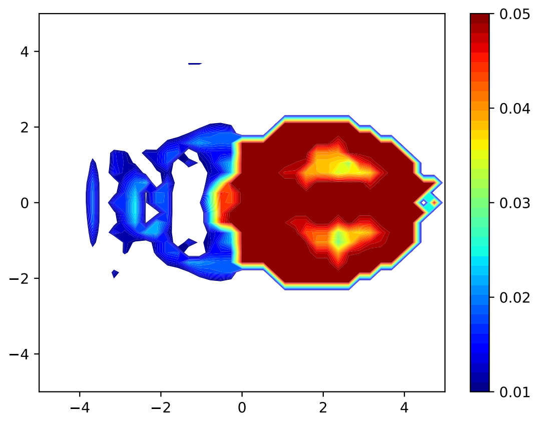
(a) Water height H, $T = 0.5$



(b) Water height H, $T = 1.0$

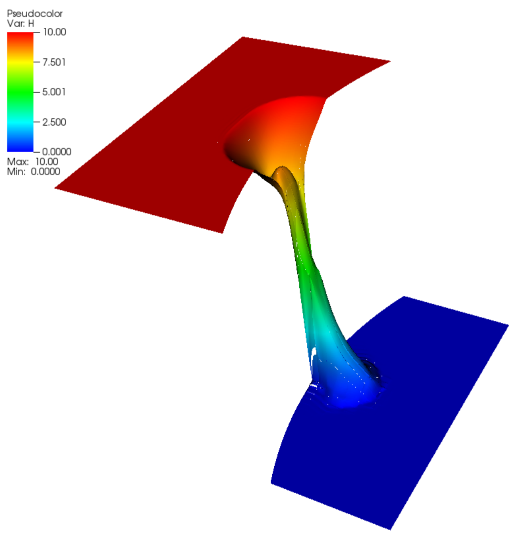


(c) Viscous coefficient ϵ, $T = 0.5$

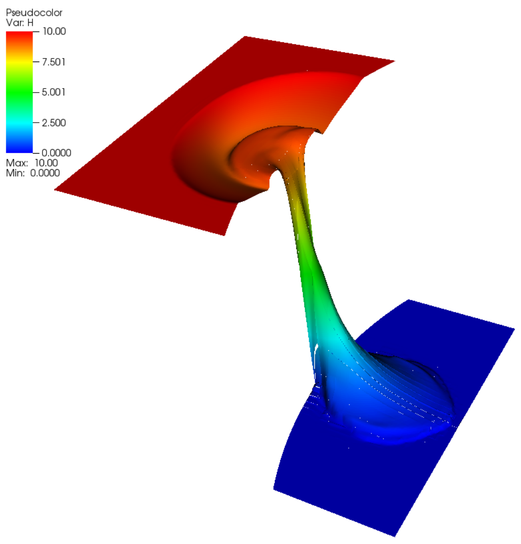


(d) Viscous coefficient ϵ, $T = 1.0$

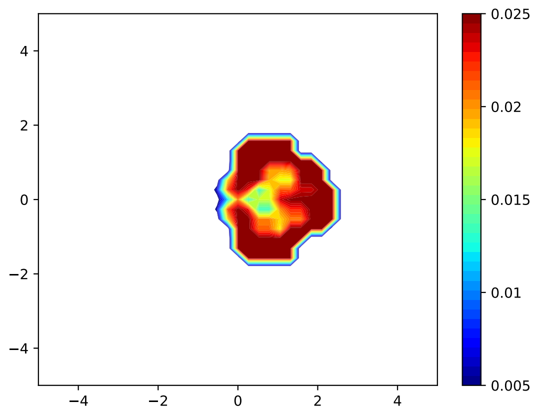
Figure 7.14.: ESDGSEM approximation with artificial viscosity for the curved dam break with zero water height on the downstream side at $N = 3$ on a 40×40 curved mesh. This result was previously published by Wintermeyer et al. in [254].



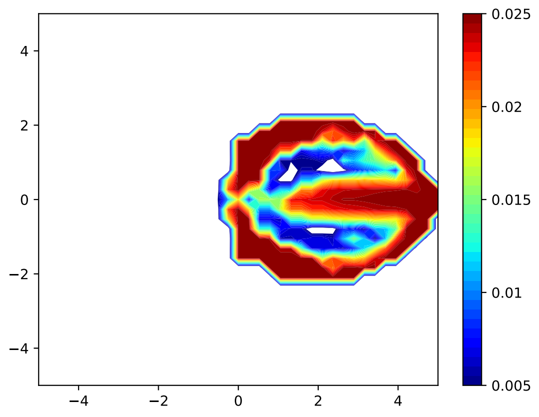
(a) Water height H, $T = 0.5$



(b) Water height H, $T = 1.0$



(c) Viscous coefficient ϵ, $T = 0.5$



(d) Viscous coefficient ϵ, $T = 1.0$

Figure 7.15.: ESDGSEM approximation with artificial viscosity for the curved dam break with zero water height on the downstream side at $N = 7$ on a 40×40 curved mesh. This result was previously published by Wintermeyer et al. in [254].

8. Tsunami Simulation

8.1. The 2004 Indian Ocean tsunami

On December, 26th, in 2004 at about 0:58 AM UTC one of the largest earthquakes ever recorded took place about 80 km to the west of the coast of Sumatra and triggered the Indian Ocean tsunami, one of the most devastating natural catastrophes of the modern age. The tsunami hit the coasts of many countries of South and Southeast Asia. Over the course of ten hours it reached coastal areas as far as East Africa. Wave heights of 30m and more were reported, the largest ones occurring on the shore of Sumatra. The total casualties exceed $292,000$ people in twelve countries, Indonesia alone reporting over $200,000$ dead. Thailand, Sri Lanka and India were among the most affected countries, as well [190].

The disaster was particularly devastating as the countries were wholly unprepared for the magnitude and impact. Researchers have intensified the efforts to investigate the phenomenon of tsunamis afterwards such that strategies and plans may be developed to properly handle possible future tsunamis. Early detection as well as accurate predictions of the tsunami arrival times would be immensely useful in this regard [197, 198, 97, 116].

8.2. The extended SW model

To simulate real world applications accurately, some extensions must be included in the shallow water equations. Specifically, one must account for the bottom friction force and the earth Coriolis force. Each is modelled with an additional source term in the momentum equations. The fully extended shallow water equations for oceanic modelling are then given by

$$\begin{aligned} h_t + (hu)_x + (hv)_y &= 0, \\ (hu)_t + \left(hu^2 + \frac{1}{2} g\, h^2\right)_x + (huv)_y &= -ghb_x - \tau_{bx} + f_{\text{cor}} hv, \\ (hv)_t + (huv)_x + \left(hv^2 + \frac{1}{2} g\, h^2\right)_y &= -ghb_y - \tau_{by} - f_{\text{cor}} hu. \end{aligned} \tag{8.1}$$

The Coriolis parameter $f_{\text{cor}} = 2\omega \sin(\phi)$ depends on the earth's angular velocity

$$\omega = 2\pi/(24 \times 3600) \approx 7.27 \times 10^{-5} \, \frac{\text{rad}}{s},$$

and the positional latitude ϕ. The additional friction term is formulated according to Chézy-Manning-Strickler [93, 212, 75] and given by

$$\begin{aligned}\tau_{bx} &= \frac{n^2 g}{h^{7/3}}\sqrt{(hu)^2 + (hv^2)}hu,\\ \tau_{by} &= \frac{n^2 g}{h^{7/3}}\sqrt{(hu)^2 + (hv^2)}hv.\end{aligned} \tag{8.2}$$

The Manning coefficient n is in this case chosen to be $0.025s/m^{1/3}$, a typical value for sand. The Manning friction is negligibly small in the deep ocean, but larger close to the shore. It offers a stabilizing effect on the waves hitting the shallow regions in the coastal areas. The maximum absolute Manning frictions over ten hour simulations of the Indian Ocean tsunami are depicted for polynomial degrees $N = 3$ and $N = 7$ in Figure 8.11.

8.3. The discrete setup

The mesh is constructed based on real world coastline data to adequately model the simulation. The coastline data sets are available at the GSHHG data base [249]. We use a quadrilateral mesh of the Indian Ocean with 4928 elements generated by SpecMesh from D.A. Kopriva [136] as depicted in Figure 8.1.

Accurate real world bathymetry data is provided by the National Center for Environmental Information (NOAA) in their ETOPO data sets [2]. The bathymetry interpolated to the LGL nodes on the mesh in Figure 8.1 is shown for polynomial degrees $N = 3$ and $N = 7$ in Figure 8.2. The earthquake locations are estimated from tide gauge measurements of the wave propagation [201]. The initial surface displacement caused by the earthquake is then estimated according to the Okada model [191]. While the five different earthquakes are slightly time delayed, they are aggregated for simplicity to obtain an accurate representation of the surface displacement at t_0. The full fault plane data that is used as input for the Okada model can be found in [241, 116] and is listed in Figure 8.3. The resulting initial free surface elevation estimated by the Okada model is shown in Figure 8.4.

There are generally two different kind of boundaries in the Indian Ocean mesh. First, there are the coastlines of Africa, India, Thailand and Indonesia as well as the islands Madagascar, Sri Lanka and Sumatra. These boundaries are treated numerically as solid wall boundaries such that approaching waves are reflected. Then, there are the open ocean boundaries to the south and south-east. These must be treated as outflow boundaries such that there are no unphysical reflections back into the domain. If the outer state is fixed at still sea level, $(h_0, 0, 0)$, over the course of the simulations, the entropy stable numerical flux constructed in (5.101) handles this correctly. There are no visible reflections over the course of the ten hour simulation.

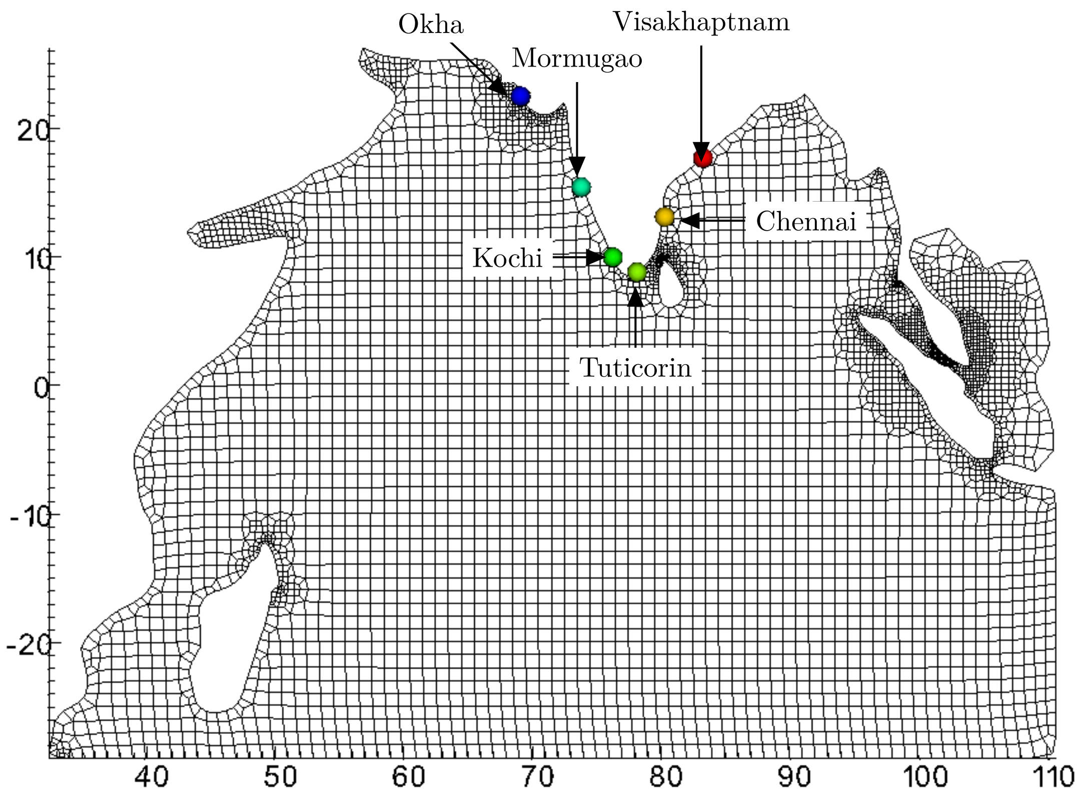
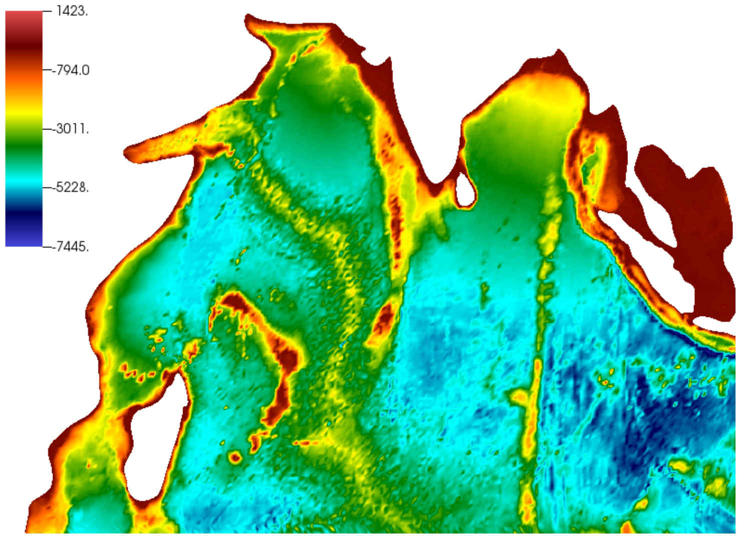
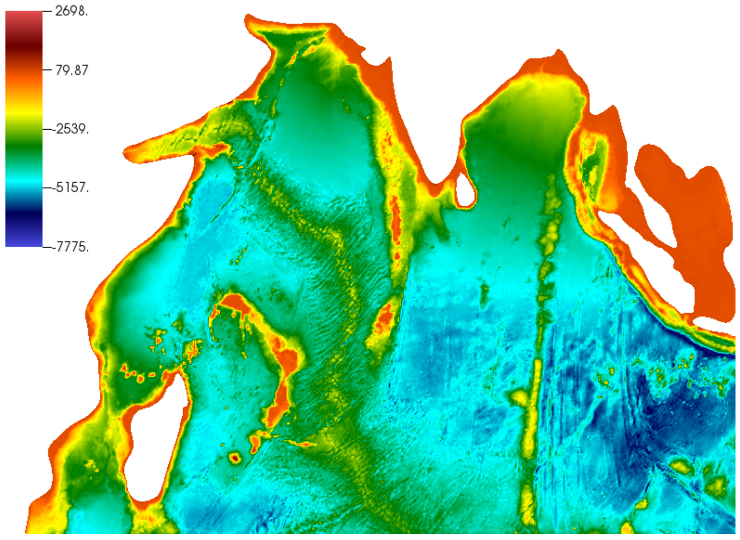


Figure 8.1.: Quadrilateral mesh of the Indian Ocean. Coordinates are given in longitude and latitude. Depicted are also the six gauge measurement stations on the Indian coast used for the comparison of tsunami arrival times. The mesh was provided by D.A. Kopriva [136].



(a) $N = 3$.



(b) $N = 7$.

Figure 8.2.: Bottom topography of the Indian Ocean interpolated for $N = 3$ and $N = 7$. The data was obtained by interpolating of the ETOPO data sets [2].

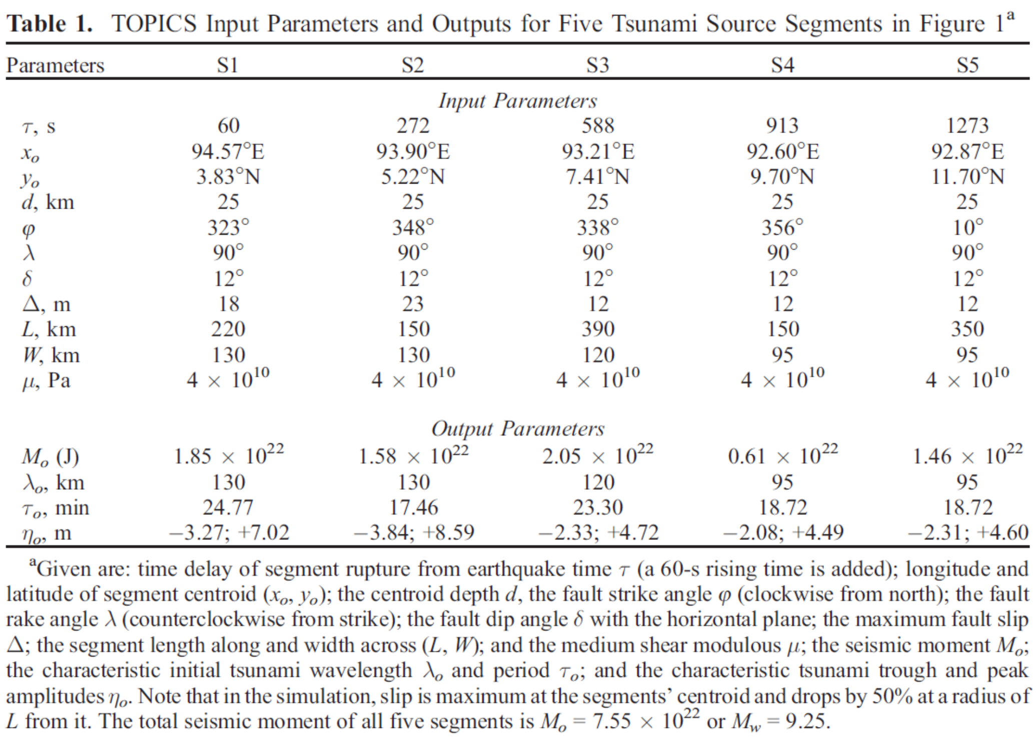


Table 1. TOPICS Input Parameters and Outputs for Five Tsunami Source Segments in Figure 1[a]

Parameters	S1	S2	S3	S4	S5
			Input Parameters		
τ, s	60	272	588	913	1273
x_o	94.57°E	93.90°E	93.21°E	92.60°E	92.87°E
y_o	3.83°N	5.22°N	7.41°N	9.70°N	11.70°N
d, km	25	25	25	25	25
φ	323°	348°	338°	356°	10°
λ	90°	90°	90°	90°	90°
δ	12°	12°	12°	12°	12°
Δ, m	18	23	12	12	12
L, km	220	150	390	150	350
W, km	130	130	120	95	95
μ, Pa	4×10^{10}	4×10^{10}	4×10^{10}	4×10^{10}	4×10^{10}
			Output Parameters		
M_o (J)	1.85×10^{22}	1.58×10^{22}	2.05×10^{22}	0.61×10^{22}	1.46×10^{22}
λ_o, km	130	130	120	95	95
τ_o, min	24.77	17.46	23.30	18.72	18.72
η_o, m	−3.27; +7.02	−3.84; +8.59	−2.33; +4.72	−2.08; +4.49	−2.31; +4.60

[a]Given are: time delay of segment rupture from earthquake time τ (a 60-s rising time is added); longitude and latitude of segment centroid (x_o, y_o); the centroid depth d, the fault strike angle φ (clockwise from north); the fault rake angle λ (counterclockwise from strike); the fault dip angle δ with the horizontal plane; the maximum fault slip Δ; the segment length along and width across (L, W); and the medium shear modulous μ; the seismic moment M_o; the characteristic initial tsunami wavelength λ_o and period τ_o; and the characteristic tsunami trough and peak amplitudes η_o. Note that in the simulation, slip is maximum at the segments' centroid and drops by 50% at a radius of L from it. The total seismic moment of all five segments is $M_o = 7.55 \times 10^{22}$ or $M_w = 9.25$.

