Technical Report Series
Center for Data and Simulation Science

Niklas Wintermeyer, Andrew R. Winters, Gregor J. Gassner, Timothy Warburton

An entropy stable discontinuous Galerkin method for the shallow water equations on curvilinear meshes with wet/dry fronts accelerated by GPUs

Technical Report ID: CDS-2018-4
Available at `http://kups.ub.uni-koeln.de/id/eprint/8656`
Submitted on October 2, 2018

An entropy stable discontinuous Galerkin method for the shallow water equations on curvilinear meshes with wet/dry fronts accelerated by GPUs

Niklas Wintermeyer[a,*], Andrew R. Winters[a], Gregor J. Gassner[a], Timothy Warburton[b]

[a] *Mathematisches Institut, Universität zu Köln, Weyertal 86-90, 50931 Köln*

[b] *Department of Mathematics, Virginia Tech, 225 Stanger Street, Blacksburg, VA 24061-0123*

Abstract

We extend the entropy stable high order nodal discontinuous Galerkin spectral element approximation for the non-linear two dimensional shallow water equations presented by Wintermeyer et al. [*N. Wintermeyer, A. R. Winters, G. J. Gassner, and D. 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*] with a shock capturing technique and a positivity preservation capability to handle dry areas. The scheme preserves the entropy inequality, is well-balanced and works on unstructured, possibly curved, quadrilateral meshes. 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. We implement the method on GPU architectures using the abstract language OCCA, a unified approach to multi-threading languages. We 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 challenge the shock capturing and positivity properties of our scheme to verify our theoretical findings.

Keywords: Shallow water equations, Discontinuous Galerkin spectral element method, Shock capturing, Positivity preservation, GPUs, OCCA

1. Introduction

The shallow water equations including a non-constant bottom topography are a system of hyperbolic balance laws

$$\begin{aligned} h_t + (hu)_x + (hv)_y &= 0, \\ (hu)_t + \left(hu^2 + \frac{1}{2} g\, h^2 \right)_x + (huv)_y &= -ghb_x, \\ (hv)_t + (huv)_x + \left(hv^2 + \frac{1}{2} g\, h^2 \right)_y &= -ghb_y, \end{aligned} \tag{1.1}$$

that are useful to model fluid flows in lakes, rivers, oceans or near coastlines, e.g [3, 28, 45]. We compactly write the system (1.1) as

$$\vec{w}_t + \nabla \cdot (\vec{f}, \vec{g})^T = \vec{\mathcal{S}}, \tag{1.2}$$

*Corresponding author

Email address: nwinterm@math.uni-koeln.de (Niklas Wintermeyer)

with $\vec{w} = (h, hu, hv)^T$, $\vec{f} = (hu, hu^2 + \frac{1}{2}gh^2, huv)^T$ and $\vec{g} = (hv, huv, hv^2 + \frac{1}{2}gh^2)^T$ and source term $\vec{\mathcal{S}} = (0, -ghb_x, -ghb_y)^T$. The water height is denoted by $h = h(x, y, t)$ and is measured from the bottom topography $b = b(x, y)$. The total water height is therefore $H = h + b$. The fluid velocities are $u = u(x, y, t)$ and $v = v(x, y, t)$. An important steady state solution of (1.1) is the preservation of a flat lake with no velocity, the so-called "lake at rest" condition

$$\begin{aligned} h + b &= \text{const}, \\ u = v &= 0. \end{aligned} \tag{1.3}$$

A numerical scheme that preserves non-trivial steady state solutions, such as the "lake at rest" problem, is *well-balanced*, e.g. [17, 32]. Methods that are not well-balanced can produce spurious waves in the magnitude of the mesh size truncation error which pollutes the solution quality. This is particularly problematic because many interesting shallow water phenomena can be interpreted as perturbations from the lake at rest condition [26].

Due to the non-linear nature of the shallow water equations (1.1), discontinuous solutions may develop regardless of the smoothness of the initial conditions. Therefore, solutions to the PDEs (1.1) are sought in the weak sense. Unfortunately, weak solutions are non-unique and additional admissibility criteria are required to extract the physically relevant solution from the family of weak solutions. One important criteria for physically relevant solutions is the second law of thermodynamics, which guarantees that the entropy of a physical system increases as the fluid evolves. In mathematics, a suitable strongly convex entropy function can be used to ensure a numerical approximation obeys the laws of thermodynamics discretely [43]. A numerical scheme that satisfies the second law of thermodynamics is said to be *entropy stable*. Due to convention, the sign of the mathematical entropy is reversed when compared to the physical entropy. While the entropy should be (nearly) conserved for smooth solutions, it must be dissipated in the presence of shocks. In order to discuss the mathematical entropy for the shallow water equations a suitable entropy function is the total energy $e = e(\vec{w})$

$$e := \frac{1}{2}h(u^2 + v^2) + \frac{1}{2}gh^2 + ghb. \tag{1.4}$$

We take the derivative of the entropy function with respect to the conservative variables $\vec{w}$ to find the set of entropy variables $\vec{q} = \frac{\partial e}{\partial \vec{w}}$, which are

$$q_1 = gH - \frac{1}{2}(u^2 + v^2), \qquad q_2 = u, \qquad q_3 = v. \tag{1.5}$$

If we contract the shallow water equations (1.1) from the left with the entropy variables and apply consistency conditions on the fluxes developed by Tadmor [42] we obtain the entropy conservation law

$$e_t + \mathcal{F}_x + \mathcal{G}_y = 0, \tag{1.6}$$

with the entropy fluxes $\mathcal{F} = \frac{1}{2}hu(u^2 + v^2) + ghu(h + b)$ and $\mathcal{G} = \frac{1}{2}hv(u^2 + v^2) + ghv(h + b)$. In the presence of discontinuities (1.6) becomes the entropy inequality

$$e_t + \mathcal{F}_x + \mathcal{G}_y \leq 0. \tag{1.7}$$

Unfortunately, for high-order numerical methods the discrete satisfaction of (1.7) does not guarantee that the approximation solution will be overshoot free, e.g. [46]. Therefore, the first contribution of this work is to add a shock capturing method that maintains the entropy stability of the nodal discontinuous Galerkin method (DGSEM) on curvilinear quadrilateral meshes developed by Wintermeyer et al. [46]. In particular, artificial viscosity is added into the two momentum equations. The amount of artificial viscosity is selected with the method developed by Persson and Peraire [33]. Even with shock capturing there can still be issues maintaining the positivity of the water height, h, particularly in flow regions where $h \to 0$. Thus, our second contribution is to incorporate the positivity preserving limiter of Xing et al. [50] in an entropy stable way. To fulfill the requirements of the positivity limiter, we formally show that the entropy stable numerical fluxes of the entropy stable discontinuous Galerkin spectral element method (ESDGSEM) preserve positive mean

water heights on two-dimensional curved meshes. We then generalize a result from Ranocha [37] to show that the positivity preserving limiter itself is entropy stable on curvilinear meshes.

Our third, and final, contribution is to implement the two-dimensional positive ESDGSEM on GPUs. The entropy stable approximation is built with specific split forms, which are linear combinations of the conservative and advective forms of the shallow water equations [17, 46]. However, the method remains fully conservative [11] albeit with additional computational complexity in the form of an increased number of arithmetic operations, but without the need for more data storage and transfer. Therefore, the ESDGSEM seems a perfect candidate for implementation on GPUs. We demonstrate that this expectation is true through careful analysis and detailed discussion of how to implement the numerical method into a unified approach for multi-threading languages OCCA [30].

This work is organized as follows: In Sec. 2 we briefly provide background details of the ESDGSEM described in [46] that serves as the baseline scheme. Next, shock capturing by way of artificial viscosity is discussed in Sec. 3. A positivity preserving limiter is presented in Sec. 4 that maintains the entropy stability of the DGSEM. Implementation details and analysis of the ESDGSEM compared to the standard DGSEM on GPUs is given in Sec. 5. Numerical results that exercise and test the capabilities of the positive ESDGSEM are provided in Sec. 6. Concluding remarks are given in the final section.

2. Entropy Stable Discontinuous Galerkin Spectral Element Method

We briefly present the ESDGSEM developed in [17, 46, 47] which we will extend in the following sections. The entropy stable scheme is based on a nodal DG approximation to the split-form shallow water equations on curved quadrilateral meshes. We start with a brief description of the curvilinear transformations and numerical approximations before summarizing the main properties of the ESDGSEM in Lemma 1. We refer to [19, 24] for a more thorough derivation of DG methods and to [6, 16, 46] for a background on the entropy stable split form DG framework.