Figure 8.3.: Specifications of the five earthquakes that caused the Indian Ocean tsunami [116]. This data is used as the input for the Okada model to approximate the initial surface displacement [191].

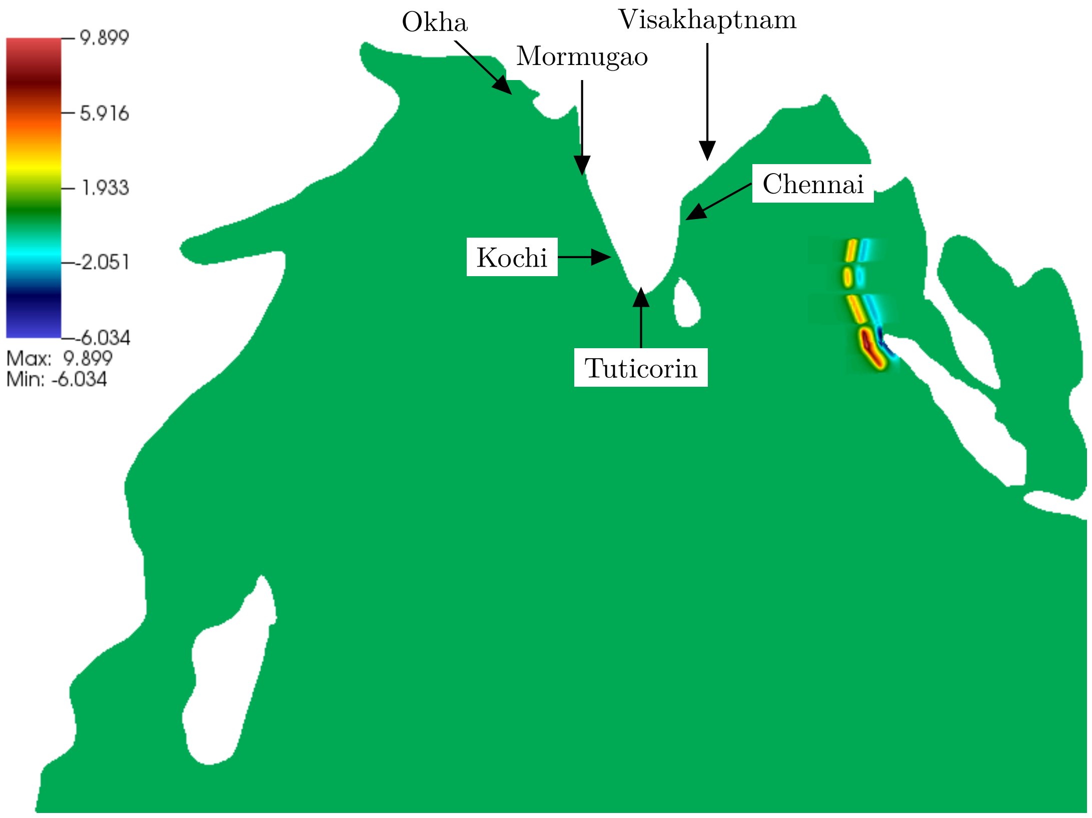


Figure 8.4.: Free surface elevation with initial surface displacement in meters approximated according to the Okada model. Also shown are the six tidal gauge measurement stations used validation.

8.4. Results

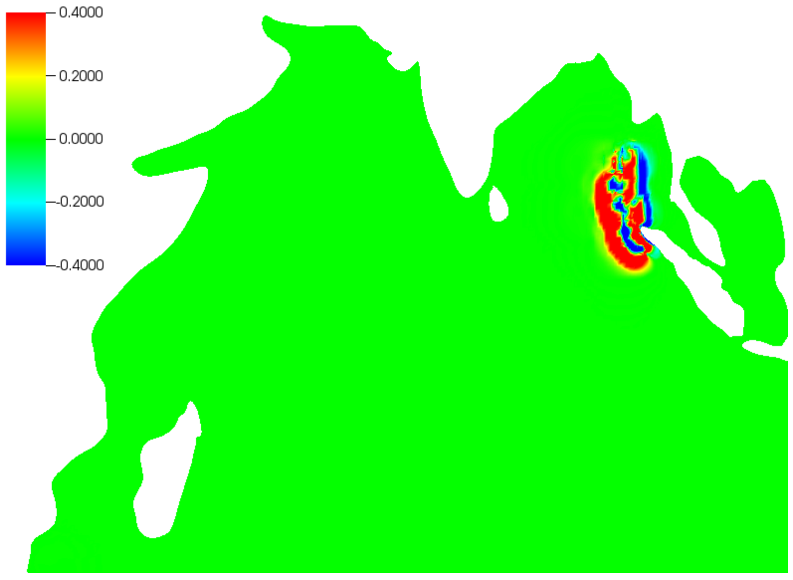
We simulate the first ten hours of the Indian Ocean tsunami but we put a special emphasis on the early stages of the tsunami. This is when the coast interaction with Sumatra takes place and also when the waves first reach Sri Lanka and the Indian coast. Thus, we show free surface elevations for polynomial orders $N = 3$ and $N = 7$ at 20,40,80 and 160 minutes after the earthquake in Figure 8.5, Figure 8.6, Figure 8.7 and Figure 8.8. These results compare well to previously reported results [75] and are consistent to each other. While both approximations track the wave front similarly, the high-order approximation provides more detail on the flow behaviour, particularly behind the shock front. This can be observed well at $t = 40$ in Figure 8.6 and also at $t = 80$ in Figure 8.7.

We are also interested in evaluating certain characteristics of the tsunami over the entire ten hour runtime of the simulation to get a general impression on its behaviour. First, we are interested in approximating the tsunami arrival times after the initial earthquake everywhere in the Indian Ocean. Here, the tsunami arrival time is defined as the first time the free surface elevation has been perturbed by $0.05m$. We show a map of arrival times in Figure 8.9. We observe that the propagation of the wave is slowed down in shallower regions and at the coast.

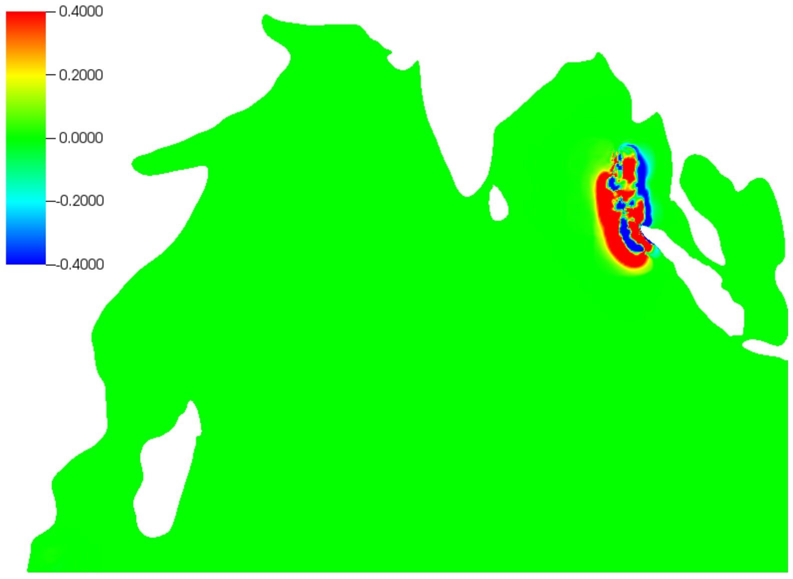
Also, we are interested in the maximum free surface elevations in every region that was reached during the full simulation. These values are reported in Figure 8.10. We see particularly high waves occur not only in coastal areas, but also in regions where the water is relatively shallow in the center of the domain. The waves propagating through the deep ocean to the south of the earthquake do not reach similar heights.

Lastly, we record the maximum Manning friction over the course of the simulation. This source term was introduced to model the dampening effect of coastal areas, so it is not surprising to see the largest values in specifically these regions. The achieved values are shown in Figure 8.11

We validate the approximations of the ESDGSEM by comparison to available tidal gauge measurements from several research stations on the Indian coast provided by the CSIR, the Indian National Institute of Oceanography [48]. The research stations we use are shown in Figure 8.1. The approximated free surface elevations over the course of the ten hour simulation are shown in Figure 8.12. As all the tidal gauge stations are located near the coast, measurements after the initial wave are relatively inaccurate. This is because the wave runup onto the shore is not well modeled in the shallow water equations. To allow for realistic wave breaking other models such as the Boussinesq equations must be considered [101]. In this ESDGSEM approximations, we simply model the shore interactions by solid wall boundary conditions. These are reflective and not absorbing as real coast line beaches may be. This is mitigated by the Manning friction that dampens the waves close to the shore. As this is a rather crude approximation, we cannot expect accurate results near the shore after arrival of the initial wave. However, the results in Figure 8.12 show that tsunami arrival times are approximated reasonably well, albeit not always with the correct magnitude of the wave. Similar results were presented by Gandham [75], reaffirming the suspicion that these inaccuracies are due to



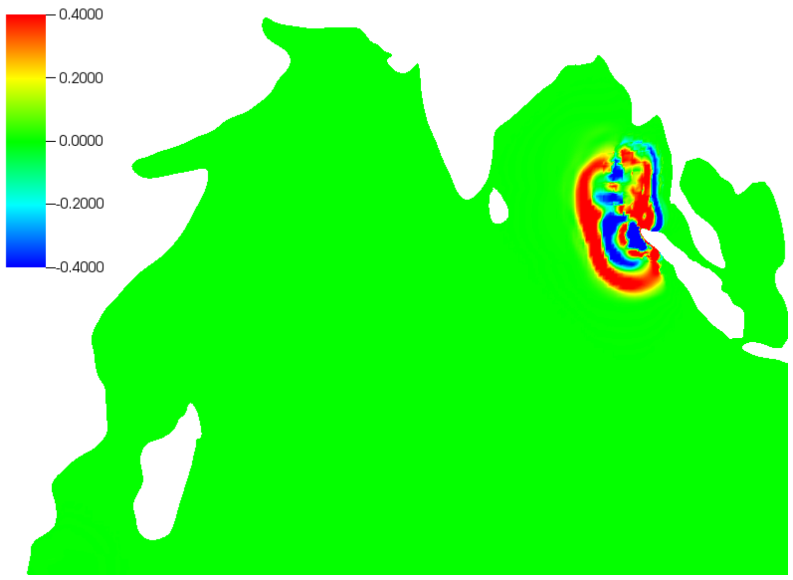
(a) $t = 20$, $N = 3$.



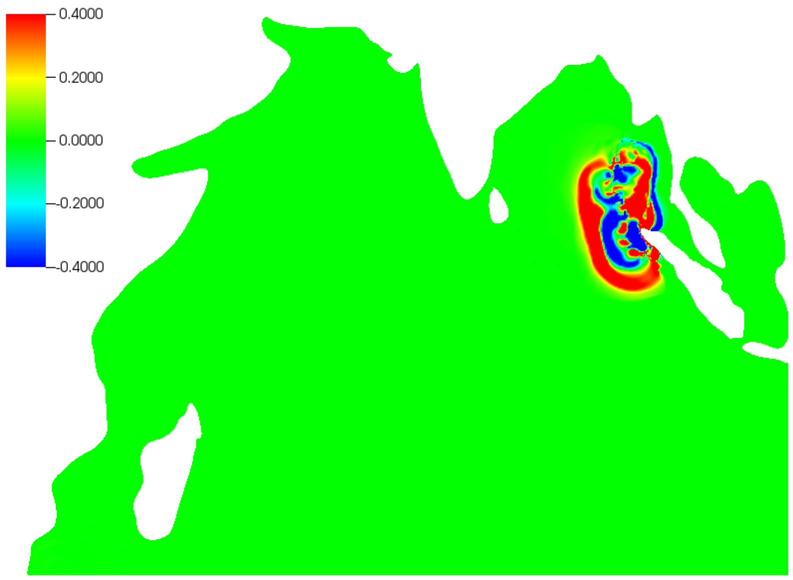
(b) $t = 20$, $N = 7$.

Figure 8.5.: ESDGSEM approximation of the free surface elevation during the Indian Ocean tsunami at $t = 20$min after the initial earthquake for polynomial orders $N = 3$ and $N = 7$. The scale for the free surface height is set to $[-0.4, 0.4]$m for comparison to previously published results by Gandham [75].

the shallow water model applied to coastal regions.

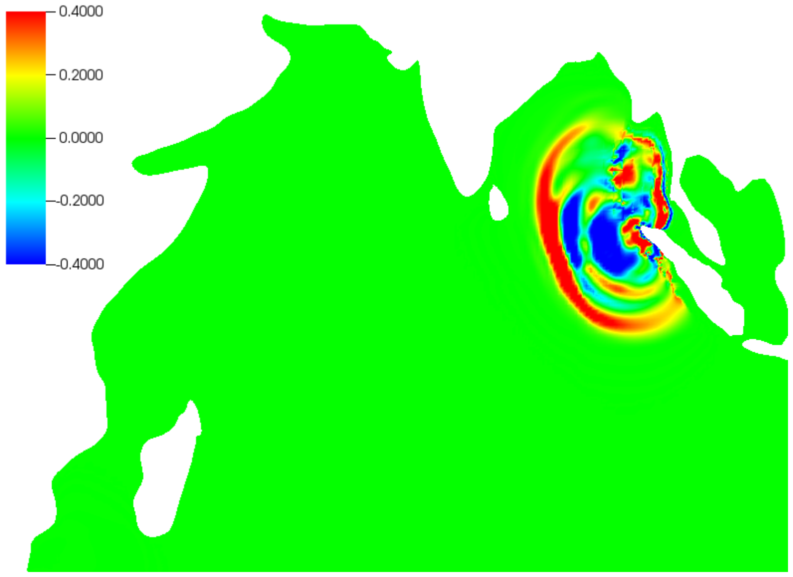


(a) $t = 40$, $N = 3$.

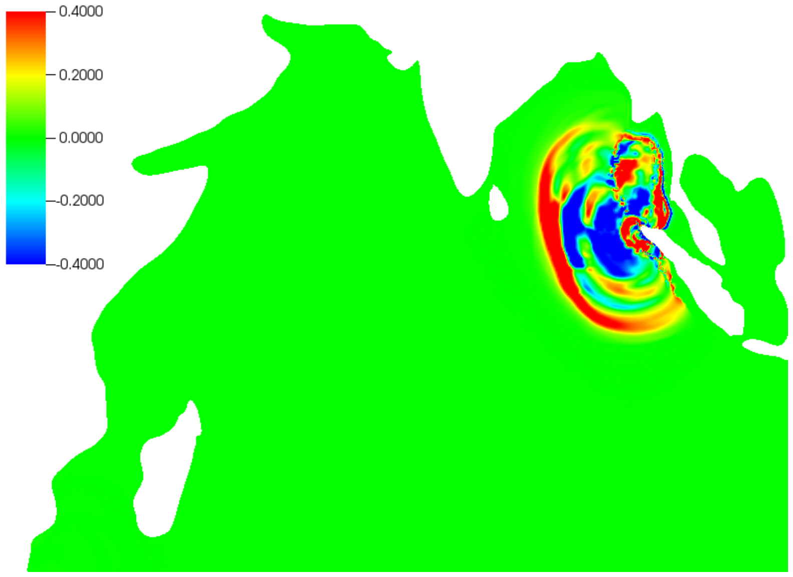


(b) $t = 40$, $N = 7$.

Figure 8.6.: ESDGSEM approximation of the free surface elevation during the Indian Ocean tsunami at $t = 40$min after the initial earthquake for polynomial orders $N = 3$ and $N = 7$. The scale for the free surface height is set to $[-0.4, 0.4]$m for comparison to previously published results by Gandham [75].

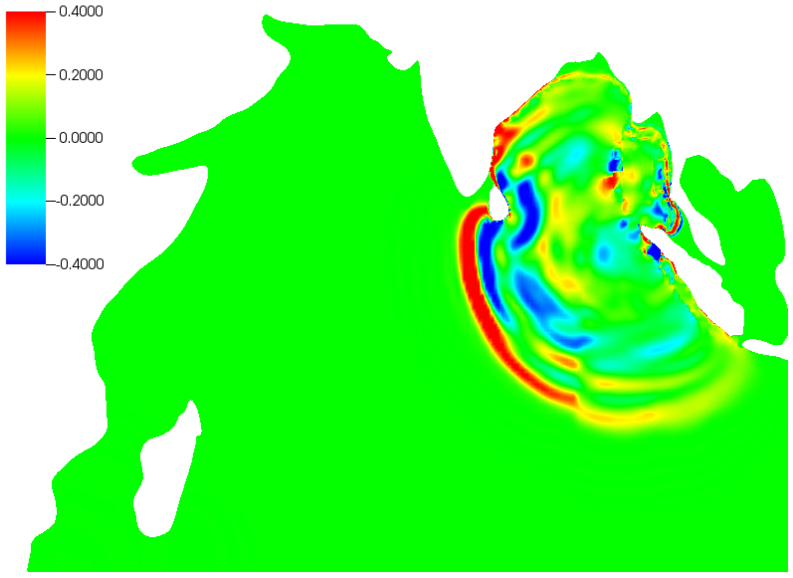


(a) $t = 80$, $N = 3$.