2.1. Curvilinear mappings

We separate the domain Ω into K non-overlapping elements $E_k \subset \Omega$ and proceed by constructing a mapping on each element between the computational reference element $E = [-1, 1]^2$ with coordinates (ξ, η) and the physical coordinates (x, y). We use a transfinite interpolation with linear blending [24], where the Jacobian is computed by

$$\mathcal{J} = x_\xi \, y_\eta - x_\eta \, y_\xi. \tag{2.1}$$

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

$$\nabla = \frac{1}{\mathcal{J}} \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix} \hat{\nabla}. \tag{2.2}$$

With a sufficiently smooth mapping we can use (2.2) to replace the physical x and y derivatives in (1.2) to obtain the PDE in reference space

$$\mathcal{J} \vec{w}_t + \hat{\nabla} \cdot (\vec{\tilde{f}}, \vec{\tilde{g}})^T = \vec{\tilde{\mathcal{S}}}, \tag{2.3}$$

where we introduce the contravariant fluxes defined by

$$\begin{aligned} \vec{\tilde{f}}(\vec{w}) &= y_\eta \, \vec{f}(\vec{w}) - x_\eta \, \vec{g}(\vec{w}), \\ \vec{\tilde{g}}(\vec{w}) &= -y_\xi \, \vec{f}(\vec{w}) + x_\xi \, \vec{g}(\vec{w}), \end{aligned} \tag{2.4}$$

and the gradient of the bottom topography transforms by (2.2) to create the contravariant source term $\vec{\tilde{\mathcal{S}}}$ given by

$$\begin{aligned} \tilde{\mathcal{S}}_1 &= 0, \\ \tilde{\mathcal{S}}_2 &= -gh \left(y_\eta b_x - y_\xi b_y \right), \\ \tilde{\mathcal{S}}_3 &= -gh \left(-x_\eta b_x + x_\xi b_y \right). \end{aligned} \tag{2.5}$$

To find the weak form of the balance law in reference space (2.3), we multiply by a smooth test function ϕ and integrate over the reference element. Then we use integration by parts to move the differentiation of the fluxes, $\hat{\nabla} \cdot (\vec{\tilde{f}}, \vec{\tilde{g}})^T$, onto the test function. We replace the fluxes across element interfaces by numerical surface fluxes $\tilde{F}^*$ and $\tilde{G}^*$ and obtain the weak form

$$\int_E \mathcal{J}\, \vec{w}\, \phi \,\mathrm{dE} + \oint_{\partial E} \phi \, \left(\vec{\tilde{F}}^*, \vec{\tilde{G}}^*\right) \cdot \vec{n} \,\mathrm{dS} - \int_E \left(\vec{\tilde{f}}, \vec{\tilde{g}}\right) \cdot \hat{\nabla} \phi \,\mathrm{dE} = \int_E \vec{\tilde{\mathcal{S}}} \phi \,\mathrm{dE}. \tag{2.6}$$

Integrating by parts once more yields the strong form

$$\int_E \mathcal{J}\, \vec{w}\, \phi \,\mathrm{dE} + \oint_{\partial E} \phi \, \left(\vec{\tilde{F}}^* - \vec{\tilde{f}}, \vec{\tilde{G}}^* - \vec{\tilde{g}}\right) \cdot \vec{n} \,\mathrm{dS} + \int_E \hat{\nabla} \cdot \left(\vec{\tilde{f}}, \vec{\tilde{g}}\right)^T \phi \,\mathrm{dE} = \int_E \vec{\tilde{\mathcal{S}}} \phi \,\mathrm{dE}. \tag{2.7}$$

2.2. Numerical Approximations

We approximate quantities of interest such as variables $\vec{w}$ and fluxes $\vec{f}$, $\vec{g}$ by polynomials of degree N and denote them by capital letters $\vec{W}$ or $\vec{F}$ and $\vec{G}$. We use a 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. We write the element-wise polynomial approximation (e.g. for a component W of $\vec{W}$) as

$$w(x,y,t)\big|_{E_k} = w(x(\xi,\eta), y(\xi,\eta), t) \approx W(\xi,\eta,t) := \sum_{i=0}^{N} \sum_{j=0}^{N} W_{i,j}(t)\, \ell_i(\xi)\, \ell_j(\eta), \tag{2.8}$$

where $\{W_{i,j}(t)\}_{i,j=0}^{N,N}$ are the time dependent nodal degrees of freedom and the one-dimensional Lagrange basis functions for the interpolant are

$$\ell_j(\xi) = \prod_{i=0, i\neq j}^{N} \frac{\xi - \xi_i}{\xi_j - \xi_i}, \qquad j = 0, \ldots, N. \tag{2.9}$$

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

$$\frac{\partial}{\partial \xi} W(\xi,\eta,t) = \sum_{i=0}^{N} \sum_{j=0}^{N} W_{i,j}(t)\, \frac{\partial}{\partial \xi} \ell_i(\xi)\, \ell_j(\eta). \tag{2.10}$$

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

$$D_{ij} := \left.\frac{\partial \ell_j}{\partial \xi}\right|_{\xi=\xi_i}, \qquad i,j = 0, \ldots, N, \tag{2.11}$$

which is used to calculate the derivative with respect to ξ at the interpolation nodes. It follows from the tensor product ansatz that the same derivative operator $\mathbf{D}$ can be used to evaluate derivatives in ξ and η direction. We also approximate integrals numerically using LGL quadrature and define the mass matrix $\mathbf{M} = \mathrm{diag}(\omega_0, \omega_1, \ldots, \omega_N)$ with LGL quadrature weights on the diagonal. By the tensor product ansatz the two-dimensional quadrature is

$$\int_{-1}^{1} \int_{-1}^{1} w(\xi,\eta)\,\mathrm{d}\xi\mathrm{d}\eta \approx \sum_{i=0}^{N} \sum_{j=0}^{N} w(\xi_i, \eta_j)\omega_i\omega_j. \tag{2.12}$$

We choose the test function ϕ in (2.6)-(2.7) to be a polynomial in the reference element E

$$\phi^E = \sum_{i=0}^{N} \sum_{j=0}^{N} \phi_{i,j}^E \ell_i(\xi)\ell_j(\eta). \tag{2.13}$$

This choice of test functions and the collocation of the interpolation and quadrature nodes enables us to use the Kronecker delta property of the Lagrange interpolating polynomials to greatly simplify the integrals in (2.6). For example, the integral on the time derivative term becomes

$$\int_E \mathcal{J} W_t \ell_i(\xi)\ell_j(\eta)\,\mathrm{dE} \approx \mathcal{J}_{ij}(W_t)_{ij}\omega_i\omega_j. \tag{2.14}$$

A significant advantage of the LGL quadrature nodes is that the corresponding derivative operator $\mathbf{D}$ (2.11) and the mass matrix $\mathbf{M}$ satisfy the summation-by-parts (SBP) property,

$$\mathbf{MD} + (\mathbf{MD})^T = \mathrm{diag}(-1, 0, \ldots, 0, 1), \tag{2.15}$$

for all polynomial orders [14]. The SBP property enables us to make a connection between LGL-based discontinuous Galerkin methods and sub-cell finite volume type differencing methods as proposed by Fisher and Carpenter [11]. In [46] we used this relation to develop an entropy stable discontinuous Galerkin spectral element method for the shallow water equations on curvilinear geometries. The SBP property (2.15) also implies $\mathbf{D} = -\mathbf{S} + \hat{\mathbf{D}}$ with surface matrix $\mathbf{S} := \mathrm{diag}\left(\frac{1}{\omega_0}, 0, \ldots, 0, -\frac{1}{\omega_N}\right)$ and $\hat{\mathbf{D}} := -\mathbf{M}^{-1}\mathbf{D}\mathbf{M}$. It also follows that strong and weak form discretizations are equivalent [25]. We rewrite the derivative operator $\mathbf{D}$ in the computations to incorporate the surface parts and a factor of 2 for the interior that stems from the split form approach [16]

$$\tilde{\mathbf{D}} := 2\mathbf{D} + \mathbf{S}. \tag{2.16}$$

The full discretization as well as a summary of the main properties of the ESDGSEM can be found in Lemma 1. We refer to [46] for the detailed proofs and derivations. As a notational convenience we introduce jump $[\![\cdot]\!]$ and average $\{\{\cdot\}\}$ operators which are defined by

$$\begin{aligned} [\![W]\!] &:= W^+ - W^-, \\ \{\{W\}\} &:= \frac{1}{2}\left(W^+ + W^-\right). \end{aligned} \tag{2.17}$$

We note that jumps only occur on element interfaces and have an orientation. The "+" and "−" states here are strictly related to the normal vector on the interface and the normal is always pointing outward from the "−" element and into the "+" element. The averages are also used for the evaluation of the interior two point fluxes and have no orientation, but require special compact notation. For example, in the $\xi-$direction, the interior arithmetic mean is

$$\{\{\cdot\}\}_{(i,m),j} = \frac{1}{2}\left((\cdot)_{ij} + (\cdot)_{mj}\right). \tag{2.18}$$

Lemma 1 (ESDGSEM)**.** *The semi-discrete split DG approximation to the two dimensional shallow water equations on curvilinear grids*

$$J\vec{W}_t + \vec{\mathcal{L}}_\xi + \vec{\mathcal{L}}_\eta = \vec{\tilde{S}} \tag{2.19}$$

with

$$\begin{aligned} \left(\vec{\mathcal{L}}_\xi\right)_{ij} &= \frac{1}{\omega_i}\left(\delta_{iN}\left[\vec{\tilde{F}}^{*,es}\right]_{Nj} - \delta_{i0}\left[\vec{\tilde{F}}^{*,es}\right]_{0j}\right) + \sum_{m=0}^{N} \tilde{D}_{im}\vec{\tilde{F}}_{(i,m),j}, \\ \left(\vec{\mathcal{L}}_\eta\right)_{ij} &= \frac{1}{\omega_j}\left(\delta_{Nj}\left[\vec{\tilde{G}}^{*,es}\right]_{iN} - \delta_{0j}\left[\vec{\tilde{G}}^{*,es}\right]_{i0}\right) + \sum_{m=0}^{N} \tilde{D}_{jm}\vec{\tilde{G}}_{i,(m,j)}, \end{aligned} \tag{2.20}$$

with curvilinear volume fluxes

$$\begin{aligned} \vec{\tilde{F}}_{(i,m),j} &:= \vec{F}^{\#}(\vec{W}_{i,j}, \vec{W}_{m,j})\,\{\{y_\eta\}\}_{(i,m),j} - \vec{G}^{\#}(\vec{W}_{i,j}, \vec{W}_{m,j})\,\{\{x_\eta\}\}_{(i,m),j}, \\ \vec{\tilde{G}}_{i,(m,j)} &:= -\vec{F}^{\#}(\vec{W}_{i,j}, \vec{W}_{i,m})\,\{\{x_\xi\}\}_{i,(j,m)} + \vec{G}^{\#}(\vec{W}_{i,j}, \vec{W}_{i,m})\,\{\{y_\xi\}\}_{i,(j,m)}, \end{aligned} \tag{2.21}$$

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

$$\begin{aligned}
\vec{F}^{\#}(\vec{W}_{i,j},\vec{W}_{m,j}) &:= \begin{pmatrix} \{\{hu\}\}_{(i,m),j} \\ \{\{hu\}\}_{(i,m),j}\,\{\{u\}\}_{(i,m),j} + g\,\{\{h\}\}^2_{(i,m),j} - \frac{1}{2}g\,\{\{h^2\}\}_{(i,m),j} \\ \{\{hu\}\}_{(i,m),j}\,\{\{v\}\}_{(i,m),j} \end{pmatrix}, \\
\vec{G}^{\#}(\vec{W}_{i,j},\vec{W}_{i,m}) &:= \begin{pmatrix} \{\{hv\}\}_{i,(m,j)} \\ \{\{hv\}\}_{i,(m,j)}\,\{\{u\}\}_{i,(m,j)} \\ \{\{hv\}\}_{i,(m,j)}\,\{\{v\}\}_{i,(m,j)} + g\,\{\{h\}\}^2_{i,(m,j)} - \frac{1}{2}g\,\{\{h^2\}\}_{i,(m,j)} \end{pmatrix}.
\end{aligned} \tag{2.22}$$

The numerical interface flux in normal direction is given by

$$\vec{\tilde{F}}^{*,es} = n_x\vec{F}^{*,es} + n_y\vec{G}^{*,es}, \tag{2.23}$$

and is a combination of the fluxes in x and y direction

$$\begin{aligned}
\vec{F}^{*,es} &= \vec{F}^{*,ec} - \frac{1}{2}\mathbf{R}_f\left|\mathbf{\Lambda}_f\right|\mathbf{R}_f^T[\![\vec{q}]\!], \\
\vec{G}^{*,es} &= \vec{G}^{*,ec} - \frac{1}{2}\mathbf{R}_g\left|\mathbf{\Lambda}_g\right|\mathbf{R}_g^T[\![\vec{q}]\!],
\end{aligned} \tag{2.24}$$

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

$$\begin{aligned}
\left|\mathbf{\Lambda}_f\right| &= \frac{1}{2g}\begin{pmatrix} |\,\{\{u\}\}+\{\{c\}\}\,| & 0 & 0 \\ 0 & 2g|\,\{\{h\}\}\,\{\{u\}\}\,| & 0 \\ 0 & 0 & |\,\{\{u\}\}-\{\{c\}\}\,| \end{pmatrix}, \\
\left|\mathbf{\Lambda}_g\right| &= \frac{1}{2g}\begin{pmatrix} |\,\{\{v\}\}+\{\{c\}\}\,| & 0 & 0 \\ 0 & 2g|\,\{\{h\}\}\,\{\{v\}\}\,| & 0 \\ 0 & 0 & |\,\{\{v\}\}-\{\{c\}\}\,| \end{pmatrix},
\end{aligned} \tag{2.25}$$

and eigenvectors

$$\begin{aligned}
\mathbf{R}_f &= \begin{pmatrix} 1 & 0 & 1 \\ \{\{u\}\}+\{\{c\}\} & 0 & \{\{u\}\}-\{\{c\}\} \\ \{\{v\}\} & 1 & \{\{v\}\} \end{pmatrix}, \\
\mathbf{R}_g &= \begin{pmatrix} 1 & 0 & 1 \\ \{\{u\}\} & 1 & \{\{u\}\} \\ \{\{v\}\}+\{\{c\}\} & 0 & \{\{v\}\}-\{\{c\}\} \end{pmatrix},
\end{aligned} \tag{2.26}$$

and the jump in entropy variables $[\![\vec{q}]\!]$ and wave speed $c=\sqrt{gh}$. The entropy conserving interface fluxes are

$$\begin{aligned}
\vec{F}^{*,ec}(\vec{W}^+,\vec{W}^-) &:= \begin{pmatrix} \{\{h\}\}\,\{\{u\}\} \\ \{\{h\}\}\,\{\{u\}\}^2 + \frac{1}{2}\,g\,\{\{h^2\}\} \\ \{\{h\}\}\,\{\{u\}\}\,\{\{v\}\} \end{pmatrix}, \\
\vec{G}^{*,ec}(\vec{W}^+,\vec{W}^-) &:= \begin{pmatrix} \{\{h\}\}\,\{\{v\}\} \\ \{\{h\}\}\,\{\{u\}\}\,\{\{v\}\} \\ \{\{h\}\}\,\{\{v\}\}^2 + \frac{1}{2}\,g\,\{\{h^2\}\} \end{pmatrix},
\end{aligned} \tag{2.27}$$

and the components of the dissipation term can be found in Appendix B. The source term discretization $\vec{\tilde{\mathcal{S}}}$ is defined by

$$\begin{aligned}
(\tilde{\mathcal{S}}_1)_{ij} &:= 0, \\
(\tilde{\mathcal{S}}_2)_{ij} &:= \frac{g}{2}\left(-hb_x + \mathcal{S}_{i,j}^{x,*}\right), \\
(\tilde{\mathcal{S}}_3)_{ij} &:= \frac{g}{2}\left(-hb_y + \mathcal{S}_{i,j}^{y,*}\right),
\end{aligned} \tag{2.28}$$

where the additional interface penalty terms $\mathcal{S}^{\cdot,*}$ that account for possibly discontinuous bottom topographies can be found in [46] and the derivatives of the bottom topography are discretized in split form

$$\begin{aligned}(b_x)_{ij} &= (y_\eta)_{ij} \sum_{m=0}^{N} 2\, D_{im} b_{mj} + \sum_{m=0}^{N} 2\, D_{im} (y_\eta b)_{mj} - (y_\xi)_{ij} \sum_{m=0}^{N} 2\, D_{jm} b_{im} - \sum_{m=0}^{N} 2\, D_{jm} (y_\xi b)_{im}, \\ (b_y)_{ij} &= (x_\eta)_{ij} \sum_{m=0}^{N} 2\, D_{im} b_{mj} + \sum_{m=0}^{N} 2\, D_{im} (x_\eta b)_{mj} - (x_\xi)_{ij} \sum_{m=0}^{N} 2\, D_{jm} b_{im} - \sum_{m=0}^{N} 2\, D_{jm} (x_\xi b)_{im}.\end{aligned} \tag{2.29}$$

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

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

1.2 Guaranteed dissipation of the total discrete energy, which is an entropy function for the shallow water equations. Hence it fulfills the discrete entropy inequality (1.7)*.*

1.3 Discrete well-balanced property for arbitrary bottom topographies.

Proof. See [46]. ■

The semi-discrete ESDGSEM is only complete when equipped with a suitable time integrator. We previously used a low storage Runge-Kutta scheme in [46] but have now switched to a strong stability preserving Runge-Kutta scheme (SSPRK). This choice is necessary for the positivity preservation proof in Section 4. We use the SSPRK method that is frequently used in wet/dry shallow water schemes as in [50] and shock capturing schemes as in [39]. The three stage scheme is

$$\begin{aligned} W^{(1)} &= W^n + \Delta t \mathcal{R}\left(W^n\right), \\ W^{(2)} &= \frac{3}{4} W^n + \frac{1}{4}\left(W^{(1)} + \Delta t \mathcal{R}\left(W^{(1)}\right)\right), \\ W^{n+1} &= \frac{1}{3} W^n + \frac{2}{3}\left(W^{(2)} + \Delta t \mathcal{R}\left(W^{(2)}\right)\right), \end{aligned} \tag{2.30}$$

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

$$\mathcal{R} = -\frac{1}{J}\left(\vec{\mathcal{L}}_\xi + \vec{\mathcal{L}}_\eta - \vec{\mathcal{S}}\right). \tag{2.31}$$