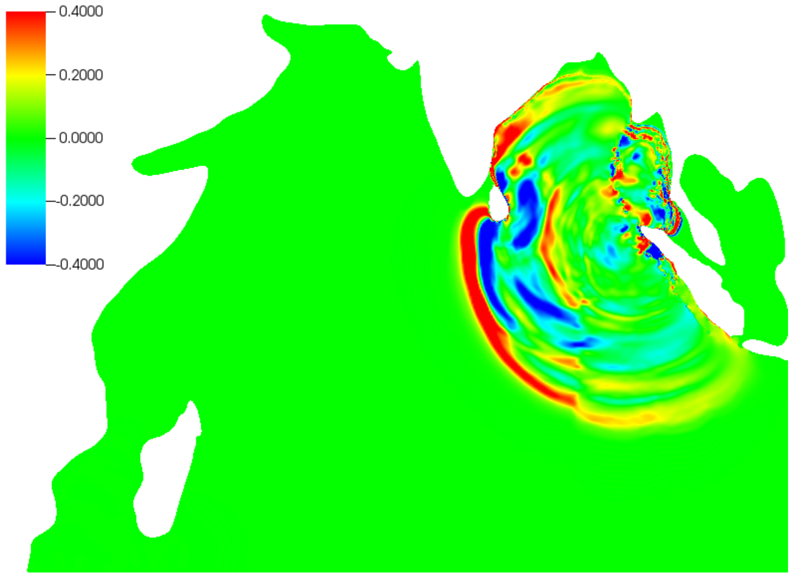


(b) $t = 80$, $N = 7$.

Figure 8.7.: ESDGSEM approximation of the free surface elevation during the Indian Ocean tsunami at $t = 80$min after the initial earthquake for polynomial orders $N = 3$ and $N = 7$. The scale for the free surface height is set to $[-0.4, 0.4]$m for comparison to previously published results by Gandham [75].

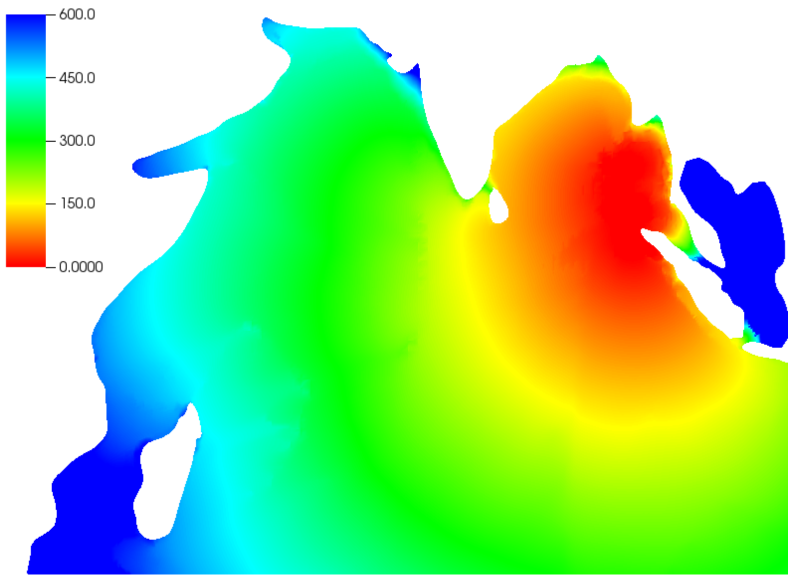


(a) $t = 160$, $N = 3$.

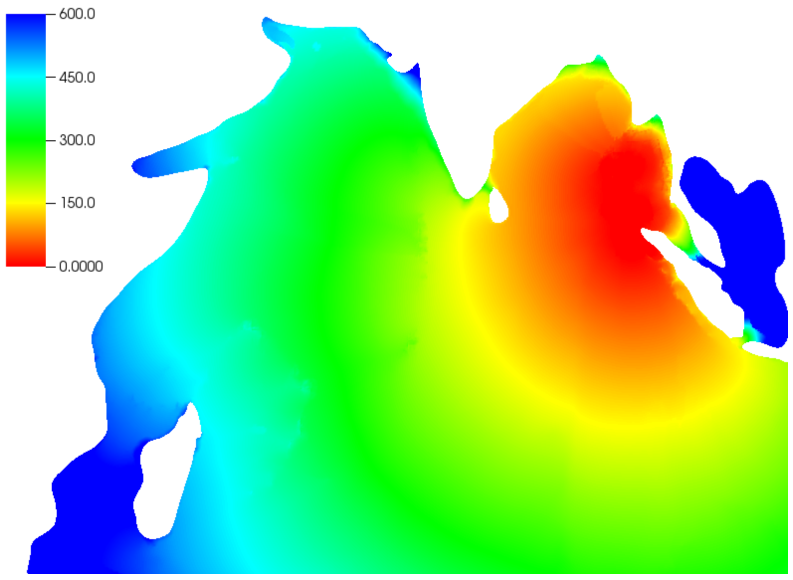


(b) $t = 160$, $N = 7$.

Figure 8.8.: ESDGSEM approximation of the free surface elevation during the Indian Ocean tsunami at $t = 160$min after the initial earthquake for polynomial orders $N = 3$ and $N = 7$. The scale for the free surface height is set to $[-0.4, 0.4]$m for comparison to previously published results by Gandham [75].

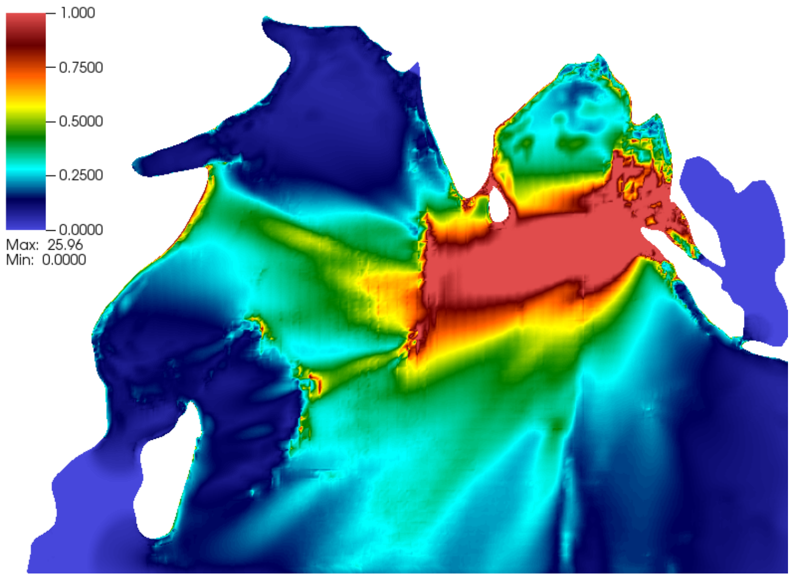


(a) $N = 3$.

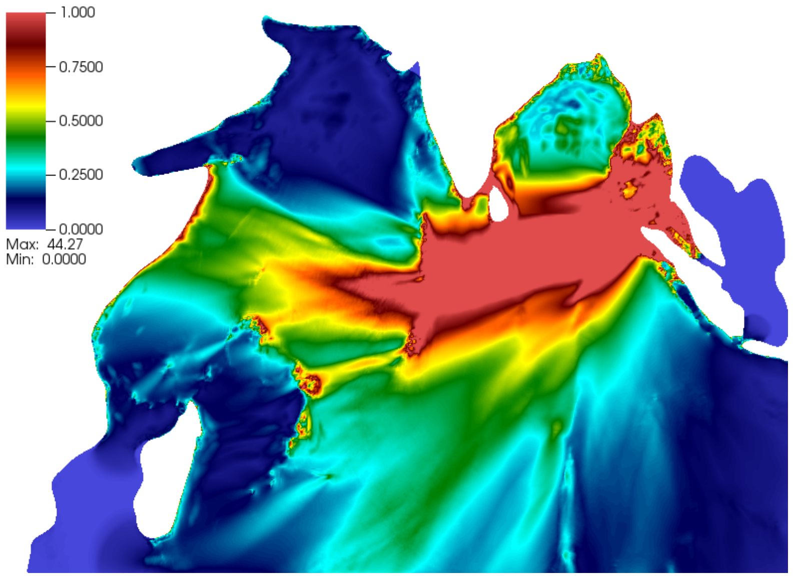


(b) $N = 7$.

Figure 8.9.: Arrival times in minutes over a ten hour simulation of the Indian Ocean tsunami.

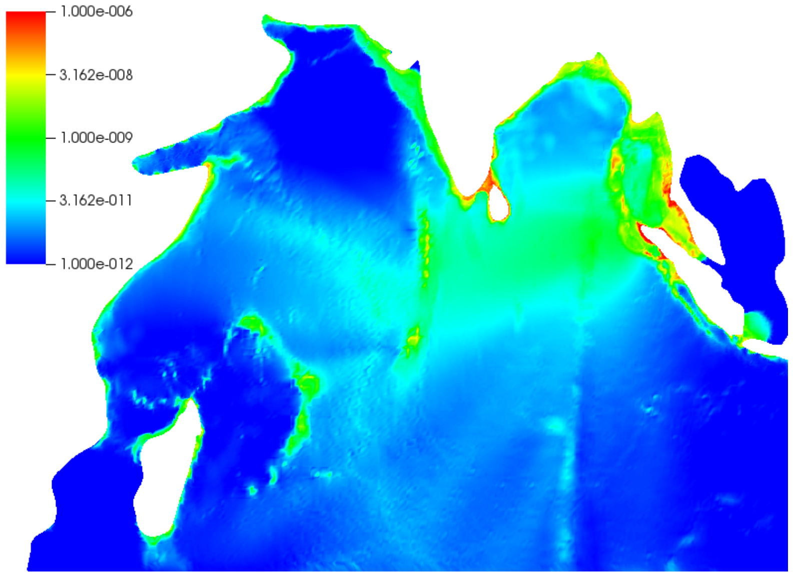


(a) $N = 3$.

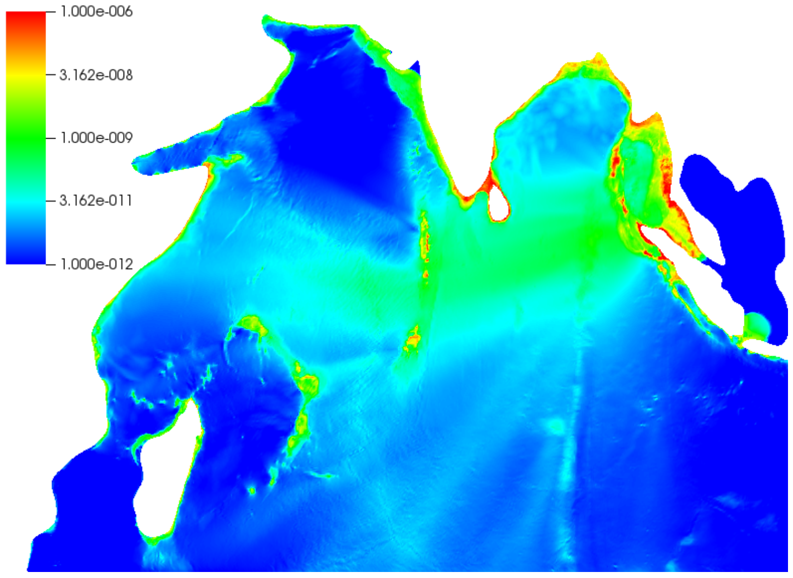


(b) $N = 7$

Figure 8.10.: Maximum surface elevations over the ten hour simulation approximated by the ESDGEM and for polynomial orders $N = 3$ and $N = 7$. The scale for the free surface height is set to $[0.0, 1.0]$m for comparison to previously published results by Gandham [75].

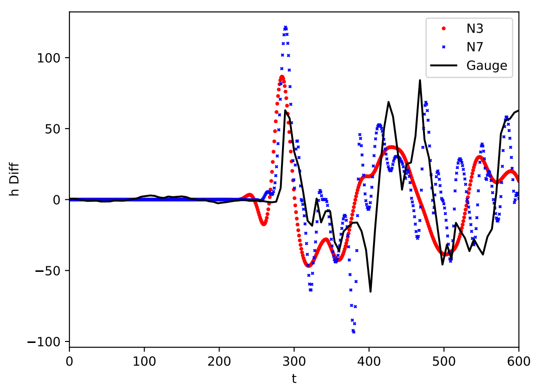


(a) Maximum Manning Friction $N = 3$.

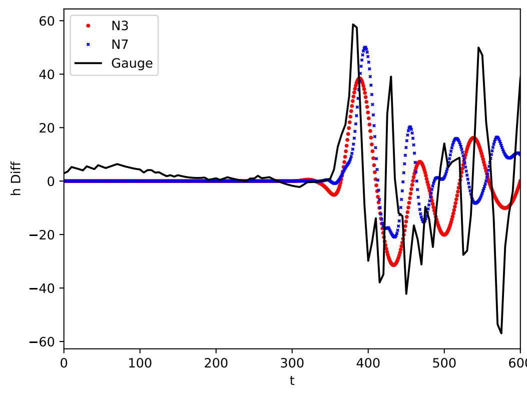


(b) Maximum Manning Friction $N = 7$.

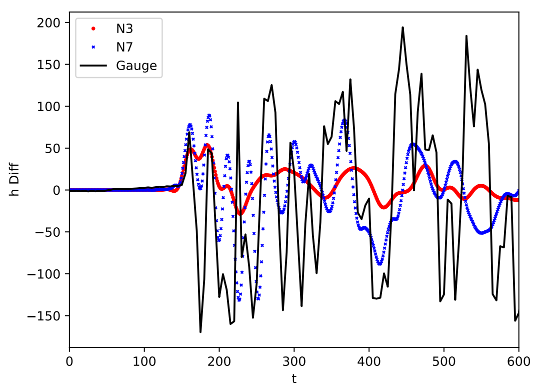
Figure 8.11.: Maximum Manning friction in m^2/s^2 over the ten hour simulation of the Indian Ocean Tsunami depicted on a logarithmic scale.



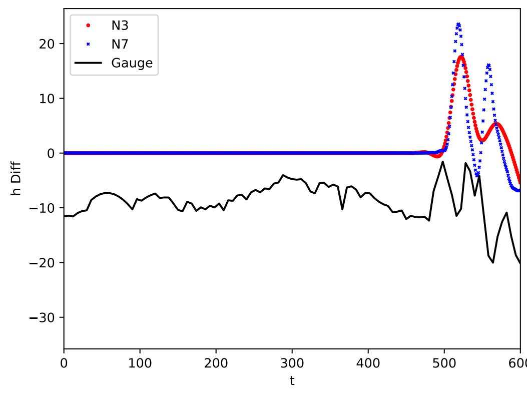
(a) Approximated arrival times in Kochi.



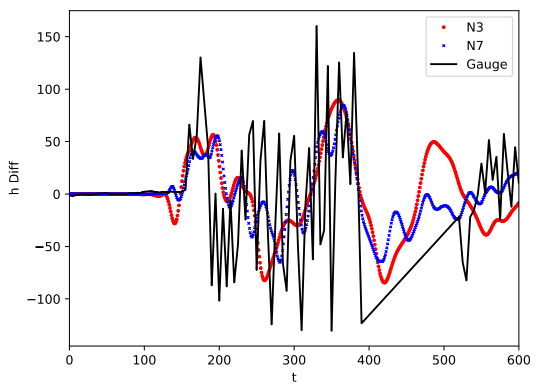
(b) Approximated arrival times in Mormugao.



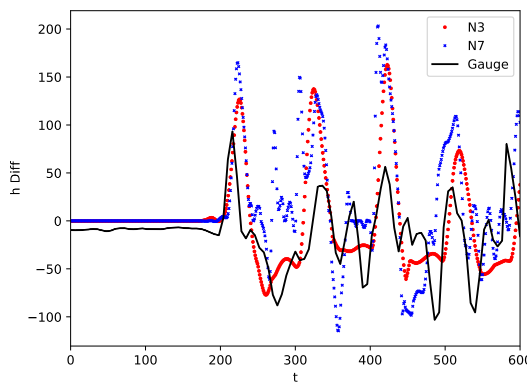
(c) Approximated arrival times in Chennai.



(d) Approximated arrival times in Okha.



(e) Approximated arrival times in Visakhapatnam.



(f) Approximated arrival times in Tuticorin.

Figure 8.12.: ESDGSEM approximation of the arrival times at different Gauge stations on the Indian coast and polynomial degrees $N = 3$ and $N = 7$. Measurements are presented for the free surface elevation level in *cm*.

9. Conclusion

The aim of this thesis was the development of a novel entropy stable discontinuous Galerkin method for the two dimensional shallow water equations. A necessary requirement for the scheme was also that it is well-balanced and applicable for unstructured curved quadrilateral meshes.

The general strategy was to use a special DG variant with SBP operators to discretize a specific split formulation of the shallow water equations. In combination with special entropy stable numerical interface fluxes and a consistent source term approximation, this led to an entropy stable method. Also, the method is capable of preserving the steady state solution of the “lake-at-rest” for all time and is, thus, well-balanced. These properties hold for arbitrary bottom topographies as well, as long as the mesh is constructed such that element interfaces align with any discontinuities.

The downside of this approach is, that a direct DG discretization of the split form shallow water equations on curved geometries leads to distinct volume integrals for all the split form terms. For each of the momentum equations alone there are forty different volume integrals. However, a remarkable equivalency found by Gassner et al. [82] allowed us to reformulate the split form discretization into the flux differencing framework of Carpenter and Fisher [64, 32]. This reformulation replaces the many distinct volume integrals by just one volume integral in each direction, albeit at the cost of evaluation a more expensive curvilinear numerical two-point flux.