3. Artificial Viscosity

The entropy stable scheme is more robust compared to a standard DGSEM for the shallow water equations [17, 46], but it is not oscillation free in the presence of shocks. These oscillations might lead to unphysical solutions and cause simulations to crash if the water height becomes negative. We discuss the addition of a positivity limiter in the next section. Usually, such positivity preserving limiters are coupled with some form of TVB limiter [52] or artificial viscosity [28] to keep oscillations in check. Our main requirement for the additional smoothing or limiting is to preserve the entropy stability of the scheme. In [15] the authors prove that under certain conditions, adding artificial viscosity to the scheme maintains the entropy stability. We choose the gradient variables carefully and use the discretization by Bassi and Rebay [1] to fulfill these conditions analogous to the strategy in [15]. For our shock capturing, we only add artificial viscosity to the momentum equations. While the smoothing is weaker without direct artificial viscosity in the continuity equation, we trivially maintain conservation of mass and have a straight forward mapping to the entropy variables, which leads to an easy proof of the requirements for the entropy stability via a theorem from [15]. In fact, we choose the entropy variables $q_1 = u$ and $q_2 = v$ as gradient variables, leading to a simple one-to-one mapping from gradient variables to entropy variables. We scale the gradient by the water height

h and a viscosity parameter ϵ to obtain the viscous fluxes. The viscosity parameter is dynamically computed for each element and is based on a smoothness measure of the water height within the element. We give details on the computation of the viscosity parameter in Appendix A. The modified shallow water equations with artificial viscosity are

$$\begin{aligned} h_t + (hu)_x + (hv)_y &= 0, \\ (hu)_t + (h\,u^2 + g\,h^2/2)_x + (huv)_y &= -g\,h\,b_x + \nabla \cdot \left(h\,\epsilon\,\vec{\mathcal{U}}\right), \\ (hv)_t + (huv)_x + (h\,v^2 + g\,h^2/2)_y &= -g\,h\,b_y + \nabla \cdot \left(h\,\epsilon\,\vec{\mathcal{V}}\right), \\ \vec{\mathcal{U}} &= \nabla u, \\ \vec{\mathcal{V}} &= \nabla v, \end{aligned} \tag{3.1}$$

or in compact flux form

$$\begin{aligned} \vec{w}_t + \nabla \cdot (\vec{f}, \vec{g})^T &= \vec{\mathcal{S}} + \nabla \cdot (\vec{f}^v, \vec{g}^v)^T, \\ \vec{\mathcal{U}} &= \nabla u, \\ \vec{\mathcal{V}} &= \nabla v, \end{aligned} \tag{3.2}$$

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 and get

$$\begin{aligned} \mathcal{J}\vec{w}_t &= -\hat{\nabla} \cdot (\vec{\tilde{f}} + \vec{\tilde{f}}^v, \vec{\tilde{g}} + \vec{\tilde{g}}^v)^T + \vec{\tilde{\mathcal{S}}}, \\ \mathcal{J}\vec{\mathcal{U}} &= \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix} \hat{\nabla} u, \\ \mathcal{J}\vec{\mathcal{V}} &= \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix} \hat{\nabla} v, \end{aligned} \tag{3.3}$$

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

Analogous steps as in Sec. 2 are then used to find the weak and strong form and their discretizations for the continuity and momentum equations. We multiply the transformed gradient equations by the same test function ϕ and integrate over the domain to find the weak form

$$\begin{aligned} \int_E \mathcal{J}\,\vec{\mathcal{U}}\,\phi\,\mathrm{dE} - \oint_{\partial E} \phi\,U^*\,\vec{n}\,\mathrm{dS} + \int_E u \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix} \hat{\nabla}\phi\,\mathrm{dE} = 0, \\ \int_E \mathcal{J}\,\vec{\mathcal{V}}\,\phi\,\mathrm{dE} - \oint_{\partial E} \phi\,V^*\,\vec{n}\,\mathrm{dS} + \int_E v \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix} \hat{\nabla}\phi\,\mathrm{dE} = 0, \end{aligned} \tag{3.4}$$

where U^* and V^* are the numerical interface states for the gradient equations. We discretize the weak form gradient equations (3.4) and use the Lagrange property to simplify. For example, the volume integral approximations for the $\vec{\mathcal{U}}$ equations are

$$\begin{aligned} \int_{-1}^{1}\int_{-1}^{1} u \left(y_\eta \frac{\partial}{\partial\xi}(\ell_i(\xi)\ell_j(\eta)) - y_\xi \frac{\partial}{\partial\eta}(\ell_i(\xi)\ell_j(\eta))\right) \mathrm{d}\xi\mathrm{d}\eta &\approx -\sum_{m=0}^{N} \hat{D}_{im} u_{mj} y_{\eta_{mj}}\,\omega_i\omega_j + \sum_{n=0}^{N} \hat{D}_{jn} u_{in} y_{\xi_{in}}\,\omega_i\omega_j, \\ \int_{-1}^{1}\int_{-1}^{1} u \left(-x_\eta \frac{\partial}{\partial\xi}(\ell_i(\xi)\ell_j(\eta)) + x_\xi \frac{\partial}{\partial\eta}(\ell_i(\xi)\ell_j(\eta))\right) \mathrm{d}\xi\mathrm{d}\eta &\approx \sum_{m=0}^{N} \hat{D}_{im} u_{mj} x_{\eta_{mj}}\,\omega_i\omega_j - \sum_{n=0}^{N} \hat{D}_{jn} u_{in} x_{\xi_{in}}\,\omega_i\omega_j. \end{aligned} \tag{3.5}$$

Similarly, the surface integrals for $\vec{\mathcal{U}}$ are approximated by

$$\begin{aligned}
\oint_{\partial E} \phi\, U^*\, \vec{n}_1 \,\mathrm{dS} &\approx -\,\delta_{i0}\, U^*_{i0} y_{\eta i0}\omega_j + \delta_{iN}\, U^*_{iN} y_{\eta i0}\omega_j - \delta_{j0}\, U^*_{0j} y_{\xi 0j}\omega_i + \delta_{jN}\, U^*_{Nj} y_{\xi Nj}\omega_i,\\
\oint_{\partial E} \phi\, U^*\, \vec{n}_2 \,\mathrm{dS} &\approx -\,\delta_{i0}\, U^*_{i0} x_{\eta i0}\omega_j + \delta_{iN}\, U^*_{iN} x_{\eta i0}\omega_j - \delta_{j0}\, U^*_{0j} x_{\xi 0j}\omega_i + \delta_{jN}\, U^*_{Nj} x_{\xi Nj}\omega_i.
\end{aligned} \tag{3.6}$$

We summarize the ESDGSEM with artificial viscosity and state the full discretizations of all the new viscous terms and gradient equations in Theorem 1 and proof that the resulting method is entropy stable.

Theorem 1 (Entropy stability of ESDGSEM with artificial viscosity)**.** *The ESDGSEM* (2.19) *with additional viscous terms as in* (3.1) *is*

$$J\vec{W}_t + \vec{\mathcal{L}}_\xi + \vec{\mathcal{L}}_\eta = \vec{\tilde{\mathcal{S}}} + \vec{\mathcal{L}}^v_\xi + \vec{\mathcal{L}}^v_\eta. \tag{3.7}$$

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

$$\begin{aligned}
(\vec{\mathcal{L}}^v_\xi)_{ij} &= \sum_{m=0}^{N} D_{im}(\vec{\tilde{F}}^v)_{mj} + \frac{1}{\omega_i}\left(\delta_{iN}\vec{\tilde{F}}^{v,*}_{Nj} - \delta_{i0}\vec{\tilde{F}}^{v,*}_{0j}\right) - \frac{1}{\omega_i}\left(\delta_{iN}\vec{\tilde{F}}^{v}_{Nj} - \delta_{i0}\vec{\tilde{F}}^{v}_{0j}\right),\\
(\vec{\mathcal{L}}^v_\eta)_{ij} &= \sum_{m=0}^{N} D_{jm}(\vec{\tilde{G}}^v)_{im} + \frac{1}{\omega_j}\left(\delta_{Nj}\vec{\tilde{G}}^{v,*}_{iN} - \delta_{0j}\vec{\tilde{G}}^{v,*}_{i0}\right) - \frac{1}{\omega_j}\left(\delta_{Nj}\vec{\tilde{G}}^{v}_{iN} - \delta_{0j}\vec{\tilde{G}}^{v}_{i0}\right).
\end{aligned} \tag{3.8}$$

The curvilinear viscous fluxes are defined with the BR1 approach as

$$\begin{aligned}
\vec{\tilde{F}}^v &= \quad y_\eta \vec{F}^v - x_\eta \vec{G}^v,\\
\vec{\tilde{G}}^v &= -y_\xi \vec{F}^v + x_\xi \vec{G}^v,
\end{aligned} \tag{3.9}$$

with viscous fluxes

$$\begin{aligned}
\vec{F}^v &= h\,\epsilon\,(\mathcal{U}_1,\,\mathcal{V}_1)^T,\\
\vec{G}^v &= h\,\epsilon\,(\mathcal{U}_2,\,\mathcal{V}_2)^T,
\end{aligned} \tag{3.10}$$

and the viscous flux interface coupling is computed with

$$\begin{aligned}
\vec{\tilde{F}}^{v,*} &= \left\{\!\left\{\vec{\tilde{F}}^v\right\}\!\right\},\\
\vec{\tilde{G}}^{v,*} &= \left\{\!\left\{\vec{\tilde{G}}^v\right\}\!\right\}.
\end{aligned} \tag{3.11}$$

The gradients $\vec{\mathcal{U}} = \nabla u$ and $\vec{\mathcal{V}} = \nabla v$ are computed by

$$\begin{aligned}
(\mathcal{U}_1)_{ij} &= (y_\eta)_{ij}\sum_{m=0}^{N}\hat{D}_{im}u_{mj} - (y_\xi)_{ij}\sum_{m=0}^{N}\hat{D}_{jm}u_{im} + \frac{(y_\eta)_{ij}}{\omega_i}\left(\delta_{iN}U^*_{Nj} - \delta_{i0}U^*_{0j}\right) + \frac{(y_\xi)_{ij}}{\omega_j}\left(\delta_{Nj}U^*_{iN} - \delta_{0j}U^*_{i0}\right)\\
(\mathcal{U}_2)_{ij} &= -(x_\eta)_{ij}\sum_{m=0}^{N}\hat{D}_{im}u_{mj} + (x_\xi)_{ij}\sum_{m=0}^{N}\hat{D}_{jm}u_{im} + \frac{(x_\eta)_{ij}}{\omega_i}\left(\delta_{iN}U^*_{Nj} - \delta_{i0}U^*_{0j}\right) + \frac{(x_\xi)_{ij}}{\omega_j}\left(\delta_{Nj}U^*_{iN} - \delta_{0j}U^*_{i0}\right)\\
(\mathcal{V}_1)_{ij} &= (y_\eta)_{ij}\sum_{m=0}^{N}\hat{D}_{im}v_{mj} - (y_\xi)_{ij}\sum_{m=0}^{N}\hat{D}_{jm}v_{im} + \frac{(y_\eta)_{ij}}{\omega_i}\left(\delta_{iN}V^*_{Nj} - \delta_{i0}V^*_{0j}\right) + \frac{(y_\xi)_{ij}}{\omega_j}\left(\delta_{Nj}V^*_{iN} - \delta_{0j}V^*_{i0}\right)\\
(\mathcal{V}_2)_{ij} &= -(x_\eta)_{ij}\sum_{m=0}^{N}\hat{D}_{im}v_{mj} + (x_\xi)_{ij}\sum_{m=0}^{N}\hat{D}_{jm}v_{im} + \frac{(x_\eta)_{ij}}{\omega_i}\left(\delta_{iN}V^*_{Nj} - \delta_{i0}V^*_{0j}\right) + \frac{(x_\xi)_{ij}}{\omega_j}\left(\delta_{Nj}V^*_{iN} - \delta_{0j}V^*_{i0}\right),
\end{aligned} \tag{3.12}$$

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

$$\begin{aligned}
U^* &= \{\{u\}\},\\
V^* &= \{\{v\}\}.
\end{aligned} \tag{3.13}$$

The ESDGSEM with artificial viscosity (3.7) *and viscous terms discretized as in* (3.8) *and* (3.12)*, is entropy stable.*

Proof. In [15] the authors show that viscous terms discretized by Bassi and Rebay [1] 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

$$\overset{\leftrightarrow}{f}{}^v(\vec{w}, \nabla\vec{w}) = \mathcal{B}^\epsilon \nabla\vec{q}, \tag{3.14}$$

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

$$\mathcal{B}^\epsilon = \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{pmatrix} \in \mathbb{R}^{6\times 6}. \tag{3.15}$$

The entropy stability requirements for the block matrix $\mathcal{B}^\epsilon$ are then that each block $\mathbf{B}_{ij}$ is symmetric

$$\mathbf{B}^\epsilon_{ij} = \left(\mathbf{B}^\epsilon_{ji}\right)^T, \tag{3.16}$$

and positive (semi-)definite

$$\sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial\vec{q}}{\partial x_i}^T \mathbf{B}^\epsilon_{ij} \frac{\partial\vec{q}}{\partial x_j} \geq 0, \qquad \forall\vec{q}. \tag{3.17}$$

The choice of gradient variables and fluxes in (3.1) make the block matrix very simple in this case

$$\mathcal{B}^\epsilon = \epsilon h \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.18}$$

We check the requirement (3.17) to find

$$\begin{aligned} \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{\partial\vec{q}}{\partial x_i}^T \mathbf{B}^\epsilon_{ij} \frac{\partial\vec{q}}{\partial x_j} &= (\vec{q}_x)^T \mathbf{B}^\epsilon_{11}\, \vec{q}_x + (\vec{q}_y)^T \mathbf{B}^\epsilon_{22}\, \vec{q}_y \\ &= \epsilon\, h \left(u_x^2 + v_x^2 + u_y^2 + v_y^2\right) \geq 0. \end{aligned} \tag{3.19}$$

■

The artificial viscosity reduces the amount and magnitude of oscillations as shown in the numerical results section, specifically in a dam break example in Section 6.5.

4. Positivity Preserving Limiter

The artificial viscosity from Sec. 3 greatly reduces the oscillations, but in cases with wet/dry regions, this is sometimes not enough as even small oscillations may render a numerical scheme unstable if they lead to negative water heights. A mechanism is needed, that strictly enforces the positivity of the water height h without destroying accuracy, conservation, well-balancedness or entropy-stability of the ESDGSEM. In, e.g., [35, 49, 50, 52] the authors have developed a positivity preserving limiter and applied it to the shallow water equation. The limiter is based on a linear scaling around element averages and thus relies on non-negative average water heights in all elements. It is then proved that non-negative average water heights are guaranteed to be preserved for an Euler time step. The proof relies on the specific choice of the numerical

flux. Numerical fluxes that preserve non-negative water heights for finite volume schemes are called positivity preserving [52]. For Cartesian meshes, this property directly translates to DG methods by way of selecting any positivity preserving numerical interface fluxes, e.g. [50]. In [52] it is proven that the Lax-Friedrichs numerical flux [35] is positivity preserving and the same is noted for the Godunov flux [10], Boltzmann type flux [34] and the Harten-Lax-van-Leer flux [18]. The positivity preserving property is shown for Euler time integration but naturally extends to SSPRK methods, e.g. (2.30), as these are convex combinations of Euler time steps, see [38] for details.

In this section we prove that the entropy stable numerical flux of the ESDGSEM (2.23) 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 [52], where the update of the mean water height is written in finite volume form. Due to generally different Jacobians of the curvilinear mappings and possible changing normals on opposing sides this is not as straightforward on curved meshes. We directly prove the positivity preservation for curved quadrilateral meshes in Lemma 2. Also, we verify that the positivity preserving limiter is entropy stable in Lemma 3. Both results are then summarized in Theorem 2.

We will now show that the ESDGSEM with Euler time integration preserves a non-negative water height for a sufficiently small time step. For notational convenience we use the notation $W_{j,s}$ (opposed to W_{ij} for internal nodal values) to denote the value of W at node j on interface s, where $s \in \{1, \dots, 4\}$ denotes the element local interface 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

$$\begin{aligned} \mathcal{J}^{\text{surf}} &:= \sqrt{y_\xi y_\xi + x_\xi x_\xi}, \qquad \text{for } \eta = \pm 1 \ , \\ \mathcal{J}^{\text{surf}} &:= \sqrt{y_\eta y_\eta + x_\eta x_\eta}, \qquad \text{for } \xi = \pm 1 \ . \end{aligned} \tag{4.1}$$

Lemma 2 (Preservation of non-negative mean water heights in ESDGSEM)**.** *If the water height, h, is non-negative for all LGL nodes then the average water height in the next time step is non-negative for all elements under the additional time step restrictions*

$$\begin{aligned} \Delta t &\leq \frac{\omega_0 \, a_{j,s}}{\left(A_{j,s} + 2 \left\{ \{ \tilde{u} \} \right\}_{j,s} \right)}, \\ \Delta t &\leq \left| \frac{\omega_0 \, a_{j,s} \, g \, h_{j,s}}{\left\{ \{ c \} \right\}_{j,s} B_{j,s} \left[\! \left[\tilde{u} \right] \! \right]_{j,s}} \right|, \qquad \textit{only if } h_{j,s} > 0, \end{aligned} \tag{4.2}$$

where we introduce the rotated normal velocity

$$\tilde{u} := n_x u + n_y v \tag{4.3}$$

for all edge nodes $j = 0, \dots, N$ on all element sides $s = 1, \dots, 4$, and

$$\begin{aligned} A &:= \left| \left\{ \{ \tilde{u} \} \right\} + \left\{ \{ c \} \right\} \right| + \left| \left\{ \{ \tilde{u} \} \right\} - \left\{ \{ c \} \right\} \right|, \\ B &:= \left| \left\{ \{ \tilde{u} \} \right\} + \left\{ \{ c \} \right\} \right| - \left| \left\{ \{ \tilde{u} \} \right\} - \left\{ \{ c \} \right\} \right|. \end{aligned} \tag{4.4}$$