This newly found method, the ESDGSEM, was extended with a shock capturing mechanism. We introduced artificial viscosity to the system and showed that for a specific choice of gradient variables and interface coupling according to Bassi and Rebay [8] maintains the entropy stability of the scheme. To limit the influence of the artificial viscosity to the shock regions we added a detector based on the works of Persson and Peraire [202], which has proven to accurately detect shock fronts and problematic regions in the solution. Furthermore, we added a positivity limiter as proposed by Zhang et al. [271]. In this context, the contribution of this thesis lies in the proof that the positivity limiter maintains non-negative water heights for the entropy stable numerical flux used in the ESDGSEM on curved meshes, under certain additional time step restrictions. We also verified that the positivity limiter adheres to the entropy stability principle.

The ESDGSEM naturally inherits the outstanding parallelizability of discontinuous Galerkin methods as no additional communication between elements is required. From an implementation standpoint, the only changes that must be made to an existing DGSEM implementation to obtain the ESDGSEM is to replace the volume integral by its flux differencing counterpart and use the entropy stable numerical flux on the interfaces. In the volume integral, this leads to an increased computational load for each nodal value due to the more costly evaluation of the curvilinear numerical two-point fluxes. The high computational complexity combined with the strong parallelizability made the ESDGSEM a perfect fit for modern GPU architectures.

Thus, we implemented the ESDGSEM on GPUs via the OCCA framework by Medina et al. [172]. Support for multiple GPUs is handled by MPI, connecting several CPU cores such that each acts as host for one GPU. We described implementation techniques that optimize the volume kernel for GPUs and compared the impact of each step on the kernel performance for both, the ESDGSEM and the standard DGSEM. Our analysis showed that both methods are memory bound for polynomial orders $N \leq 7$, at least on the GTX 1080 and the Tesla V100 we used for the computations. Thus, for these modern GPUs, there is virtually no price to pay for the entropy stable and well-balanced approximation provided by the ESDGSEM.

Finally, we used the newly developed ESDGSEM on GPUs to simulate the 2004 Indian Ocean tsunami. Although the mesh used in the computation was significantly smaller than the one used by Gandham [75], the results are similar. We obtained good approximations for the tsunami arrival times across six different tidal measurement stations on the Indian coast. We note that the simulation was quickly unstable (at $t < 1$ for the standard DGSEM if the same parameters were used otherwise and even for very low CFL numbers).

To develop global atmospheric models, the shallow water equations have been extended to the sphere [200]. While discontinuous Galerkin methods exists for the spherical model, e.g. [84, 85, 166, 145], it would be interesting to see if an entropy stable method can be developed following the strategies presented in this work. The first step would, of course, be to find a suitable entropy function that takes into the account the additional source terms which stem from the geometry as well as Coriolis forces.

The ESDGSEM itself could also benefit from further optimization. The added artificial viscosity presented in this work effectively mitigates the oscillations in the presence of shocks. However, it is not only computationally expensive but also may reduce the stable time step. It may be worthwhile to pursue other strategies to find an entropy stable shock capturing technique to increase the robustness of the scheme with less added computational cost.

Also, it would be useful to enhance the scheme such that it fulfills the extended well-balanced property. This includes steady state solutions that are combinations of the "lake-at-rest" and dry areas. The ESDGSEM is only capable of maintaining this steady state if the transition from wet to dry aligns with element interfaces. One strategy to approach this is presented by Bonev et al. [20], who replace the derivative operator for those elements with a special local derivative operator based on the wet/dry state of each node. While our experimentation with this operator worked well for the steady state solution, it proved less robust for some of the wet/dry test cases shown in this work. Also, since the new operator does not have the SBP property, it does not maintain the entropy stability of the scheme. Thus, further investigation into well-balanced and entropy stable schemes for partially flooded elements could prove beneficial.

Bibliography

[1] M. Ainsworth. Dispersive and dissipative behaviour of high order discontinuous Galerkin finite element methods. *Journal of Computational Physics*, 198:106–130, 2004.

[2] C. Amante and B.W. Eakins. ETOPO1 1 arc-minute global relief model: Procedures, data sources and analysis. NOAA Technical Memorandum NESDIS NGDC-24. National Geophysical Data Center, NOAA. doi:10.7289/V5C8276M, 2009. Accessed: 2018-10-22.

[3] Vijaya R Ambati and Onno Bokhove. Space–time discontinuous Galerkin discretization of rotating shallow water equations. *Journal of Computational Physics*, 225(2):1233–1261, 2007.

[4] Marco Ament, Gunter Knittel, Daniel Weiskopf, and Wolfgang Strasser. A parallel preconditioned conjugate gradient solver for the Poisson problem on a multi-GPU platform. In *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*, pages 583–592. IEEE, 2010.

[5] Athanasios Antoniou, Konstantinos Karantasis, Eleftherios Polychronopoulos, and John Ekaterinaris. Acceleration of a finite-difference WENO scheme for large-scale simulations on many-core architectures. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 525, 2010.

[6] Emmanuel Audusse, François Bouchut, Marie-Odile Bristeau, Rupert Klein, and Benoît Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM Journal of Scientitic Computing*, 25(6):2050–2065, 2004.

[7] Jan O Backhaus. A semi-implicit scheme for the shallow water equations for application to shelf sea modelling. *Continental Shelf Research*, 2(4):243–254, 1983.

[8] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 131:267–279, 1997.

[9] F. Bassi and S. Rebay. A high-order discontinuous Galerkin finite element method solution of the 2D Euler equations. *J. Comput. Phys.*, 138:251–285, 1997.

[10] Jurjen A Battjes and Robert Jan Labeur. *Unsteady flow in open channels.* Cambridge University Press, 2017.

[11] Nicole Beisiegel. *High-order Adaptive Discontinuous Galerkin Inundation Modeling*. PhD thesis, Universität Hamburg Hamburg, 2014.

[12] Nicole Beisiegel and Jörn Behrens. Quasi-nodal third-order Bernstein polynomials in a discontinuous Galerkin model for flooding and drying. *Environmental Earth Sciences*, 74(11):7275–7284, 2015.

[13] Jens Berg and Jan Nordström. Stable Robin solid wall boundary conditions for the Navier-Stokes equations. *Journal of Computational Physics*, 230(19):7519–7532, 2011.

[14] Alfredo Bermudez and Ma Elena Vázquez. Upwind methods for hyperbolic conservation laws with source terms. *Computers & Fluids*, 23(8):1049–1071, 1994.

[15] Paul-Emile Bernard, Nicolas Chevaugeon, Vincent Legat, Eric Deleersnijder, and Jean-François Remacle. High-order h-adaptive discontinuous Galerkin methods for ocean modelling. *Ocean Dynamics*, 57(2):109–121, 2007.

[16] G. A. Blaisdell, E. T. Spyropoulos, and J. H. Qin. The effect of the formulation of nonlinear terms on aliasing errors in spectral methods. *Applied Numerical Mathematics*, 21(3):207–219, 1996.

[17] Benjamin Block, Peter Virnau, and Tobias Preis. Multi-GPU accelerated multi-spin Monte Carlo simulations of the 2D Ising model. *Computer Physics Communications*, 181(9):1549–1556, 2010.

[18] Marvin Bohm, Andrew R Winters, Dominik Derigs, Gregor J Gassner, Stefanie Walch, and Joachim Saur. An entropy stable nodal discontinuous Galerkin method for the resistive MHD equations: Continuous analysis and GLM divergence cleaning. *arXiv preprint arXiv:1711.05576*, 2017.

[19] Andreas Bollermann, Sebastian Noelle, and Mária Lukáčová-Medvid'ová. Finite volume evolution Galerkin methods for the shallow water equations with dry beds. *Communications in Computational Physics*, 10(2):371–404, 2011.

[20] Boris Bonev, Jan S. Hesthaven, Francis X. Giraldo, and Michal A. Kopera. Discontinuous Galerkin scheme for the spherical shallow water equations with applications to tsunami modeling and prediction. Technical report, École Polytechnique Fédérale de Lausanne, 2017.

[21] Michael J Briggs, Costas E Synolakis, Gordon S Harkins, and Debra R Green. Laboratory experiments of tsunami runup on a circular island. In *Tsunamis: 1992–1994*, pages 569–593. Springer, 1995.

[22] Andre R Brodtkorb, Christopher Dyken, Trond R Hagen, Jon M Hjelmervik, and Olaf O Storaasli. State-of-the-art in heterogeneous computing. *Scientific Programming*, 18(1):1–33, 2010.

[23] André R Brodtkorb, Trond R Hagen, and Martin L Sætra. Graphics processing unit (GPU) programming strategies and trends in GPU computing. *Journal of Parallel and Distributed Computing*, 73(1):4–13, 2013.

[24] André R Brodtkorb, Martin L Sætra, and Mustafa Altinakar. Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. *Computers & Fluids*, 55:1–12, 2012.

[25] P Brufau, ME Vázquez-Cendón, and P García-Navarro. A numerical model for the flooding and drying of irregular domains. *International Journal for Numerical Methods in Fluids*, 39(3):247–275, 2002.

[26] Steve Bryson, Yekaterina Epshteyn, Alexander Kurganov, and Guergana Petrova. Well-balanced positivity preserving central-upwind scheme on triangular grids for the Saint-Venant system. *ESAIM: Mathematical Modelling and Numerical Analysis*, 45(3):423–446, 2011.

[27] Carsten Burstedde, Omar Ghattas, Michael Gurnis, Tobin Isaac, Georg Stadler, Tim Warburton, and Lucas Wilcox. Extreme-scale AMR. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE Computer Society, 2010.

[28] Valerio Caleffi, Alessandro Valiani, and Anna Bernini. Fourth-order balanced source term treatment in central WENO schemes for shallow water equations. *Journal of Computational Physics*, 218(1):228–245, 2006.

[29] Alberto Canestrelli, Annunziato Siviglia, Michael Dumbser, and Eleuterio F Toro. Well-balanced high-order centred schemes for non-conservative hyperbolic systems. applications to shallow water equations with fixed and mobile bed. *Advances in Water Resources*, 32(6):834–844, 2009.

[30] M. Carpenter, T. Fisher, E. Nielsen, and S. Frankel. Entropy stable spectral collocation schemes for the Navier–Stokes equations: Discontinuous interfaces. *SIAM Journal on Scientific Computing*, 36(5):B835–B867, 2014.

[31] M. Carpenter and C. Kennedy. Fourth-order $2N$-storage Runge-Kutta schemes. Technical Report NASA TM 109112, NASA Langley Research Center, 1994.

[32] Mark H Carpenter and Travis C Fisher. High-order entropy stable formulations for computational fluid dynamics. In *21st AIAA Computational Fluid Dynamics Conference*, page 2868, 2013.

[33] Mark H Carpenter and David Gottlieb. Spectral methods on arbitrary grids. Technical report, INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING HAMPTON VA, 1995.

[34] Mark H Carpenter, David Gottlieb, and Saul Abarbanel. Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes. *Journal of Computational Physics*, 111(2):220–236, 1994.

[35] Patrice Castonguay, Peter E Vincent, and Antony Jameson. A new class of high-order energy stable flux reconstruction schemes for triangular elements. *Journal of Scientific Computing*, 51(1):224–256, 2012.

[36] M.J. Castro, J.M. Gallardo, and C. Parés. High-order finite volume schemes based on reconstruction of states for solving hyperbolic systems with nonconservative products. Applications to shallow-water systems. *Math Comput*, 75:1103–1134, 2006.

[37] Ali Cevahir, Akira Nukada, and Satoshi Matsuoka. High performance conjugate gradient solver on multi-GPU clusters using hypergraph partitioning. *Computer Science-Research and Development*, 25(1-2):83–91, 2010.

[38] Jesse Chan. On discretely entropy conservative and entropy stable discontinuous Galerkin methods. *Journal of Computational Physics*, 362:346–374, 2018.

[39] Jesse Chan, Zheng Wang, Russell J Hewett, and T Warburton. Reduced storage nodal discontinuous Galerkin methods on semi-structured prismatic meshes. *Computers & Mathematics with Applications*, 73(5):775–793, 2017.

[40] Jesse Chan, Zheng Wang, Axel Modave, Jean-Francois Remacle, and Tim Warburton. GPU-accelerated discontinuous Galerkin methods on hybrid meshes. *Journal of Computational Physics*, 318:142–168, 2016.

[41] Jesse Chan and Tim Warburton. GPU-accelerated Bernstein–Bézier discontinuous Galerkin methods for wave problems. *SIAM Journal on Scientific Computing*, 39(2):A628–A654, 2017.

[42] Praveen Chandrashekar. Kinetic energy preserving and entropy stable finite volume schemes for compressible Euler and Navier-Stokes equations. *Communications in Computational Physics*, 14(5):1252–1286, 2013.

[43] Tianheng Chen and Chi-Wang Shu. Entropy stable high order discontinuous Galerkin methods with suitable quadrature rules for hyperbolic conservation laws. *Journal of Computational Physics*, 345:427–461, 2017.

[44] Yue Cheng, Fengyan Li, Jianxian Qiu, and Liwei Xu. Positivity-preserving DG and central DG methods for ideal MHD equations. *Journal of Computational Physics*, 238:255–280, 2013.

[45] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. *Discontinuous Galerkin Methods.* Lecture Notes in Computational Science and Engineering. Springer, 2000.

[46] Richard Courant, Eugene Isaacson, and Mina Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Communications on Pure and Applied Mathematics*, 5(3):243–255, 1952.

[47] Jared Crean, Jason E Hicken, David C Del Rey Fernández, David W Zingg, and Mark Carpenter. High-order, entropy-conservative discretizations of the Euler equations for complex geometries. In *23rd AIAA Computational Fluid Dynamics Conference*, page 4496, 2017.

[48] CSIR. CSIR - National Institute of Oceanography. `http://www.nio.org/index/option/com_nomenu/task/show/tid/2/sid/18/id/11`.

[49] Hossein Mahmoodi Darian and Vahid Esfahanian. Assessment of WENO schemes for multi-dimensional Euler equations using GPU. *International Journal for Numerical Methods in Fluids*, 76(12):961–981, 2014.

[50] Clint Dawson, Corey Jason Trahan, Ethan J Kubatko, and Joannes J Westerink. A parallel local timestepping Runge–Kutta discontinuous Galerkin method with applications to coastal ocean modeling. *Computer Methods in Applied Mechanics and Engineering*, 259:154–165, 2013.

[51] B De St Venant. Theorie du mouvement non-permanent des eaux avec application aux crues des rivers et a l'introduntion des marees dans leur lit. *Academic de Sci. Comptes Redus*, 73(99):148–154, 1871.

[52] Dominik Derigs, Gregor J Gassner, Stefanie Walch, and Andrew R Winters. Entropy stable finite volume approximations for ideal magnetohydrodynamics. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 120(3):153–219, 2018.

[53] Dominik Derigs, Andrew R Winters, Gregor J Gassner, Stefanie Walch, and Marvin Bohm. Ideal GLM-MHD : About the entropy consistent nine-wave magnetic field divergence diminishing ideal magnetohydrodynamics equations. *Journal of Computational Physics*, 364:420–467, 2018.

[54] Olivier Desjardins, Guillaume Blanquart, Guillaume Balarac, and Heinz Pitsch. High order conservative finite difference scheme for variable density low mach number turbulent flows. *Journal of Computational Physics*, 227(15):7125–7159, 2008.

[55] José M Domínguez, Alejandro JC Crespo, Daniel Valdez-Balderas, Benedict D Rogers, and Moncho Gómez-Gesteira. New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters. *Computer Physics Communications*, 184(8):1848–1860, 2013.

[56] F. Ducros, F. Laporte, T. Soulères, V. Guinot, P. Moinat, and B. Caruelle. High-order fluxes for conservative skew-symmetric-like schemes in structured meshes: Application to compressible flows. *Journal of Computational Physics*, 161:114–139, 2000.

[57] Bernd Einfeldt, Claus-Dieter Munz, Philip L Roe, and Björn Sjögreen. On Godunov-type methods near low densities. *Journal of Computational Physics*, 92(2):273–295, 1991.

[58] Vahid Esfahanian, Hossein Mahmoodi Darian, and SM Iman Gohari. Assessment of WENO schemes for numerical simulation of some hyperbolic equations using GPU. *Computers & Fluids*, 80:260–268, 2013.

[59] William John Feiereisen, William C Reynolds, and Joel H Ferziger. Numerical simulation of a compressible homogeneous, turbulent shear flow. 1981.

[60] David C Del Rey Fernández, Pieter D Boom, and David W Zingg. A generalized framework for nodal first derivative summation-by-parts operators. *Journal of Computational Physics*, 266:214–239, 2014.

[61] T. Fisher, M.H. Carpenter, J. Nordström, N. K. Yamaleev, and C. Swanson. Discretely conservative finite-difference formulations for nonlinear conservation laws in split form: Theory and boundary conditions. *Journal of Computational Physics.*, 234:pp. 353–375, 2013.

[62] T.C. Fisher, M.H. Carpenter, J. Nordström, N.K. Yamaleev, and R.C. Swanson. Discretely conservative finite-difference formulations for nonlinear conservation laws in split form: Theory and boundary conditions. Technical report, NASA/TM-2011-217307, 2011.

[63] Travis C. Fisher. *High-order L_2 stable multi-domain finite difference method for compressible flows*. PhD thesis, Purdue University, 2012.

[64] Travis C Fisher and Mark H Carpenter. High-order entropy stable finite difference schemes for nonlinear conservation laws: Finite domains. *Journal of Computational Physics*, 252:518–557, 2013.

[65] Travis C. Fisher and Mark H. Carpenter. High-order entropy stable finite difference schemes for nonlinear conservation laws: Finite domains. *Journal of Computational Physics*, 252:518–557, 2013.

[66] Travis C. Fisher, Mark H. Carpenter, Jan Nordström, Nail K. Yamaleev, and Charles Swanson. Discretely conservative finite-difference formulations for nonlinear conservation laws in split form: Theory and boundary conditions. *Journal of Computational Physics*, 234:353–375, 2013.