For the Legendre Gauss Lobatto nodes, the quadrature weight ω_0 is given by $\omega_0 = \frac{1}{2} N(N-1)$. We also have geometric scaling factors on the interfaces given by $a_{j,s} := \frac{\mathcal{J}_{j,s}}{\mathcal{J}^{surf}_{j,s}}$ with volume and surface Jacobians defined in (2.1) *and* (4.1)*.*

Proof. We first examine the numerical flux. Since the shallow water equations are rotationally invariant, we can calculate the numerical flux in normal direction and then rotate back. We want to guarantee positive water height and, thus, proceed to examine the numerical flux contributions for the water height equation. The first entry of the entropy stable numerical flux in the normal direction can be simplified to

$$F_1^{*,es}(\vec{\tilde{W}}^+, \vec{\tilde{W}}^-) = \left\{ \{ h \} \right\} \left\{ \{ \tilde{u} \} \right\} - \frac{1}{4g} \left(A \left[\! \left[gh + gb \right] \! \right] + \left\{ \{ c \} \right\} B \left[\! \left[\tilde{u} \right] \! \right] \right), \tag{4.5}$$

where $\tilde{u} = n_x u + n_y v$ and full details on the derivation of (4.5) are given in Appendix B. As the ESDGSEM is a conservative numerical scheme, we can write the update of the element average water height in one Euler time step as

$$\begin{aligned}\overline{h}^{t_{n+1}} &= \overline{h}^{t_n} - \frac{\Delta t}{|E|}\sum_{s=1}^{4}\sum_{j=0}^{N}\omega_j \mathcal{J}_{j,s}^{\text{surf}} \tilde{F}_1^{*,es}\left(W_{j,s}^{\text{int}}, W_{j,s}^{\text{ext}}, n_{j,s}\right) \\ &= \overline{h}^{t_n} - \frac{\Delta t}{|E|}\sum_{s=1}^{4}\sum_{j=0}^{N}\omega_j \mathcal{J}_{j,s}^{\text{surf}} F_1^{*,es}\left(\tilde{W}_{j,s}^{\text{int}}, \tilde{W}_{j,s}^{\text{ext}}\right),\end{aligned} \tag{4.6}$$

where $\mathcal{J}_{j,s}^{\text{surf}}$ is the surface Jacobian at node j on interface s defined in (4.1). We can also write the average water height $\overline{h}^{t_n}$ as

$$\begin{aligned}\overline{h}^{t_n} &= \frac{1}{|E|}\sum_{j=0}^{N}\sum_{i=0}^{N} h_{ij}\mathcal{J}_{ij}\omega_i\omega_j \\ &= \frac{1}{|E|}\sum_{i=1}^{N-1}\sum_{j=1}^{N-1} h_{ij}\mathcal{J}_{ij}\omega_i\omega_j + \frac{1}{2}\frac{1}{|E|}\sum_{s=1}^{4}\sum_{j=1}^{N-1} h_{j,s}\mathcal{J}_{j,s}\omega_0\omega_j + \frac{1}{2}\frac{1}{|E|}\sum_{s=1}^{4}\sum_{j=0}^{N} h_{j,s}\mathcal{J}_{j,s}\omega_0\omega_j \\ &= \frac{1}{|E|}\sum_{i=1}^{N-1}\sum_{j=1}^{N-1} h_{ij}\mathcal{J}_{ij}\omega_i\omega_j + \frac{1}{2}\frac{1}{|E|}\sum_{s=1}^{4}\sum_{j=1}^{N-1} h_{j,s}\mathcal{J}_{j,s}\omega_0\omega_j \\ &\quad + \frac{1}{2|E|}\sum_{j=0}^{N} h_{j,3}\mathcal{J}_{j,3}\omega_0\omega_j + \frac{1}{2|E|}\sum_{j=0}^{N} h_{j,1}\mathcal{J}_{j,1}\omega_0\omega_j \\ &\quad + \frac{1}{2|E|}\sum_{i=0}^{N} h_{j,4}\mathcal{J}_{j,4}\omega_0\omega_j + \frac{1}{2|E|}\sum_{i=0}^{N} h_{j,2}\mathcal{J}_{j,2}\omega_0\omega_j.\end{aligned} \tag{4.7}$$

Inserting the new expression for the average water height (4.7) into the update scheme (4.6) we find

$$\begin{aligned}\overline{h}^{t_{n+1}} &= \frac{1}{|E|}\sum_{i=1}^{N-1}\sum_{j=1}^{N-1} h_{ij}\mathcal{J}_{ij}\omega_i\omega_j + \frac{1}{2}\frac{1}{|E|}\sum_{s=1}^{4}\sum_{j=1}^{N-1} h_{j,s}\mathcal{J}_{j,s}\omega_0\omega_j \\ &\quad + \frac{1}{|E|}\sum_{j=0}^{N}\mathcal{J}_{j,1}\omega_0\omega_j\left[\frac{1}{2}h_{j,1} - \frac{\Delta t}{\omega_0\, a_{j,1}}F_1^{*,es}\left(\tilde{W}_{j,1}^{\text{int}}, \tilde{W}_{j,1}^{\text{ext}}\right)\right] \\ &\quad + \frac{1}{|E|}\sum_{j=0}^{N}\mathcal{J}_{j,3}\omega_0\omega_j\left[\frac{1}{2}h_{j,3} - \frac{\Delta t}{\omega_0\, a_{j,3}}F_1^{*,es}\left(\tilde{W}_{j,3}^{\text{int}}, \tilde{W}_{j,3}^{\text{ext}}\right)\right] \\ &\quad + \frac{1}{|E|}\sum_{j=0}^{N}\mathcal{J}_{j,2}\omega_0\omega_j\left[\frac{1}{2}h_{j,2} - \frac{\Delta t}{\omega_0\, a_{j,2}}F_1^{*,es}\left(\tilde{W}_{j,2}^{\text{int}}, \tilde{W}_{j,2}^{\text{ext}}\right)\right] \\ &\quad + \frac{1}{|E|}\sum_{j=0}^{N}\mathcal{J}_{j,4}\omega_0\omega_j\left[\frac{1}{2}h_{j,4} - \frac{\Delta t}{\omega_0\, a_{j,4}}F_1^{*,es}\left(\tilde{W}_{j,4}^{\text{int}}, \tilde{W}_{j,4}^{\text{ext}}\right)\right],\end{aligned} \tag{4.8}$$

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

$$\frac{1}{2}h_{j,s} - \frac{\Delta t}{\omega_0\, a_{j,s}}F_1^{*,es}\left(\tilde{W}_{j,s}^{\text{int}}, \tilde{W}_{j,s}^{\text{ext}}\right), \qquad s = 1, \ldots, 4 \quad . \tag{4.9}$$

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 (4.5) and assuming continuous bottom topographies across element interfaces we find

$$\begin{aligned}
&\frac{1}{2}h^- - \frac{1}{4g}\frac{\Delta t}{\omega_0\, a}\left(4g\,\{\{h\}\}\,\{\{\tilde{u}\}\} - gA\,[\![h]\!] - \{\{c\}\}\,B\,[\![\tilde{u}]\!]\right)\\
&= \frac{1}{2}h^- - \frac{1}{4g}\frac{\Delta t}{\omega_0\, a}\left(gh^+\left(2\,\{\{\tilde{u}\}\} - A\right) + gh^-\left(2\,\{\{\tilde{u}\}\} + A\right) - \{\{c\}\}\,B\,[\![\tilde{u}]\!]\right) \qquad (4.10)\\
&= \frac{1}{4}\frac{\Delta t}{\omega_0\, a}h^+\left(A - 2\,\{\{\tilde{u}\}\}\right) + \frac{1}{4}h^-\left(1 - \frac{\Delta t}{\omega_0\, a}\left(A + 2\,\{\{\tilde{u}\}\}\right)\right) + \frac{1}{4}\left(h^- + \frac{\Delta t}{g\,\omega_0\, a}\,\{\{c\}\}\,B\,[\![\tilde{u}]\!]\right)
\end{aligned}$$

We examine the three terms in (4.10) individually. The first term is always non-negative, since

$$A = \left|\,\{\{\tilde{u}\}\} + \{\{c\}\}\,\right| + \left|\,\{\{\tilde{u}\}\} - \{\{c\}\}\,\right| \geq 2\left|\,\{\{\tilde{u}\}\}\,\right|. \qquad (4.11)$$

For the second term we require an additional time step condition. From

$$\left(1 - \frac{\Delta t}{\omega_0\, a}\left(A + 2\,\{\{\tilde{u}\}\}\right)\right) \overset{!}{\geq} 0, \qquad (4.12)$$

we find

$$\Delta t \overset{!}{\leq} \frac{\omega_0\, a}{\left(A + 2\,\{\{\tilde{u}\}\}\right)}. \qquad (4.13)$$

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 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

$$\frac{\Delta t}{g\,\omega_0\, a}\,\{\{c\}\}\,B\tilde{u}^+. \qquad (4.14)$$

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

$$B = \left|\,\{\{\tilde{u}\}\} + \{\{c\}\}\,\right| - \left|\,\{\{\tilde{u}\}\} - \{\{c\}\}\,\right| = \left|\frac{1}{2}\tilde{u}^+ + \{\{c\}\}\right| - \left|\frac{1}{2}\tilde{u}^+ - \{\{c\}\}\right|, \qquad (4.15)$$

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

$$\frac{\Delta t}{g\,\omega_0\, a}\frac{\{\{c\}\}\,B\,[\![\tilde{u}]\!]}{h^-} \overset{!}{\geq} -1. \qquad (4.16)$$

This condition guarantees non negativity for the third term in (4.10) and can only be violated if $B\,[\![\tilde{u}]\!] < 0$. There are two sets of conditions 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

$$\Delta t \overset{!}{\leq} \left|\frac{g\,\omega_0\, a\, h^-}{\{\{c\}\}\,B\,[\![\tilde{u}]\!]}\right|. \qquad (4.17)$$

This proof for one Euler time step extends to SSPRK methods as used in the ESDGSEM as seen in, e.g., [50]. ■

While Lemma 2 guarantees a non-negative average water height in the next time step, we still need to ensure point-wise non-negativity. We use the limiter applied to the shallow water equations by Xing et al [50] and developed in [35, 49, 52] which is a linear scaling around the element average

$$\widehat{\vec{W}}_{ij} = \theta\left(\vec{W}_{ij} - \overline{\vec{W}}_E\right) + \overline{\vec{W}}_E, \qquad (4.18)$$

where θ is computed by

$$\theta = \min\left(1, \frac{\overline{h}_E}{\overline{h}_E - m_E}\right), \tag{4.19}$$

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 2, this limiter ensures positive water height for all computation nodes. It can be shown that this limiter maintains high order accuracy and is conservative [51]. We show that the positivity preserving limiter is also entropy stable in Lemma 3. 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. Then it follows immediately that the global entropy is also not increased by this procedure.

Lemma 3 (Entropy Stability of Positivity Preservation)**.** *An entropy stable method coupled with the positivity preserving limiter* (4.18) *is still entropy stable.*

Proof. We prove this result in a similar fashion to Ranocha [37]. Let $\mathcal{E}(\vec{W})$ denote the discrete total energy (entropy) within an element with solution polynomial $\vec{W}$. Also, we introduced the limited value of the solution polynomial around the element average $\widehat{\vec{W}}$ (4.18). Then

$$\begin{aligned}
\overline{\mathcal{E}\left(\widehat{\vec{W}}\right)} = \frac{1}{|E|}\sum_{i=0}^{N}\sum_{j=0}^{N}\mathcal{E}\left(\widehat{\vec{W}}_{ij}\right) J_{ij}\,\omega_i\omega_j &\overset{\mathcal{E}\text{ convex}}{\leq} \frac{1}{|E|}\theta\sum_{i=0}^{N}\sum_{j=0}^{N}\mathcal{E}(\vec{W}_{ij})J_{ij}\,\omega_i\omega_j + \frac{1}{|E|}(1-\theta)\sum_{i=0}^{N}\sum_{j=0}^{N}\mathcal{E}\left(\overline{\vec{W}}\right) J_{ij}\,\omega_i\omega_j \\
&= \frac{1}{|E|}\theta\sum_{i=0}^{N}\sum_{j=0}^{N}\mathcal{E}(\vec{W}_{ij})J_{ij}\,\omega_i\omega_j + (1-\theta)\mathcal{E}\left(\overline{\vec{W}}\right) \\
&\overset{\text{Jensen's inequality}}{\leq} \frac{1}{|E|}\theta\sum_{i=0}^{N}\sum_{j=0}^{N}\mathcal{E}(\vec{W}_{ij})J_{ij}\,\omega_i\omega_j + (1-\theta)\overline{\mathcal{E}(\vec{W})} \\
&= \theta\overline{\mathcal{E}(\vec{W})} + (1-\theta)\overline{\mathcal{E}(\vec{W})} \\
&= \overline{\mathcal{E}(\vec{W})}.
\end{aligned} \tag{4.20}$$

So, the entropy of the modified solution polynomial $\hat{W}$ is less or equal to the entropy of the unmodified polynomial and it follows that the positivity limiter does not increase the entropy of the system. ■

We summarize the results of this section in Theorem 2.

Theorem 2 (ESDGSEM with positivity limiter)**.** *The ESDGSEM* (2.19) *combined with the positivity preserving limiter* (4.18) *and the additional time step restrictions* (4.2) *fulfills all the properties from Lemma 1 and also guarantees non-negative water heights for all LGL-nodes.*

Proof. We proved in Lemma 2 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 preserving limiter (4.18) then guarantees non-negative water height at all LGL nodes. Finally, Lemma 3 proves that the positivity preserving limiter is entropy stable. We note that the properties from Lemma 1 of mass conservation and the well-balancedness are unaffected by the positivity preservation procedure. ■

Remark 1. Theorem 2 also holds when combining the positivity preserving limiter with the artificial viscosity shock capturing from Sec. 3 since no artificial viscosity is added to the continuity equation.

Remark 2. The positivity limiter does not affect the well-balanced property of the ESDGSEM 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

$$\begin{aligned}
h &= \max\{H_{\text{const}} - b, 0\} \\
u &= v = 0,
\end{aligned} \tag{4.21}$$

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 $H = h + b = \text{const}$ 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. [3].

5. GPU Implementation

Discontinuous Galerkin implementations on graphics processing units (GPUs) have been previously studied, for example in [8, 13, 21, 22, 23, 31]. DG algorithms with explicit time integration are particularly well suited to the massively parallel GPU architectures as most of the computational work is element local and elements are only coupled through interface exchanges for the computation of the surface integrals. None of the modifications in the ESDGSEM change the strong parallelizability. The ESDGSEM is, however, more computationally expensive than the standard DGSEM when counting the number of operations. Specifically the split form volume integral requires additional floating point operations compared to a standard volume integral. 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 unleashed GPU processing power.

The ESDGSEM GPU implementation is based on the abstract language OCCA, a unified approach to multi-threading languages, developed by Medina et al. [30]. OCCA compiles OCCA kernel language (OKL) code at runtime for either CPU (Serial, OpenMP) or GPU (OpenCL, CUDA) architectures. Our main test system features a NVIDIA GTX 1080 which is a higher end consumer grade card at the time this paper is written. We compile the kernels in CUDA and run the code in single precision as the GTX 1080 lacks double precision processing power. Only about $\frac{1}{32}$ of its CUDA cores are suited for double precision computations such that the theoretical peak performance is only 257 GLFOPS/s compared to the 8228 GFLOPS/s for single precision calculation. Thus, we use a different GPU, the NVIDIA Tesla V100, to produce double precision results. The Tesla V100 is a card designed for scientific computations and very well suited for double precision calculations as the double precision processing power is half of the single precision processing power. We show the double precision results in Subsection 5.1.

In the implementation, we set up the mesh and problem on the CPU host and copy all the necessary data onto the GPU at the beginning of the computation. Then, data is only transferred back for MPI communication when using multiple GPUs, or for visualization purposes. Otherwise the whole computation is done on the GPU. As host to device data transfer is rather slow, this is an important aspect for the efficiency of the code. The different parts of the ESDGSEM are separated into individual kernels. There are, for instance, kernels for the computation of the convective surface and volume integrals as well as their viscous counterparts. In both cases, the most computationally expensive kernel in terms of floating point operations are the volume kernels. This is true for the implementation of the standard discontinuous Galerkin method as well as the ESDGSEM described in this paper. As the only difference between a standard DGSEM method and the ESDGSEM proposed in this paper is the volume integral, we focus our analysis on this specific part of the computation. We provide an overview of the runtimes of all the different kernels executed in the computation in Section 5.2.

The computation of the ESDGSEM volume integral requires significantly more operations than the standard DGSEM volume kernel. To see this, we restate the algorithm for the volume integrals at a node i, j:

$$\begin{aligned} (\text{Split Form Volume})_{ij} &= \sum_{l=0}^{N} \tilde{\mathbf{D}}_{il} \tilde{F}^{\#}_{(l,i),j} + \sum_{l=0}^{N} \tilde{\mathbf{D}}_{jl} \tilde{G}^{\#}_{i,(j,l)}, \\ (\text{Standard Volume})_{ij} &= \sum_{l=0}^{N} D_{il} \tilde{F}_{l,j} + \sum_{l=0}^{N} D_{jl} \tilde{G}_{i,l}. \end{aligned} \tag{5.1}$$

While the formulas look similar, 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 need to be evaluated once for 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. While there are performance numbers provided by the manufacturer, in practice these theoretical 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 computed by

$$\text{effective bandwidth} = \frac{\text{Bytes Read + Bytes Written}}{t}. \tag{5.2}$$

We compare the effective bandwidth of the volume kernels with the effective bandwidth of a GPU memory copy with the same number of bytes read and written. This memory copy is executed with a cudaMemcpyDeviceToDevice command and the resulting bandwidth is called the MemCopy-bandwidth. Since a cudaMemcpy reads and writes each entry, a buffer of the size $\frac{\text{Bytes Read + Bytes Written}}{2}$ is used to calculate the MemCopy-bandwidth. We compute an empirical MemCopy roofline by scaling the GFLOPS/s achieved by the kernel with the factor of MemCopy-bandwidth over effective kernel bandwidth

$$\text{empirical MemCopy roofline} = \frac{\text{GFLOPS/s} \times \text{MemCopy-bandwidth}}{\text{effective Kernel bandwidth}}. \tag{5.3}$$