[67] Travis C. Fisher, Mark H. Carpenter, Nail K. Yamaleev, and Steven H. Frankel. Boundary closures for fourth-order energy stable weighted essentially non-oscillatory finite-difference schemes. *Journal of Computational Physics*, 230(10):3727 – 3752, 2011.

[68] Ulrik S. Fjordholm, Siddhartha Mishra, and Eitan Tadmor. Well-balanced and energy stable schemes for the shallow water equations with discontinuous topography. *Journal of Computational Physics*, 230(14):5587–5609, 2011.

[69] Ulrik S Fjordholm, Siddhartha Mishra, and Eitan Tadmor. Arbitrarily high-order accurate entropy stable essentially nonoscillatory schemes for systems of conservation laws. *SIAM Journal on Numerical Analysis*, 50(2):544–573, 2012.

[70] Ulrik Skre Fjordholm. *High-order accurate entropy stable numercial schemes for hyperbolic conservation laws.* ETH Zurich, 2013.

[71] Lucas Friedrich, Gero Schnücke, Andrew R Winters, David C Fernández, Gregor J Gassner, and Mark H Carpenter. Entropy stable space-time discontinuous Galerkin schemes with summation-by-parts property for hyperbolic conservation laws. *arXiv preprint arXiv:1808.08218*, 2018.

[72] Martin Fuhry, Andrew Giuliani, and Lilia Krivodonova. Discontinuous Galerkin methods on graphics processing units for nonlinear hyperbolic conservation laws. *International Journal for Numerical Methods in Fluids*, 76(12):982–1003, 2014.

[73] Daniele Funaro and David Gottlieb. A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations. *Mathematics of computation*, 51(184):599–613, 1988.

[74] José M Gallardo, Carlos Parés, and Manuel Castro. On a well-balanced high-order finite volume scheme for shallow water equations with topography and dry areas. *Journal of Computational Physics*, 227(1):574–601, 2007.

[75] Rajesh Gandham. *High performance high-order numerical methods: Applications in ocean modeling.* PhD thesis, Rice University, 2015.

[76] Rajesh Gandham, David Medina, and Timothy Warburton. GPU accelerated discontinuous Galerkin methods for shallow water equations. *Communications in Computational Physics*, 18(1):37–64, 2015.

[77] Michael Garland, Scott Le Grand, John Nickolls, Joshua Anderson, Jim Hardwick, Scott Morton, Everett Phillips, Yao Zhang, and Vasily Volkov. Parallel computing experiences with CUDA. *IEEE micro*, (4):13–27, 2008.

[78] G. Gassner. A skew-symmetric discontinuous Galerkin spectral element discretization and its relation to SBP-SAT finite difference methods. *SIAM Journal on Scientific Computing*, 35(3):A1233–A1253, 2013.

[79] G. Gassner, F. Lörcher, and C.-D. Munz. A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes. *J. Comput. Phys.*, 224(2):1049–1063, 2007.

[80] Gregor J. Gassner. A kinetic energy preserving nodal discontinuous Galerkin spectral element method. *International Journal for Numerical Methods in Fluids*, 76(1):28–50, 2014.

[81] Gregor J. Gassner, Andrew R. Winters, Florian J. Hindenlang, and David A. Kopriva. The BR1 scheme is stable for the compressible Navier-Stokes equations. *Journal of Scientific Computing (accepted manuscript)*, 2017.

[82] Gregor J. Gassner, Andrew R. Winters, and David A. Kopriva. Split form nodal discontinuous Galerkin schemes with summation-by-parts property for the compressible Euler equations. *Journal of Computational Physics*, 327:39–66, 2016.

[83] Gregor J Gassner, Andrew R Winters, and David A Kopriva. A well balanced and entropy conservative discontinuous Galerkin spectral element method for the shallow water equations. *Applied Mathematics and Computation*, 272:291–308, 2016.

[84] Francis X Giraldo, Jan S Hesthaven, and Tim Warburton. Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations. *Journal of Computational Physics*, 181(2):499–525, 2002.

[85] Francis X Giraldo and Tim Warburton. A high-order triangular discontinuous Galerkin oceanic shallow water model. *International journal for numerical methods in fluids*, 56(7):899–925, 2008.

[86] Dominik Göddeke, Robert Strzodka, and Stefan Turek. *Accelerating double precision FEM simulations with GPUs.*

[87] Nico Godel, Nigel Nunn, Tim Warburton, and Markus Clemens. Scalability of higher-order discontinuous Galerkin FEM computations for solving electromagnetic wave propagation problems on GPU clusters. *IEEE Transactions on Magnetics*, 46(8):3469–3472, 2010.

[88] Nico Godel, Steffen Schomann, Tim Warburton, and Markus Clemens. GPU accelerated Adams–Bashforth multirate discontinuous Galerkin FEM simulation of high-frequency electromagnetic fields. *IEEE Transactions on magnetics*, 46(8):2735–2738, 2010.

[89] Sergei Konstantinovich Godunov. An interesting class of quasilinear systems. In *Dokl. Acad. Nauk SSSR*, volume 139, pages 521–523, 1961.

[90] N Goedel, T Warburton, and M Clemens. GPU accelerated discontinuous Galerkin FEM for electromagnetic radio frequency problems. In *Antennas and Propagation Society International Symposium, 2009. APSURSI'09. IEEE*, pages 1–4. IEEE, 2009.

[91] L Gosse. A well-balanced flux-vector splitting scheme designed for hyperbolic systems of conservation laws with source terms. *Computers & Mathematics with Applications*, 39(9-10):135–159, 2000.

[92] David Gottlieb and Chi-Wang Shu. On the Gibbs phenomenon and its resolution. *SIAM review*, 39(4):644–668, 1997.

[93] Olivier Gourgue, Richard Comblen, Jonathan Lambrechts, Tuomas Kärnä, Vincent Legat, and Eric Deleersnijder. A flux-limiting wetting–drying method for finite-element shallow-water models, with application to the Scheldt Estuary. *Advances in Water Resources*, 32(12):1726–1739, 2009.

[94] JM Greenberg, AY Leroux, R Baraille, and A Noussair. Analysis and approximation of conservation laws with source terms. *SIAM Journal on Numerical Analysis*, 34(5):1980–2007, 1997.

[95] Joshua M Greenberg and Alain-Yves LeRoux. A well-balanced scheme for the numerical processing of source terms in hyperbolic equations. *SIAM Journal on Numerical Analysis*, 33(1):1–16, 1996.

[96] Michael Griebel and Peter Zaspel. A multi-GPU accelerated solver for the three-dimensional two-phase incompressible Navier-Stokes equations. *Computer Science-Research and Development*, 25(1-2):65–73, 2010.

[97] Stéphan T Grilli, Mansour Ioualalen, Jack Asavanant, Fengyan Shi, James T Kirby, and Philip Watts. Source constraints and model simulation of the december 26, 2004, Indian Ocean tsunami. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 133(6):414–428, 2007.

[98] Bertil Gustafsson. The convergence rate for difference approximations to mixed initial boundary value problems. *Mathematics of Computation*, 29(130):396–406, 1975.

[99] Bertil Gustafsson. *High order difference methods for time dependent PDE*, volume 38. Springer Science & Business Media, 2007.

[100] Trond Runar Hagen, Knut-Andreas Lie, and Jostein R Natvig. Solving the Euler equations on graphics processing units. In *International Conference on Computational Science*, pages 220–227. Springer, 2006.

[101] Luc Hamm, Per A Madsen, and D Howell Peregrine. Wave transformation in the nearshore zone: a review. *Coastal Engineering*, 21(1-3):5–39, 1993.

[102] Ami Harten and Stanley Osher. Uniformly high-order accurate nonoscillatory schemes. i. *SIAM Journal on Numerical Analysis*, 24(2):279–309, 1987.

[103] Amiram Harten. On the symmetric form of systems of conservation laws with entropy. *Journal of computational physics*, 49:151–164, 1983.

[104] Amiram Harten, James M Hyman, Peter D Lax, and Barbara Keyfitz. On finite-difference approximations and entropy conditions for shocks. *Communications on pure and applied mathematics*, 29(3):297–322, 1976.

[105] Amiram Harten, Peter D Lax, and Bram van Leer. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM review*, 25(1):35–61, 1983.

[106] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications.* Springer Verlag, New York, 2008.

[107] Jan S. Hesthaven. *Numerical methods for Conservation laws: From Analysis to Algorithms.* SIAM Publishing, Philadelphia, 2018.

[108] Jason E Hicken, David C Del Rey Fernández, and David W Zingg. Multidimensional summation-by-parts operators: General theory and application to simplex elements. *SIAM Journal on Scientific Computing*, 38(4):A1935–A1958, 2016.

[109] Jason E Hicken and David W Zingg. Parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms. *AIAA journal*, 46(11):2773–2786, 2008.

[110] Jason E Hicken and David W Zingg. Superconvergent functional estimates from summation-by-parts finite-difference discretizations. *SIAM Journal on Scientific Computing*, 33(2):893–922, 2011.

[111] Jason E Hicken and David W Zingg. Summation-by-parts operators and high-order quadrature. *Journal of Computational and Applied Mathematics*, 237(1):111–125, 2013.

[112] Andreas Hiltebrand and Siddhartha Mishra. Entropy stability and well-balancedness of space-time DG for the shallow water equations with bottom topography. Technical Report 2015-13, ETH Zürich, 2015.

[113] Leo H Holthuijsen. *Waves in oceanic and coastal waters.* Cambridge university press, 2010.

[114] Albert E Honein and Parviz Moin. Higher entropy conservation and numerical stability of compressible turbulence simulations. *Journal of Computational Physics*, 201(2):531–545, 2004.

[115] H. T. Huynh. A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods. In *18th AIAA Computational Fluid Dynamics Conference*, AIAA 2007-4079, June 25-28 2007.

[116] M Ioualalen, J Asavanant, N Kaewbanjak, ST Grilli, JT Kirby, and P Watts. Modeling the 26 december 2004 Indian Ocean tsunami: Case study of impact in Thailand. *Journal of Geophysical Research: Oceans*, 112(C7), 2007.

[117] Farzad Ismail and Philip L. Roe. Affordable, entropy-consistent Euler flux functions II: Entropy production at shocks. *Journal of Computational Physics*, 228(15):5410–5436, 2009.

[118] Dana Jacobsen, Julien Thibault, and Inanc Senocak. An MPI-CUDA implementation for massively parallel incompressible flow computations on multi-GPU clusters. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 522, 2010.

[119] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *Journal of computational physics*, 126(1):202–228, 1996.

[120] Shi Jin. A steady-state capturing method for hyperbolic systems with geometrical source terms. *ESAIM: Mathematical Modelling and Numerical Analysis*, 35(4):631–645, 2001.

[121] Ali Karakus, Noel Chalmers, Kasia Swirydowicz, and Timothy Warburton. GPU acceleration of a high-order discontinuous Galerkin incompressible flow solver. *arXiv preprint arXiv:1801.00246*, 2017.

[122] Konstantinos I Karantasis, Eleftherios D Polychronopoulos, and John A Ekaterinaris. High order accurate simulation of compressible flows on GPU clusters over software distributed shared memory. *Computers & Fluids*, 93:18–29, 2014.

[123] G. E. Karniadakis and S. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, USA, 2005.

[124] Christopher A. Kennedy and Andrea Gruber. Reduced aliasing formulations of the convective terms within the Navier–Stokes equations for a compressible fluid. *Journal of Computational Physics*, 227:1676–1700, 2008.

[125] Georges Kesserwani and Qiuhua Liang. A discontinuous Galerkin algorithm for the two-dimensional shallow water equations. *Computer Methods in Applied Mechanics and Engineering*, 199(49-52):3356–3368, 2010.

[126] Andreas Klöckner. Loo. py: transformation-based code generation for GPUs and CPUs. In *Proceedings of ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming*, page 82. ACM, 2014.

[127] Andreas Klöckner. Loo. py: From FORTRAN to performance via transformation and substitution rules. In *Proceedings of the 2nd ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming*, pages 1–6. ACM, 2015.

[128] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, Bryan Catanzaro, Paul Ivanov, and Ahmed Fasih. PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. *Parallel Computing*, 38(3):157–174, 2012.

[129] Andreas Klöckner, Tim Warburton, Jeff Bridge, and Jan S Hesthaven. Nodal discontinuous Galerkin methods on graphics processors. *Journal of Computational Physics*, 228(21):7863–7882, 2009.

[130] Andreas Klöckner, Tim Warburton, and Jan S Hesthaven. Viscous shock capturing in a time-explicit discontinuous Galerkin method. *Mathematical Modelling of Natural Phenomena*, 6(3):57–83, 2011.

[131] Andreas Klöckner, Timothy Warburton, and Jan S Hesthaven. Solving wave equations on unstructured geometries. *GPU Computing Gems*, 2:225, 2012.

[132] Andreas Klöckner, Timothy Warburton, and Jan S Hesthaven. High-order discontinuous Galerkin methods by GPU metaprogramming. In *GPU Solutions to Multi-scale Problems in Science and Engineering*, pages 353–374. Springer, 2013.

[133] Andreas Klöckner, Lucas C Wilcox, and Tim Warburton. Array program transformation with loo. py by example: High-order finite elements. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming*, pages 9–16. ACM, 2016.

[134] JC Kok. A high-order low-dispersion symmetry-preserving finite-volume method for compressible flow on curvilinear grids. *Journal of Computational Physics*, 228(18):6811–6832, 2009.

[135] D. A. Kopriva. Metric identities and the discontinuous spectral element method on curvilinear meshes. *Journal of Scientific Computing*, 26(3):301–327, March 2006.

[136] D.A. Kopriva. SpecMesh. Private communication.

[137] David A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers.* Springer Publishing Company, Incorporated, 1st edition, 2009.

[138] David A. Kopriva and Gregor Gassner. On the quadrature and weak form choices in collocation type discontinuous Galerkin spectral element methods. *Journal of Scientific Computing*, 44(2):136–155, 2010-08-01.

[139] Sean E Krakiwsky, Laurence E Turner, and Michal M Okoniewski. Acceleration of finite-difference time-domain (fdtd) using graphics processor units (GPU). In *Microwave Symposium Digest, 2004 IEEE MTT-S International*, volume 2, pages 1033–1036. IEEE, 2004.

[140] H.-O. Kreiss and J. Olliger. Comparison of accurate methods for the integration of hyperbolic equations. *Tellus*, 24:199–215, 1972.

[141] H-O Kreiss and Godela Scherer. Finite element and finite difference methods for hyperbolic partial differential equations. In *Mathematical aspects of finite elements in partial differential equations*, pages 195–212. Elsevier, 1974.

[142] Harish Kumar and Siddhartha Mishra. Entropy stable numerical schemes for two-fluid plasma equations. *Journal of scientific computing*, 52(2):401–425, 2012.

[143] Alexander Kurganov and Doron Levy. Central-upwind schemes for the Saint-Venant system. *ESAIM: Mathematical Modelling and Numerical Analysis*, 36(3):397–425, 2002.

[144] Oak Ridge National Laboratory. Titan configuration. `https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan`. Accessed: 2018-08-30.

[145] Matthias Läuter, Francis X Giraldo, Dörthe Handorf, and Klaus Dethloff. A discontinuous Galerkin method for the shallow water equations in spherical triangular coordinates. *Journal of Computational Physics*, 227(24):10226–10242, 2008.

[146] Peter Lax. Shock waves and entropy. In *Contributions to nonlinear functional analysis*, pages 603–634. Elsevier, 1971.

[147] Peter Lax and Burton Wendroff. Systems of conservation laws. *Communications on Pure and Applied mathematics*, 13(2):217–237, 1960.

[148] Peter D Lax. Hyperbolic systems of conservation laws ii. *Communications on pure and applied mathematics*, 10(4):537–566, 1957.

[149] Peter D Lax. *Hyperbolic systems of conservation laws and the mathematical theory of shock waves*, volume 11. SIAM, 1973.

[150] Peter D Lax. *Hyperbolic partial differential equations*, volume 14. American Mathematical Soc., 2006.

[151] Peter D Lax and Robert D Richtmyer. Survey of the stability of linear finite difference equations. *Communications on pure and applied mathematics*, 9(2):267–293, 1956.

[152] P. LeFloch and C. Rohde. High-order schemes, entropy inequalities, and nonclassical shocks. *SIAM Journal on Numerical Analysis*, 37(6):2023–2060, 2000.

[153] Philippe G LeFloch. *Hyperbolic Systems of Conservation Laws: The theory of classical and nonclassical shock waves.* Springer Science & Business Media, 2002.

[154] Philippe G LeFloch, Jean-Marc Mercier, and Christian Rohde. Fully discrete, entropy conservative schemes of arbitrary order. *SIAM Journal on Numerical Analysis*, 40(5):1968–1992, 2002.

[155] Randall J LeVeque. Numerical methods for conservation laws. 1992. *Birkhäuser Basel.*

[156] Randall J. LeVeque. Balancing source terms and flux gradients in high-resolution Godunov methods: the quasi-steady wave-propagation algorithm. *Journal of Computational Physics*, 146(1):346–365, 1998.

[157] Gang Li, Valerio Caleffi, and Zhengkun Qi. A well-balanced finite difference WENO scheme for shallow water flow model. *Applied Mathematics and Computation*, 265:1–16, 2015.

[158] Wei Li, Xiaoming Wei, and Arie Kaufman. Implementing lattice Boltzmann computation on graphics hardware. *The Visual Computer*, 19(7-8):444–456, 2003.

[159] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially nonoscillatory schemes. *Journal of Computational Physics*, 115:200–212, 1994.

[160] Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. *Journal of computational physics*, 115(1):200–212, 1994.

[161] Yong Liu, Chi-Wang Shu, and Mengping Zhang. Entropy stable high order discontinuous Galerkin methods for ideal compressible MHD on structured meshes. *Journal of Computational Physics*, 354:163–178, 2018.

[162] F. Lörcher, G. Gassner, and C.-D. Munz. An explicit discontinuous Galerkin scheme with local time-stepping for general unsteady diffusion equations. *J. Comput. Phys.*, 227(11):5649–5670, 2008.

[163] Mária Lukácová-Medvid'ová and Eitan Tadmor. On the entropy stability of the Roe-type finite volume methods. In *Proceedings of the twelfth international conference on hyperbolic problems, American Mathematical Society, eds. J.-G. Liu et al*, volume 67, 2009.

[164] Maria Lukáčová-Medvid'ová, Sebastian Noelle, and Marcus Kraft. Well-balanced finite volume evolution Galerkin methods for the shallow water equations. *Journal of computational physics*, 221(1):122–147, 2007.

[165] Tomas Lundquist and Jan Nordström. The SBP-SAT technique for initial value problems. *Journal of Computational Physics*, 270:86–104, 2014.

[166] S Marras, Michal A Kopera, and Francis X Giraldo. Simulation of shallow-water jets with a unified element-based continuous/discontinuous Galerkin model with grid flexibility on the sphere. *Quarterly Journal of the Royal Meteorological Society*, 141(690):1727–1739, 2015.

[167] Simone Marras, Michal A Kopera, Emil M Constantinescu, Jenny Suckale, and Francis X Giraldo. A residual-based shock capturing scheme for the continuous/discontinuous spectral element solution of the 2D shallow water equations. *Advances in Water Resources*, 114:45–63, 2018.

[168] Ken Mattsson, Magnus Svärd, and Jan Nordström. Stable and accurate artificial dissipation. *Journal of Scientific Computing*, 21(1):57–79, 2004.

[169] Ken Mattsson, Magnus Svärd, and Mohammad Shoeybi. Stable and accurate schemes for the compressible Navier–Stokes equations. *Journal of Computational Physics*, 227(4):2293–2316, 2008.

[170] David Medina. *OKL: A unified language for parallel architectures.* PhD thesis, Rice University, 2015.

[171] David Medina, Amik St-Cyr, and Timothy Warburton. High-order finite-differences on multi-threaded architectures using OCCA. In *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014*, pages 365–373. Springer, 2015.

[172] David S Medina, Amik St-Cyr, and Timothy Warburton. OCCA: A unified approach to multi-threading languages. *arXiv preprint arXiv:1403.0968*, 2014.

[173] Marshal L. Merriam. An entropy-based approach to nonlinear stability. *NASA Technical Memorandum*, 101086(64):1–154, 1989.

[174] Michael S Mock. Systems of conservation laws of mixed type. *Journal of Differential equations*, 37(1):70–88, 1980.

[175] Axel Modave, Andreas Atle, Jesse Chan, and Tim Warburton. A GPU-accelerated nodal discontinuous Galerkin method with high-order absorbing boundary conditions and corner/edge compatibility. *International Journal for Numerical Methods in Engineering*, 112(11):1659–1686, 2017.

[176] Axel Modave, Amik St-Cyr, Wim A Mulder, and Tim Warburton. A nodal discontinuous Galerkin method for reverse-time migration on GPU clusters. *Geophysical Journal International*, 203(2):1419–1435, 2015.

[177] Axel Modave, Amik St-Cyr, and Tim Warburton. GPU performance analysis of a nodal discontinuous Galerkin method for acoustic and elastic models. *Computers & Geosciences*, 91:64–76, 2016.

[178] Cathleen S Morawetz. On the non-existence of continuous transonic flows past profiles i. *Communications on Pure and Applied Mathematics*, 9(1):45–68, 1956.

[179] John Nickolls and William J Dally. The GPU computing era. *IEEE micro*, 30(2), 2010.

[180] Sebastian Noelle, Normann Pankratz, Gabriella Puppo, and Jostein R Natvig. Well-balanced finite volume schemes of arbitrary order of accuracy for shallow water flows. *Journal of Computational Physics*, 213(2):474–499, 2006.

[181] Sebastian Noelle, Yulong Xing, and Chi-Wang Shu. High-order well-balanced finite volume WENO schemes for shallow water equation with moving water. *Journal of Computational Physics*, 226(1):29–58, 2007.

[182] Jan Nordström and Martin Björck. Finite volume approximations and strict stability for hyperbolic problems. *Applied numerical mathematics*, 38(3):237–255, 2001.

[183] Jan Nordström, Karl Forsberg, Carl Adamsson, and Peter Eliasson. Finite volume methods, unstructured meshes and strict stability for hyperbolic problems. *Applied Numerical Mathematics*, 45(4):453–473, 2003.

[184] Jan Nordström, Jing Gong, Edwin Van der Weide, and Magnus Svärd. A stable and conservative high order multi-block method for the compressible Navier–Stokes equations. *Journal of Computational Physics*, 228(24):9020–9035, 2009.

[185] Jan Nordström and Cristina La Cognata. Energy stable boundary conditions for the nonlinear incompressible Navier–Stokes equations. *Mathematics of Computation*, 2018.

[186] Jan Nordström and Tomas Lundquist. Summation-by-parts in time. *Journal of Computational Physics*, 251:487–499, 2013.

[187] Jan Nordström and Tomas Lundquist. Summation-by-parts in time: the second derivative. *SIAM Journal on Scientific Computing*, 38(3):A1561–A1586, 2016.

[188] CUDA Nvidia. Programming guide, 2010.

[189] Christian Obrecht, Frédéric Kuznik, Bernard Tourancheau, and Jean-Jacques Roux. Multi-GPU implementation of the lattice Boltzmann method. *Computers & Mathematics with Applications*, 65(2):252–261, 2013.

[190] The Editors of Encyclopaedia Britannica. Indian Ocean tsunami of 2004. `https://www.britannica.com/event/Indian-Ocean-tsunami-of-2004`, 2018. Accessed: 2018-11-1.

[191] Yoshimitsu Okada. Internal deformation due to shear and tensile faults in a half-space. *Bulletin of the Seismological Society of America*, 82(2):1018–1040, 1992.

[192] Stanley Osher. Riemann solvers, the entropy condition, and difference. *SIAM Journal on Numerical Analysis*, 21(2):217–235, 1984.

[193] Stanley Osher and Eitan Tadmor. On the convergence of difference approximations to scalar conservation laws. *Mathematics of Computation*, 50(181):19–51, 1988.

[194] Michal Osusky and David W Zingg. Parallel Newton–Krylov–Schur flow solver for the Navier–Stokes equations. *AIAA journal*, 51(12):2833–2851, 2013.

[195] Carlos Parés. Numerical methods for nonconservative hyperbolic systems: A theoretical framework. *SIAM Journal on Numerical Analysis*, 44(1):300–321, 2006.

[196] Carlos Parés and Manuel Castro. On the well-balance property of Roe's method for nonconservative hyperbolic systems. Applications to shallow-water systems. *ESAIM: mathematical modelling and numerical analysis*, 38(5):821–852, 2004.

[197] Raphaël Paris, Franck Lavigne, Patrick Wassmer, and Junun Sartohadi. Coastal sedimentation associated with the December 26, 2004 tsunami in Lhok Nga, west Banda Aceh (Sumatra, Indonesia). *Marine Geology*, 238(1-4):93–106, 2007.

[198] Raphaël Paris, Patrick Wassmer, Junun Sartohadi, Franck Lavigne, Benjamin Barthomeuf, Emilie Desgages, Delphine Grancher, Philippe Baumert, Franck Vautier, Daniel Brunstein, et al. Tsunamis as geomorphic crises: Lessons from the December 26, 2004 tsunami in Lhok Nga, West Banda Aceh (Sumatra, Indonesia). *Geomorphology*, 104(1-2):59–72, 2009.

[199] Matteo Parsani, Mark H Carpenter, and Eric J Nielsen. Entropy stable wall boundary conditions for the three-dimensional compressible Navier–Stokes equations. *Journal of Computational Physics*, 292:88–113, 2015.

[200] Joseph Pedlosky. *Geophysical fluid dynamics*. Springer Science & Business Media, 2013.

[201] Donald B Percival, Donald W Denbo, Marie C Eblé, Edison Gica, Harold O Mofjeld, Michael C Spillane, Liujuan Tang, and Vasily V Titov. Extraction of tsunami source coefficients via inversion of dart buoy data. *Natural hazards*, 58(1):567–590, 2011.

[202] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. *AIAA Journal*, 112, 2006.

[203] Benoıt Perthame. Second-order Boltzmann schemes for compressible Euler equations in one and two space dimensions. *SIAM Journal on Numerical Analysis*, 29(1):1–19, 1992.

[204] Benoıt Perthame and Chi-Wang Shu. On positivity preserving finite volume schemes for Euler equations. *Numerische Mathematik*, 73(1):119–130, 1996.

[205] Dang Hieu Phung. Numerical study of long wave runup on a conical island. *VNU Journal of Science, Earth Sciences*, 24:79–86, 2008.

[206] Hendrik Ranocha. Shallow water equations: Split-form, entropy stable, well-balanced, and positivity preserving numerical methods. *GEM-International Journal on Geomathematics*, 8(1):85–133, 2017.

[207] Hendrik Ranocha, Philipp Öffner, and Thomas Sonar. Summation-by-parts operators for correction procedure via reconstruction. *Journal of Computational Physics*, 311:299–328, 2016.

[208] Deep Ray, Praveen Chandrashekar, Ulrik S Fjordholm, and Siddhartha Mishra. Entropy stable scheme on two-dimensional unstructured grids for Euler equations. *Communications in Computational Physics*, 19(5):1111–1140, 2016.

[209] W.H. Reed and T.R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[210] P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 135(2):250–258, 1997.

[211] Benedict D Rogers, Alistair GL Borthwick, and Paul H Taylor. Mathematical balancing of flux gradient and source terms prior to using Roe's approximate Riemann solver. *Journal of Computational Physics*, 192(2):422–451, 2003.

[212] Marie Rousseau, Olivier Cerdan, Olivier Delestre, Fabrice Dupros, Francois James, and Stéphane Cordier. Overland flow modelling with the shallow water equation using a well balanced numerical scheme: Adding efficiency or just more complexity? 2012.

[213] Viktor Vladimirovich Rusanov. The calculation of the interaction of non-stationary shock waves with barriers. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 1(2):267–279, 1961.

[214] Martin L Sætra, André R Brodtkorb, and Knut-Andreas Lie. Efficient GPU-implementation of adaptive mesh refinement for the shallow-water equations. *Journal of Scientific Computing*, 63(1):23–48, 2015.

[215] Martin Lilleeng Sætra and André Rigland Brodtkorb. Shallow water simulations on multiple GPUs. In *International Workshop on Applied Parallel Computing*, pages 56–66. Springer, 2010.

[216] Francesco Salvadore, Matteo Bernardini, and Michela Botti. GPU accelerated flow solver for direct numerical simulation of turbulent flows. *Journal of Computational Physics*, 235:129–142, 2013.

[217] Chi-Wang Shu. High-order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD. *International Journal of Computational Fluid Dynamics*, 17(2):107–118, 2003.

[218] Chi-Wang Shu. High order weighted essentially nonoscillatory schemes for convection dominated problems. *SIAM review*, 51(1):82–126, 2009.

[219] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988.

[220] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes, II. *Journal of Computational Physics*, 83(1):32–78, 1989.

[221] Martin Siebenborn and Volker Schulz. GPU accelerated discontinuous Galerkin methods for Euler equations and its adjoint. In *Proceedings of the High Performance Computing Symposium*, page 3. Society for Computer Simulation International, 2013.

[222] Martin Siebenborn, Volker Schulz, and Stephan Schmidt. A curved-element unstructured discontinuous Galerkin method on GPUs for the Euler equations. *Computing and Visualization in Science*, 15(2):61–73, 2012.

[223] Joel Smoller. *Shock waves and reaction—diffusion equations*, volume 258. Springer Science & Business Media, 2012.

[224] John C Strikwerda. *Finite difference schemes and partial differential equations*, volume 88. Siam, 2004.

[225] Magnus Svärd. On coordinate transformations for summation-by-parts operators. *Journal of Scientific Computing*, 20(1):29–42, 2004.

[226] Magnus Svärd, Mark H Carpenter, and Jan Nordström. A stable high-order finite difference scheme for the compressible Navier–Stokes equations, far-field boundary conditions. *Journal of Computational Physics*, 225(1):1020–1038, 2007.

[227] Magnus Svärd and Jan Nordström. Stability of finite volume approximations for the Laplacian operator on quadrilateral and triangular grids. *Applied Numerical Mathematics*, 51(1):101–125, 2004.

[228] Magnus Svärd and Jan Nordström. On the order of accuracy for difference approximations of initial-boundary value problems. *Journal of Computational Physics*, 218(1):333–352, 2006.

[229] Magnus Svärd and Jan Nordström. A stable high-order finite difference scheme for the compressible Navier–Stokes equations: No-slip wall boundary conditions. *Journal of Computational Physics*, 227(10):4805–4824, 2008.

[230] Magnus Svärd and Hatice Özcan. Entropy-stable schemes for the Euler equations with far-field and wall boundary conditions. *Journal of Scientific Computing*, 58(1):61–89, 2014.

[231] Kasia Świrydowicz, Noel Chalmers, Ali Karakus, and Timothy Warburton. Acceleration of tensor-product operations for high-order finite element methods. *arXiv preprint arXiv:1711.00903*, 2017.

[232] Costas Emmanuel Synolakis. The runup of solitary waves. *Journal of Fluid Mechanics*, 185:523–545, 1987.

[233] Eitan Tadmor. Numerical viscosity and the entropy condition for conservative difference schemes. *Mathematics of Computation*, 43:369–381, 1984.

[234] Eitan Tadmor. The numerical viscosity of entropy stable schemes for systems of conservation laws. i. *Mathematics of Computation*, 49(179):91–103, 1987.

[235] Eitan Tadmor. Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems. *Acta Numerica*, 12:451–512, 5 2003.

[236] Eitan Tadmor and Weigang Zhong. Entropy stable approximations of Navier–Stokes equations with no artificial numerical viscosity. *Journal of Hyperbolic Differential Equations*, 3(03):529–559, 2006.

[237] P. A. Tassi, O. Bokhove, and C. A. Vionnet. Space discontinuous Galerkin method for shallow water flows–kinetic and HLLC flux, and potential vorticity generation. *Advances in Water Resourses*, 30(4):998–1015, 2007.

[238] Julien Thibault and Inanc Senocak. CUDA implementation of a Navier-Stokes solver on multi-GPU desktop platforms for incompressible flows. In *47th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, page 758, 2009.

[239] top500.org. Top 500 GPUs. `https://www.top500.org`. Accessed: 2018-00-19.

[240] E.F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer, June 1999.

[241] USGS. United States Geological Survey. `http://usgs.gov/`.

[242] Stefan Vater, Nicole Beisiegel, and Jörn Behrens. A limiter-based well-balanced discontinuous Galerkin method for shallow-water flows with wetting and drying: One-dimensional case. *Advances in water resources*, 85:1–13, 2015.

[243] Marıa Elena Vázquez-Cendón. Improved treatment of source terms in upwind schemes for the shallow water equations in channels with irregular geometry. *Journal of Computational Physics*, 148(2):497–526, 1999.

[244] Brian C Vermeire, Freddie D Witherden, and Peter E Vincent. On the utility of GPU accelerated high-order methods for unsteady flow simulations: A comparison with industry-standard tools. *Journal of Computational Physics*, 334:497–521, 2017.

[245] Peter E Vincent, Patrice Castonguay, and Antony Jameson. A new class of high-order energy stable flux reconstruction schemes. *Journal of Scientific Computing*, 47(1):50–72, 2011.

[246] Peter E Vincent, Antony M Farrington, Freddie D Witherden, and Antony Jameson. An extended range of stable-symmetric-conservative flux reconstruction correction functions. *Computer Methods in Applied Mechanics and Engineering*, 296:248–272, 2015.

[247] Senka Vukovic and Luka Sopta. ENO and WENO schemes with the exact conservation property for one-dimensional shallow water equations. *Journal of Computational Physics*, 179(2):593–621, 2002.

[248] Timothy Warburton. A low-storage curvilinear discontinuous Galerkin method for wave problems. *SIAM Journal on Scientific Computing*, 35(4):A1987–A2012, 2013.

[249] Pål Wessel and Walter HF Smith. A global, self-consistent, hierarchical, high-resolution shoreline database. *Journal of Geophysical Research: Solid Earth*, 101(B4):8741–8743, 1996.

[250] G. B. Whitham. *Linear and Nonlinear Waves.* John Wiley and Sons, New York, 1974.

[251] Lucas C Wilcox, Francis X Giraldo, Timothy Campbell, Andreas Klöckner, Timothy Warburton, and Timothy Whitcomb. NPS-NRL-Rice-UIUC collaboration on navy atmosphere-ocean coupled models on many-core computer architectures annual report. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA DEPT OF APPLIED MATHEMATICS, 2014.

[252] DM Williams and Antony Jameson. Energy stable flux reconstruction schemes for advection–diffusion problems on tetrahedra. *Journal of Scientific Computing*, 59(3):721–759, 2014.

[253] Niklas Wintermeyer, Andrew R. Winters, Gregor J. Gassner, and David A. Kopriva. An entropy stable nodal discontinuous Galerkin method for the two dimensional shallow water equations on unstructured curvilinear meshes with discontinuous bathymetry. *Journal of Computational Physics*, 340:200–242, 2017.

[254] Niklas Wintermeyer, Andrew R Winters, Gregor J Gassner, and Timothy Warburton. An entropy stable discontinuous Galerkin method for the shallow water equations on curvilinear meshes with wet/dry fronts accelerated by GPUs. *arXiv preprint arXiv:1804.02221*, 2018.

[255] Andrew R Winters, Dominik Derigs, Gregor J Gassner, and Stefanie Walch. A uniquely defined entropy stable matrix dissipation operator for high mach number ideal MHD and compressible Euler simulations. *Journal of Computational Physics*, 332:274–289, 2017.

[256] Andrew R Winters and Gregor J Gassner. A comparison of two entropy stable discontinuous Galerkin spectral element approximations for the shallow water equations with non-constant topography. *Journal of Computational Physics*, 301:357–376, 2015.

[257] Andrew R Winters and Gregor J Gassner. Affordable, entropy conserving and entropy stable flux functions for the ideal MHD equations. *Journal of Computational Physics*, 304:72–108, 2016.

[258] Andrew R Winters, Rodrigo C Moura, Gianmarco Mengaldo, Gregor J Gassner, Stefanie Walch, Joaquim Peiro, and Spencer J Sherwin. A comparative study on polynomial dealiasing and split form discontinuous Galerkin schemes for under-resolved turbulence computations. *Journal of Computational Physics*, 2018.

[259] Freddie D Witherden, Antony M Farrington, and Peter E Vincent. PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach. *Computer Physics Communications*, 185(11):3028–3040, 2014.

[260] Freddie D Witherden, Brian C Vermeire, and Peter E Vincent. Heterogeneous computing on mixed unstructured grids with PyFR. *Computers & Fluids*, 120:173–186, 2015.

[261] Wang Xian and Aoki Takayuki. Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster. *Parallel Computing*, 37(9):521–535, 2011.

[262] Yulong Xing and Chi-Wang Shu. A new approach of high order well-balanced finite volume WENO schemes and discontinuous Galerkin methods for a class of hyperbolic systems with source terms. *Communications in Computational Physics*, 1(1):100–134, 2006.

[263] Yulong Xing and Chi-Wang Shu. High-order finite volume WENO schemes for the shallow water equations with dry states. *Advances in Water Resources*, 34(8):1026–1038, 2011.

[264] Yulong Xing and Xiangxiong Zhang. Positivity-preserving well-balanced discontinuous Galerkin methods for the shallow water equations on unstructured triangular meshes. *Journal of Scientific Computing*, 57(1):19–41, 2013.

[265] Yulong Xing, Xiangxiong Zhang, and Chi-Wang Shu. Positivity-preserving high order well-balanced discontinuous Galerkin methods for the shallow water equations. *Advances in Water Resources*, 33(12):1476–1493, 2010.

[266] Xue-Jun Yang, Xiang-Ke Liao, Kai Lu, Qing-Feng Hu, Jun-Qiang Song, and Jin-Shu Su. The TianHe-1A supercomputer: its hardware and software. *Journal of computer science and technology*, 26(3):344–351, 2011.

[267] Hamed Zakerzadeh and Ulrik S Fjordholm. High-order accurate, fully discrete entropy stable schemes for scalar conservation laws. *IMA Journal of Numerical Analysis*, 36(2):633–654, 2015.

[268] Peter Zaspel and Michael Griebel. Solving incompressible two-phase flows on multi-GPU clusters. *Computers & Fluids*, 80:356–364, 2013.

[269] Mengping Zhang and Chi-Wan Shu. An analysis of three different formulations of the discontinuous Galerkin method for diffusion equations. *Mathematical Models and Methods in Applied Sciences*, 13(3):395–413, 2003.

[270] Xiangxiong Zhang and Chi-Wang Shu. On maximum-principle-satisfying high order schemes for scalar conservation laws. *Journal of Computational Physics*, 229(9):3091–3120, 2010.

[271] Xiangxiong Zhang and Chi-Wang Shu. On positivity-preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes. *Journal of Computational Physics*, 229(23):8918–8934, 2010.

[272] Jian G Zhou, Derek M Causon, Clive G Mingham, and David M Ingram. The surface gradient method for the treatment of source terms in the shallow-water equations. *Journal of Computational physics*, 168(1):1–25, 2001.

A. Appendix

A.1. Proof of Lemma 1

Proof. We start by showing that the total energy is an entropy function $e = \frac{1}{2}hu^2 + \frac{1}{2}gh^2 + ghb$ for the one dimensional shallow water equations. The entropy variables $\vec{q}$ in terms of the conservative variables $\vec{w}$ are

$$\vec{q} := \frac{\partial e}{\partial \vec{w}} = \begin{pmatrix} g(h+b) - \frac{1}{2}u^2 \\ u \end{pmatrix} = \begin{pmatrix} g(w_1 + b) - \frac{1}{2}\left(\frac{w_2}{w_1}\right)^2 \\ \frac{w_2}{w_1}. \end{pmatrix} \tag{A.1}$$

To show convexity, we differentiate again to compute the Hessian of the entropy function,

$$\frac{\partial e}{\partial \vec{w}^2} = \begin{pmatrix} g + \frac{w_2^2}{w_1^3} & -\frac{w_2}{w_1^2} \\ -\frac{w_2}{w_1^2} & \frac{1}{w_1} \end{pmatrix}. \tag{A.2}$$

The matrix is clearly symmetric. We check for positive definiteness by computing the minors. The first minor, $g + \frac{w_2^2}{w_1^3}$ is positive for $w_1 = h > 0$. For the second minor we see

$$\det \begin{pmatrix} g + \frac{w_2^2}{w_1^3} & -\frac{w_2}{w_1^2} \\ -\frac{w_2}{w_1^2} & \frac{1}{w_1} \end{pmatrix} = g\frac{1}{w_1} \tag{A.3}$$

which is positive for the physical assumpation $w_1 = h > 0$. Contracting the system (2.29) by the entropy variables (2.43) we find

$$\begin{aligned} \vec{q}^T\left(\vec{f}_x - \vec{S}\right) &= \left(g(h+b) - \tfrac{1}{2}u^2, \quad u\right)\left(\begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{pmatrix}_x + \begin{pmatrix} 0 \\ ghb_x \end{pmatrix}\right) \\ &= g(h+b)(hu)_x - \frac{1}{2}u^2(hu)_x + u(hu^2)_x + \frac{g}{2}u(h^2)_x + ghub_x \\ &= \left(ghu(h+b) + \frac{1}{2}hu^3\right)_x = \mathcal{F}_x, \end{aligned} \tag{A.4}$$

where we applied the product rule to get the last equality. Expressing the conservative variables in terms of entropy variables we find

$$\begin{aligned} w_1(\vec{q}) &= \frac{1}{g}q_1 + \frac{1}{2g}q_2^2 - b \\ w_2(\vec{q}) &= w_1 q_2 = \frac{1}{g}q_1 q_2 + \frac{1}{2g}q_2^3 - bq_2. \end{aligned} \tag{A.5}$$

The associated entropy Jacobian $\mathbf{H} = \vec{w}_{\vec{q}}$ is

$$\mathbf{H} := \frac{1}{g}\begin{pmatrix} 1 & q_2 \\ q_2 & q_1 + \frac{3}{2}q_2^2 - gb \end{pmatrix} = \frac{1}{g}\begin{pmatrix} 1 & u \\ u & gh + u^2 \end{pmatrix}. \tag{A.6}$$

In the case of $b = 0$, we note that $h = \frac{1}{g}q_1 + \frac{1}{2g}q_2^2$ and the flux expressed in terms of the entropy variables is

$$\vec{f}(\vec{q}) = \begin{pmatrix} hq_2 \\ hq_2^2 + \frac{g}{2}h^2 \end{pmatrix}. \tag{A.7}$$

The derivatives of h and $h^2 = \frac{1}{g^2}\left(q_1^2 + q_1q_2^2 + \frac{1}{4}q_2^4\right)$ with respect to the entropy variables are

$$\begin{aligned} h_{\vec{q}} &= \frac{1}{g}\left(1, q_2\right), \\ h^2_{\vec{q}} &= \frac{1}{g^2}\left(2q_1 + q_2^2, 2q_1q_2 + q_2^3\right). \end{aligned} \tag{A.8}$$

With (A.8), we find that the derivatives of the fluxes with respect to the entropy variables are

$$\begin{aligned} \vec{f}_{\vec{q}} &= \frac{1}{g}\begin{pmatrix} q_2 & gh + q_2^2 \\ q_2^2 + \frac{1}{2}(2q_1 + q_2^2) & q_2^3 + 2ghq_2 + q_1q_2 + \frac{1}{2}q_2^3 \end{pmatrix} \\ &= \frac{1}{g}\begin{pmatrix} u & gh + u^2 \\ u^2 + gh & u^3 + 3ghu \end{pmatrix} \end{aligned} \tag{A.9}$$

We use (A.6) and (A.9) in (2.8)

$$\begin{aligned} \vec{w}_{\vec{q}}\vec{q}_t + \vec{f}_{\vec{q}}\vec{q}_x &= \frac{1}{g}\begin{pmatrix} 1 & u \\ u & gh + u^2 \end{pmatrix}\vec{q}_t + \frac{1}{g}\begin{pmatrix} u & gh + u^2 \\ u^2 + gh & u^3 + 3ghu \end{pmatrix}\vec{q}_x \\ &= \begin{pmatrix} \left(h + b - \frac{1}{2g}u^2\right)_t + \frac{1}{g}u\,(u)_t \\ \frac{1}{g}u\left(g(h+b) - \frac{1}{2}u^2\right)_t + \left(h + \frac{1}{g}u^2\right)u_t \end{pmatrix} + \frac{1}{g}\begin{pmatrix} u\left(gh - \frac{1}{2}u^2\right)_x + (gh + u^2)\,u_x \\ (u^2 + gh)\left(gh - \frac{1}{2}u^2\right)_x + (u^3 + 3ghu)\,u_x \end{pmatrix} \\ &= \begin{pmatrix} h_t \\ (hu)_t \end{pmatrix} + \begin{pmatrix} (hu)_x \\ (\frac{g}{2}h^2)_x + (u^2)\,(h)_x + (2hu)\,u_x \end{pmatrix} \\ &= \begin{pmatrix} h_t \\ (hu)_t \end{pmatrix} + \begin{pmatrix} hu \\ \frac{g}{2}h^2 + hu^2 \end{pmatrix}_x \\ &= \vec{w}_t + \vec{f}_x. \end{aligned} \tag{A.10}$$

Thus, we demonstrated that the requirements (2.7) and (2.8) are fulfilled. For the second part of the Lemma we examine the entropy potential Φ and the entropy flux potential Ψ. The entropy potential and entropy flux potential are

$$\begin{aligned} \Phi &= \langle q, w \rangle - e \\ &= gh(h+b) - \frac{1}{2}hu^2 + hu^2 - \frac{1}{2}hu^2 - \frac{1}{2}gh^2 - ghb \\ &= \frac{1}{2}gh^2, \end{aligned} \tag{A.11}$$

and

$$\begin{aligned}\Psi &= \langle q, f\rangle - \mathcal{F}\\ &= ghu(h+b) - \frac{1}{2}hu^3 + u(hu^2 + \frac{1}{2}gh^2) - \frac{1}{2}hu^3 - ghu(h+b)\\ &= \frac{1}{2}gh^2u.\end{aligned} \tag{A.12}$$

We verify that the pair satisfies property (2.27) for the purely conservative equations (i.e. $b = 0$) by first expressing the entropy potential and the entropy flux potential in terms of the entropy variables

$$\begin{aligned}\frac{1}{2g}q_1^2 + \frac{1}{2g}q_1q_2^2 + \frac{1}{8g}q_2^4 &= \frac{1}{2}gh^2 - \frac{1}{2}hu^2 + \frac{1}{8g}u^4 + \frac{1}{2}hu^2 - \frac{1}{4g}u^4 + \frac{1}{8g}u^4\\ &= \frac{1}{2}gh^2 = \Phi,\\ q_2\Phi &= \frac{1}{2g}q_1^2q_2 + \frac{1}{2g}q_1q_2^3 + \frac{1}{8g}q_2^5\\ &= \frac{1}{2}gh^2u = \Psi.\end{aligned} \tag{A.13}$$

Taking the derivatives with respect to the entropy variables we see

$$\begin{aligned}\Phi_{q_1} &= \frac{1}{g}q_1 + \frac{1}{2g}q_2^2 = h - \frac{1}{2g}u^2 + \frac{1}{2g}u^2 = h = w_1,\\ \Phi_{q_2} &= \frac{1}{g}q_1q_2 + \frac{1}{2g}q_2^3 = hu = w_2,\end{aligned} \tag{A.14}$$

which is exactly the symmetrization property of the entropy potential (2.27). Similarly we see that the derivatives of the entropy potential flux with respect to the entropy variables are exactly the physical fluxes,

$$\begin{aligned}\Psi_{q_1} &= \frac{1}{g}q_1q_2 + \frac{1}{2g}q_2^2 = hu = f_1,\\ \Psi_{q_2} &= \frac{1}{2}gh^2 + hu = f_2.\end{aligned} \tag{A.15}$$

■

A.2. Proof of Lemma 2

Proof. We prove that the total energy is an entropy function

$$e = \frac{1}{2}h\left(u^2 + v^2\right) + \frac{1}{2}gh^2 + ghb, \tag{A.16}$$

for the two dimensional shallow water equations. We express the entropy variables $\vec{q}$ in terms of the conservative variables $\vec{w}$:

$$\vec{q} := \frac{\partial e}{\partial \vec{w}} = \begin{pmatrix} g(h+b) - \frac{1}{2}u^2 - \frac{1}{2}v^2\\ u\\ v\end{pmatrix} = \begin{pmatrix} g(w_1+b) - \frac{1}{2}\frac{w_2^2}{w_1^2} - \frac{1}{2}\frac{w_3^2}{w_1^2}\\ \frac{w_2}{w_1}\\ \frac{w_3}{w_1}.\end{pmatrix} \tag{A.17}$$

The Hessian of this entropy function is

$$\frac{\partial e}{\partial \vec{w}^2} = \begin{pmatrix} g + \frac{w_2^2}{w_1^3} + \frac{w_3^2}{w_1^3} & -\frac{w_2}{w_1^2} & -\frac{w_3}{w_1^2} \\ -\frac{w_2}{w_1^2} & \frac{1}{w_1} & 0 \\ -\frac{w_3}{w_1^2} & 0 & \frac{1}{w_1} \end{pmatrix}, \tag{A.18}$$

and is obviously symmetric. Checking the minors once more, we find the first minor to be positive, $g + \frac{w_2^2}{w_1^3} + \frac{w_3^2}{w_1^3} > 0$. For the second minor we see

$$\det \begin{pmatrix} g + \frac{w_2^2}{w_1^3} + \frac{w_3^2}{w_1^3} & -\frac{w_2}{w_1^2} \\ -\frac{w_2}{w_1^2} & \frac{1}{w_1} \end{pmatrix} = g\frac{1}{w_1} + \frac{w_3^2}{w_1^4} \tag{A.19}$$

which is positive for $w_1 = h > 0$. Finally, for the determinant of the Hessian itself we see

$$\det \begin{pmatrix} g + \frac{w_2^2}{w_1^3} + \frac{w_3^2}{w_1^3} & -\frac{w_2}{w_1^2} & -\frac{w_3}{w_1^2} \\ -\frac{w_2}{w_1^2} & \frac{1}{w_1} & 0 \\ -\frac{w_3}{w_1^2} & 0 & \frac{1}{w_1} \end{pmatrix} = \frac{g}{w_1^2} + \frac{w_2^2}{w_1^5} + \frac{w_3^2}{w_1^5} - \frac{w_3^2}{w_1^5} \quad - \frac{w_2^2}{w_1^5} = \frac{g}{w_1^2} > 0. \tag{A.20}$$

Thus the Hessian is SPD by Sylvester's criterion and e is convex. Contracting the system (2.33) by the entropy variables (2.48) we recover the divergence of the entropy fluxes

$$\begin{aligned}
&\vec{q}^{T}\left(\vec{f}_x + \vec{g}_y - \vec{s}\right) \\
&= \left(g(h+b) - \tfrac{1}{2}u^2 - \tfrac{1}{2}v^2, \quad u, \quad v\right)\left(\begin{pmatrix} hu \\ hu^2 + \tfrac{1}{2}gh^2 \\ huv \end{pmatrix}_x + \begin{pmatrix} hv \\ huv \\ hv^2 + \tfrac{1}{2}gh^2 \end{pmatrix}_y + \begin{pmatrix} 0 \\ ghb_x \\ ghb_y \end{pmatrix}\right) \\
&= g(h+b)(hu)_x - \frac{1}{2}u^2(hu)_x - \frac{1}{2}v^2(hu)_x + u(hu^2)_x + v(huv)_x + \frac{g}{2}u(h^2)_x + ghub_x \\
&\quad + g(h+b)(hv)_y - \frac{1}{2}u^2(hv)_y - \frac{1}{2}v^2(hv)_y + u(huv)_y + v(hv^2)_y + \frac{g}{2}v(h^2)_y + ghvb_y \\
&= \left(ghu(h+b) + \frac{1}{2}hu^3 + \frac{1}{2}huv^2\right)_x + \left(ghv(h+b) + \frac{1}{2}hv^3 + \frac{1}{2}hu^2v\right)_y \\
&= \mathcal{F}_x + \mathcal{G}_y.
\end{aligned} \tag{A.21}$$

The conservative variables in terms of entropy variables are

$$\begin{aligned}
w_1(\vec{q}) &= \frac{1}{g}q_1 + \frac{1}{2g}q_2^2 + \frac{1}{2g}q_3^2 - b, \\
w_2(\vec{q}) &= \frac{1}{g}q_1q_2 + \frac{1}{2g}q_2^3 + \frac{1}{2g}q_2q_3^2 - bq_2, \\
w_3(\vec{q}) &= \frac{1}{g}q_1q_3 + \frac{1}{2g}q_3^3 + \frac{1}{2g}q_3q_2^2 - bq_2.
\end{aligned} \tag{A.22}$$

The associated entropy Jacobian $\mathbf{H} = \vec{w}_{\vec{q}}$ is

$$\begin{aligned}\mathbf{H} &:= \frac{1}{g}\begin{pmatrix}1 & q_2 & q_3\\ q_2 & q_1 + \frac{3}{2}q_2^2 + \frac{1}{2}q_3^2 - gb & q_2q_3\\ q_3 & q_2q_3 & q_1 + \frac{3}{2}q_3^2 + \frac{1}{2}q_2^2 - gb\end{pmatrix}\\ &= \frac{1}{g}\begin{pmatrix}1 & u & v\\ u & gh+u^2 & uv\\ v & uv & gh+v^2\end{pmatrix}.\end{aligned} \tag{A.23}$$

In the case of $b = 0$, we note that $h = \frac{1}{g}q_1 + \frac{1}{2g}\left(q_2^2 + q_3^2\right)$ and the fluxes expressed in terms of the entropy variables is

$$\vec{f}(\vec{q}) = \begin{pmatrix}hq_2\\ hq_2^2 + \frac{g}{2}h^2\\ hq_2q_3\end{pmatrix}, \qquad \vec{g}(\vec{q}) = \begin{pmatrix}hq_3\\ hq_2q_3\\ hq_3^2 + \frac{g}{2}h^2\end{pmatrix}. \tag{A.24}$$

The derivatives of h and $h^2 = \frac{1}{g^2}\left(q_1^2 + q_1(q_2^2q_3^2) + \frac{1}{4}(q_2^4 + q_2^2q_3^2 + q_3^4)\right)$ with respect to the entropy variables are

$$\begin{aligned}h_q &= \frac{1}{g}\left(1, \quad q_2, \quad q_3\right)\\ h_q^2 &= \frac{1}{g^2}\left(2q_1 + q_2^2q_3^2, \quad 2q_1q_2q_3^2 + q_2^3 + \frac{1}{2}q_2q_3^2, \quad 2q_1q_2^2q_3 + q_3^3 + \frac{1}{2}q_2^2q_3\right).\end{aligned} \tag{A.25}$$

With (A.25), we find that the flux derivatives with respect to the entropy variables are

$$\begin{aligned}\vec{f}_{\vec{q}} &= \frac{1}{g}\begin{pmatrix}q_2 & q_2^2 + gh & q_3q_2\\ q_2^2 + gh & q_2^3 + 3ghq_2 & q_2^2q_3 + ghq_3\\ q_2q_3 & q_2^2q_3 + ghq_3 & q_2q_3^2 + ghq_2\end{pmatrix}\\ &= \frac{1}{g}\begin{pmatrix}u & u^2 + gh & uv\\ u^2 + gh & u^3 + 3ghu & u^2v + ghv\\ uv & u^2v + ghv & uv^2 + ghu\end{pmatrix},\end{aligned} \tag{A.26}$$

and also for $\vec{g}$

$$\begin{aligned}\vec{g}_{\vec{q}} &= \frac{1}{g}\begin{pmatrix}q_3 & q_3q_2 & q_3^2 + gh\\ q_2q_3 & q_2^2q_3 + ghq_3 & q_3^2q_2 + ghq_2\\ q_3^2 + gh & q_2q_3^2 + ghq_2 & q_3^3 + 3ghq_3\end{pmatrix}\\ &= \frac{1}{g}\begin{pmatrix}v & vu & v^2 + gh\\ uv & u^2v + ghv & v^2u + ghu\\ v^2 + gh & uv^2 + ghu & v^3 + 3ghv\end{pmatrix},\end{aligned} \tag{A.27}$$

To show that these terms lead to the symmetric form (2.25), we must show that the terms contract to the correct terms from the original equation, i.e. $\vec{w}_t$, $\vec{f}_x$ and $\vec{g}_y$, individually. We use the entropy Jacobian (A.23) along with the flux Jacobians (A.26) and (A.27) and inspect the terms individually, starting with $w_q q_t$. We use the product rule to recover the time derivative of the conservative variables,

$$\vec{w}_{\vec{q}}\,\vec{q}_t = \frac{1}{g}\begin{pmatrix}1 & u & v\\ u & gh+u^2 & uv\\ v & uv & gh+v^2\end{pmatrix}\begin{pmatrix}g(h+b)-\frac{1}{2}u^2-\frac{1}{2}v^2\\ u\\ v\end{pmatrix}_t = \begin{pmatrix}h\\ hu\\ hv\end{pmatrix}_t = \vec{w}_t. \tag{A.28}$$

For the fluxes we see

$$\begin{aligned}\vec{f}_{\vec{q}}\,\vec{q}_x &= \frac{1}{g}\begin{pmatrix}u & u^2+gh & uv\\ u^2+gh & u^3+3ghu & u^2v+ghv\\ uv & u^2v+ghv & uv^2+ghu\end{pmatrix}\begin{pmatrix}g(h+b)-\frac{1}{2}u^2-\frac{1}{2}v^2\\ u\\ v\end{pmatrix}_x\\ &= \begin{pmatrix}hu\\ hu^2+\frac{g}{2}h^2\\ huv\end{pmatrix}_x = \vec{f}_x,\end{aligned} \tag{A.29}$$

and

$$\begin{aligned}\vec{g}_{\vec{q}}\,\vec{q}_y &= \frac{1}{g}\begin{pmatrix}v & vu & v^2+gh\\ uv & u^2v+ghv & v^2u+ghu\\ v^2+gh & uv^2+ghu & v^3+3ghv\end{pmatrix}\begin{pmatrix}g(h+b)-\frac{1}{2}u^2-\frac{1}{2}v^2\\ u\\ v\end{pmatrix}_y\\ &= \begin{pmatrix}hv\\ huv\\ hv^2+\frac{g}{2}h^2\end{pmatrix}_y = \vec{g}_y.\end{aligned} \tag{A.30}$$

Thus, we have shown both requirements (2.24) and (2.25) and conclude that the total energy is an entropy function for the two dimensional shallow water equations. The entropy potential and entropy potential fluxes are in this case found to be

$$\begin{aligned}\Phi &= \langle \vec{q}, \vec{w}\rangle - e\\ &= gh(h+b) - \frac{1}{2}hu^2 - \frac{1}{2}hv^2 + hu^2 + hv^2 - \frac{1}{2}hu^2 - \frac{1}{2}hv^2 - \frac{1}{2}gh^2 - ghb\\ &= \frac{1}{2}gh^2\end{aligned} \tag{A.31}$$

and

$$\begin{aligned}\Psi_f =& \langle q, f\rangle - \mathcal{F}\\ =& ghu(h+b) - \frac{1}{2}hu^3 - \frac{1}{2}huv^2 + u(hu^2 + \frac{1}{2}gh^2) + huv^2\\ &- ghu(h+b) - \frac{1}{2}hu^3 - \frac{1}{2}huv^2\\ =& \frac{1}{2}gh^2u\end{aligned} \tag{A.32}$$

as well as

$$\begin{aligned}\Psi_g =&\langle q, g\rangle - \mathcal{G}\\ =&ghv(h+b) - \frac{1}{2}hu^2v - \frac{1}{2}hv^3 + hu^2v + hv^3 + \frac{1}{2}gh^2v\\ &- ghv(h+b) - \frac{1}{2}hu^2v - \frac{1}{2}hv^3\\ =&\frac{1}{2}gh^2v.\end{aligned} \tag{A.33}$$

We verify that the triplet (Φ, Ψ_f, Ψ_g) satisfies property (2.27) for the purely conservative equations (i.e. $b = 0$) by first expressing the entropy potential and the entropy potential flux in terms of the entropy variables. We start with the entropy potential Φ,

$$\begin{aligned}\Phi =&\frac{1}{2g}q_1^2 + \frac{1}{2g}q_1\left(q_2^2+q_3^2\right) + \frac{1}{8g}\left(q_2^2+q_3^2\right)^2\\ =&\frac{1}{2}gh^2 - \frac{1}{2}h(u^2+v^2) + \frac{1}{8g}(u^2+v^2)^2\\ &+\frac{1}{2}h\left(u^2+v^2\right) - \frac{1}{4g}\left(u^2+v^2\right)^2 + \frac{1}{8g}\left(u^2+v^2\right)\\ =&\frac{1}{2}gh^2.\end{aligned} \tag{A.34}$$

We proceed with the entropy potential fluxes

$$\begin{aligned}\Psi_f = q_2\Phi &= \frac{1}{2g}q_1^2q_2 + \frac{1}{2g}q_1q_2\left(q_2^2+q_3^2\right) + \frac{1}{8g}q_2\left(q_2^2+q_3^2\right)^2\\ &= \frac{1}{2}gh^2u,\end{aligned} \tag{A.35}$$

and

$$\begin{aligned}\Psi_g = q_3\Phi &= \frac{1}{2g}q_1^2q_3 + \frac{1}{2g}q_1q_3\left(q_2^2+q_3^2\right) + \frac{1}{8g}q_3\left(q_2^2+q_3^2\right)^2\\ &= \frac{1}{2}gh^2v.\end{aligned} \tag{A.36}$$

Taking the derivatives with respect to the entropy variables we see

$$\begin{aligned}\Phi_{q_1} &= \frac{1}{g}q_1 + \frac{1}{2g}\left(q_2^2+q_3^2\right) = h = w_1,\\ \Phi_{q_2} &= \frac{1}{g}q_1q_2 + \frac{1}{2g}q_2^3 + \frac{1}{2g}q_2q_3^2 = hu = w_2,\\ \Phi_{q_3} &= hu = w_2,\end{aligned} \tag{A.37}$$

which is exactly the symmetrization property of the entropy potential (2.27). Similarly we see that the derivatives of the entropy potential fluxes with respect to the entropy variables are exactly the physical fluxes,

$$\begin{aligned}(\Psi_f)_{\vec{q}} &= \vec{f},\\ (\Psi_g)_{\vec{q}} &= \vec{g}.\end{aligned} \tag{A.38}$$

■

A.3. Proof of Lemma 4

Proof. Without loss of generality, we examine $\mathcal{E}_{\alpha\beta}$. The discrete mean value of the correction term is given by

$$\begin{aligned}
&\sum_{i=0}^{N}\sum_{j=0}^{N}\left(\mathcal{E}_{\alpha\beta}\right)_{ij}\omega_i\omega_j\\
&=\sum_{i=0}^{N}\sum_{j=0}^{N}\left(-\sum_{m=0}^{N}D_{im}\alpha_{mj}\beta_{mj}+\alpha_{ij}\sum_{m=0}^{N}D_{im}\beta_{mj}+\beta_{ij}\sum_{m=0}^{N}D_{im}\alpha_{mj}\right)\omega_i\omega_j\\
&=-\sum_{j=0}^{N}\left(\sum_{i=0}^{N}\sum_{m=0}^{N}\left(B_{im}-D_{mi}\omega_m\right)\alpha_{mj}\beta_{mj}\right)\omega_j\\
&\quad+\sum_{j=0}^{N}\left(\sum_{i=0}^{N}\alpha_{ij}\sum_{m=0}^{N}\left(B_{im}-D_{mi}\omega_m\right)\beta_{mj}\right)\omega_j\\
&\quad+\sum_{i=0}^{N}\sum_{j=0}^{N}\left(\beta_{ij}\sum_{m=0}^{N}D_{im}\alpha_{mj}\right)\omega_i\omega_j\\
&=\sum_{j=0}^{N}\sum_{m=0}^{N}\alpha_{mj}\beta_{mj}\left(\sum_{i=0}^{N}Q_{mi}\right)\omega_j-\sum_{j=0}^{N}\left(\sum_{i=0}^{N}\alpha_{ij}\sum_{m=0}^{N}Q_{mi}\beta_{mj}\right)\omega_j\\
&\quad-\sum_{j=0}^{N}\left(\sum_{i=0}^{N}\left(B_{ii}\right)\alpha_{ij}\beta_{ij}\right)\omega_j+\sum_{j=0}^{N}\left(\sum_{i=0}^{N}\alpha_{ij}\left(B_{ii}\right)\beta_{ij}\right)\omega_j\\
&\quad+\sum_{i=0}^{N}\sum_{j=0}^{N}\left(\beta_{ij}\sum_{m=0}^{N}D_{im}\alpha_{mj}\right)\omega_i\omega_j\\
&=-\sum_{j=0}^{N}\left(\sum_{i=0}^{N}\alpha_{ij}\sum_{m=0}^{N}Q_{mi}\beta_{mj}\right)\omega_j+\sum_{i=0}^{N}\sum_{j=0}^{N}\left(\beta_{ij}\sum_{m=0}^{N}Q_{im}\alpha_{mj}\right)\omega_j\\
&=\sum_{j=0}^{N}\left(-\sum_{m=0}^{N}\beta_{mj}\sum_{i=0}^{N}\alpha_{ij}Q_{mi}+\sum_{i=0}^{N}\beta_{ij}\sum_{m=0}^{N}Q_{im}\alpha_{mj}\right)\omega_j=0,
\end{aligned}\tag{A.39}$$

where we used $\omega_i D_{im} = B_{im} - D_{mi}\omega_m$, the SBP properties from Corollary 1 and the structure of $\mathbf{B}$. In the last line we see that the two inner sums are identical apart from swapped i, m notation and thus cancel out. ■

A.4. Proof of Lemma 8

Proof. We prove the simplified flux difference formula (4.37) in i-direction and fix j. We recall the definition of the high-order flux extension on curvilinear grids (4.33),

$$\begin{aligned}\overline{\tilde{F}}_{0,j} &:= \tilde{F}_{0,j},\\ \overline{\tilde{F}}_{i,j} &:= \sum_{m=i}^{N}\sum_{\ell=0}^{i-1} 2\,Q_{\ell m}\left(F^{\#}_{(\ell,m),j} + G^{\#}_{(\ell,m),j}\right),\\ \overline{\tilde{F}}_{N+1,j} &:= \tilde{F}_{N,j},\end{aligned} \tag{A.40}$$

and consider the special cases $i = 0$ and $i = N$ first. For $i = 0$ we have

$$\begin{aligned}\overline{\tilde{F}}_{1,j} - \overline{\tilde{F}}_{0,j} &= \sum_{m=1}^{N} 2Q_{0m}\left(F^{\#}_{(0,m),j} + G^{\#}_{(0,m),j}\right) - \tilde{F}_{0,j},\\ &= \sum_{m=0}^{N} 2Q_{0m}\left(F^{\#}_{(0,m),j} + G^{\#}_{(0,m),j}\right),\end{aligned} \tag{A.41}$$

where we use the fact that $Q_{00} = -\frac{1}{2}$. For $i = N$ we have

$$\begin{aligned}\overline{\tilde{F}}_{N+1,j} - \overline{\tilde{F}}_{N,j} &= \tilde{F}_{N,j} - \sum_{m=0}^{N-1} 2Q_{mN}\left(F^{\#}_{(m,N),j} + G^{\#}_{(m,N),j}\right),\\ &= \tilde{F}_{N,j} + \sum_{m=0}^{N-1} 2Q_{Nm}\left(F^{\#}_{(m,N),j} + G^{\#}_{(m,N),j}\right),\\ &= \sum_{m=0}^{N} 2Q_{Nm}\left(F^{\#}_{(m,N),j} + G^{\#}_{(m,N),j}\right),\end{aligned} \tag{A.42}$$

where we use the fact that $Q_{NN} = \frac{1}{2}$. For $i = 1, \ldots, N-1$ we examine $\overline{\tilde{\vec{F}}}_{i+1,j}$ to get

$$\begin{aligned}\overline{\tilde{F}}_{i+1,j} &= \sum_{m=i+1}^{N}\sum_{\ell=0}^{i} 2Q_{\ell m}\left(F^{\#}_{(\ell,m),j} + G^{\#}_{(\ell,m),j}\right),\\ &= \sum_{m=i}^{N}\sum_{\ell=0}^{i} 2Q_{\ell m}\left(F^{\#}_{(\ell,m),j} + G^{\#}_{(\ell,m),j}\right) - \sum_{\ell=0}^{i} 2Q_{\ell i}\left(F^{\#}_{(\ell,i),j} + G^{\#}_{(\ell,i),j}\right),\\ &= \sum_{m=i}^{N}\sum_{\ell=0}^{i-1} 2Q_{\ell m}\left(F^{\#}_{(\ell,m),j} + G^{\#}_{(\ell,m),j}\right)\\ &\quad + \sum_{m=i}^{N} 2Q_{im}\left(F^{\#}_{(i,m),j} + G^{\#}_{(i,m),j}\right) - \sum_{\ell=0}^{i} 2Q_{\ell i}\left(F^{\#}_{(\ell,i),j} + G^{\#}_{(\ell,i),j}\right),\\ &= \overline{\tilde{F}}_{i,j} + \sum_{m=0}^{N} 2Q_{im}\left(F^{\#}_{(i,m),j} + G^{\#}_{(i,m),j}\right),\end{aligned} \tag{A.43}$$

where we used that $Q_{ij} = -Q_{ij}$ for $i \neq j$ and $Q_{ii} = 0$ for $i = 1, \ldots, N-1$ in the last step. Thus, we are able to generalise the calculation of the flux difference for $i = 0, \ldots N$

$$\overline{\tilde{F}}_{i+1,j} - \overline{\tilde{F}}_{i,j} = \sum_{m=0}^{N} 2Q_{im} \left(F^{\#}_{(i,m),j} + G^{\#}_{(i,m),j} \right). \tag{A.44}$$

We then premultiply by the inverse of $\mathbf{M}$ to obtain the desired flux differencing result (4.37) for $\sum_{m=0}^{N} M_{ii}^{-1} \Delta_{im} \overline{\tilde{F}}_{mj}$,

$$\frac{\overline{\tilde{F}}_{i+1,j} - \overline{\tilde{F}}_{i,j}}{\omega_i} = \frac{1}{\omega_i} \sum_{m=0}^{N} 2Q_{im} \left(F^{\#}_{(i,m),j} + G^{\#}_{(i,m),j} \right). \tag{A.45}$$

An identical strategy can be used in the j index direction to rewrite the flux difference in the $y-$direction, $\sum_{m=0}^{N} M_{jj}^{-1} \Delta_{jm} \overline{\tilde{G}}_{im}$, in the similar indicial form (4.38). ∎

Erklärung

Ich versichere, dass ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie - abgesehen von unten angegebenen Teilpublikationen - noch nicht veröffentlicht worden ist, sowie, dass ich eine solche Veröffentlichung vor Abschluss des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen der Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. Gregor Gassner betreut worden

Köln, den 28.2.2019

(Niklas Wintermeyer)

Publikationsliste

Verzeichnis veröffentlichter Wissenschaftlicher Arbeiten

1. An entropy stable discontinuous Galerkin method for the shallow water equations on curvilinear meshes with wet/dry fronts accelerated by GPUs (akzeptiert im Journal of Computational Physics, 2018) [254]
2. An entropy stable nodal discontinuous Galerkin method for the two dimensional shallow water equations on unstructured curvilinear meshes with discontinuous bathymetry (veröffentlicht im Journal of Computational Physics, 2017) [253]

Köln, den 28.2.2019

(Niklas Wintermeyer)