The MemCopy roofline is a good upper bound on kernel performance whenever a kernel is limited by the memory bandwidth or **memory-bound**. When a kernel's performance is limited by the computations, or **compute-bound**, stricter bounds are needed. Another limiting factor can be the shared memory bandwidth. The shared memory bandwidth is estimated by

$$\text{shared memory bandwidth} = \#\text{cores} \times \#\text{SIMD Lanes} \times \|\text{word in bytes}\| \times \text{clock frequency}, \tag{5.4}$$

which for the GTX 1080 is

$$\text{Shared-Mem-Bandwidth} = 20 \times 32 \times 4\text{bytes} \times 1.607\text{GHz} = 4113.92\text{GB/s}. \tag{5.5}$$

With the shared memory bandwidth we can estimate a bounding roofline by

$$\text{shared memory roofline} = \text{Shared-Mem-Bandwidth} \times \frac{\text{flops per block}}{\text{shared-mem bytes loaded+stored per block}}. \tag{5.6}$$

We combine the MemCopy roofline and the shared memory roofline to find the combined roofline we will use in the following kernel analysis:

$$\text{combined roofline} = \min\left(\text{shared memory roofline}, \text{MemCopy roofline}\right). \tag{5.7}$$

These performance bounds give us a sharper hardware limit on the performance than numbers provided by the manufacturer and whenever the actual kernel performance is close to the roofline we can be satisfied with the kernel efficiency. In compute bound cases it may, however, still be impossible to actually reach the performance bound provided by the roofline.

Starting with a straightforward, naive implementation of the split form volume integral from (5.1) we sequentially introduce optimization steps and document the impact on the kernel performance. We measure

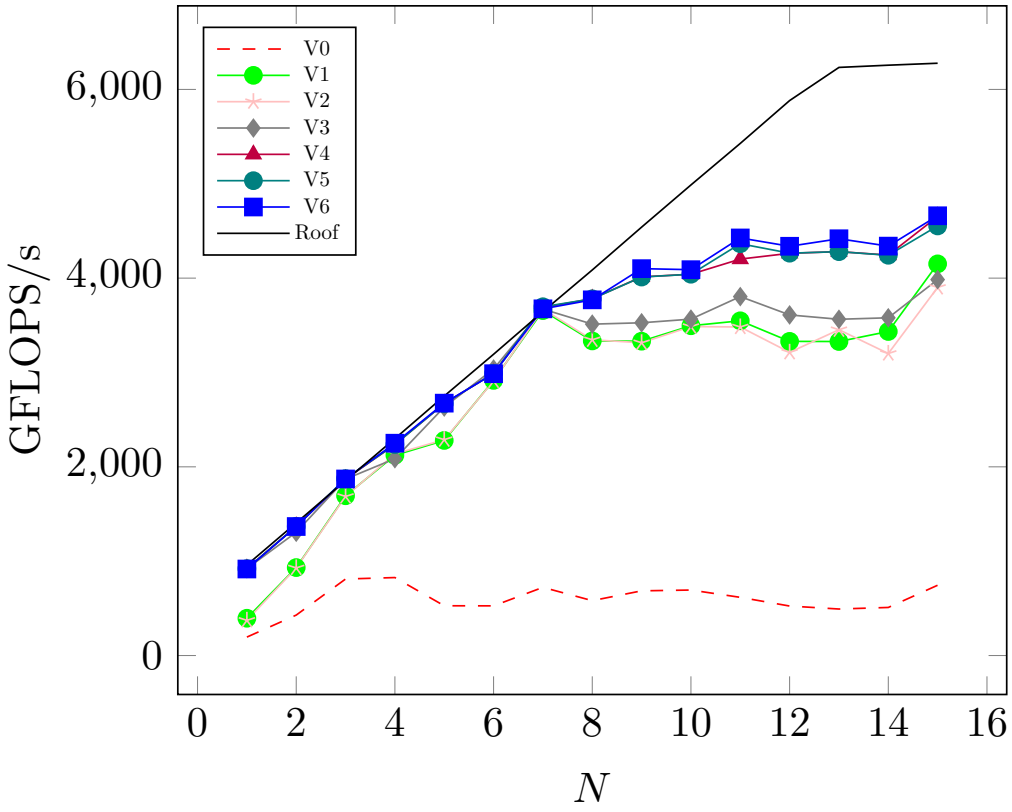


Figure 1: Comparison of all ESDGSEM volume kernel versions and the memory roofline on a NVIDIA GTX 1080 in single precision.

the average kernel runtime for a sample test case over 1000 kernel executions. We count the number of floating point operations and obtain the performance measure GFLOPS/s as the number of floating point operations performed during each second of runtime. We increase the polynomial order N and decrease the number of elements K to increase the computational complexity of the volume integral, while staying as close as possible to 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 4. The optimization steps applied in each different kernel version are described in Appendix C. The optimization techniques are similar to previous works, e.g. in [13] and subject to current research, e.g. [9, 20, 40].

We illustrate the impact of optimization techniques and plot the achieved performance of the different kernel versions and the empirical roofline for split form and standard kernels in Figure 1 and Figure 2. Finally, we compare the most optimized versions for each kernel and show operation counts and runtimes for $N = 1, \ldots, 15$ with similar degrees of freedom in Figure 3 and for similar memory loads in Figure 4. The achieved performances of the ESDGSEM volume kernel are memory bound and thus close to optimal for $N \leq 7$ and then stagnate as the performance becomes compute bound. The standard volume kernel behaves differently as even for $N = 15$ the achieved performance lies close to the roofline. As split form and standard kernels require the same amount of data, the runtime is almost identical in the memory bound region. For higher polynomial orders, the split form volume kernel becomes compute bound and the runtimes deviate. Here we can clearly observe the diverging number of floating point operations. However, while there is a factor of 6 difference in the number 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. Another observation to point out is that the achieved GFLOPS/s of the split form kernel are higher by a factor of 3 to 5.5. This suggests that the ESDGSEM volume kernel makes good use of the computational capabilities of the GPU architecture and is indeed a very good fit. Especially for the lower polynomial orders the additional computational complexity is completely mitigated by GPU processing power.

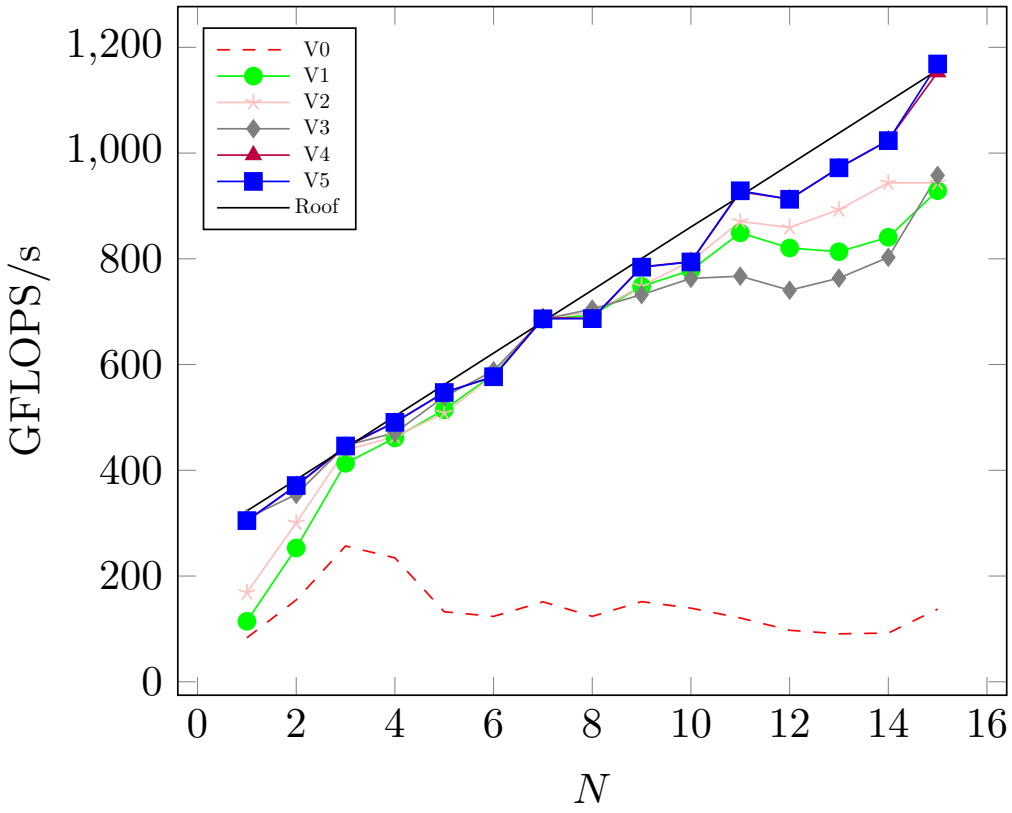


Figure 2: Comparison of all standard DGSEM volume kernel versions and the memory roofline on a NVIDIA GTX 1080 in single precision.

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

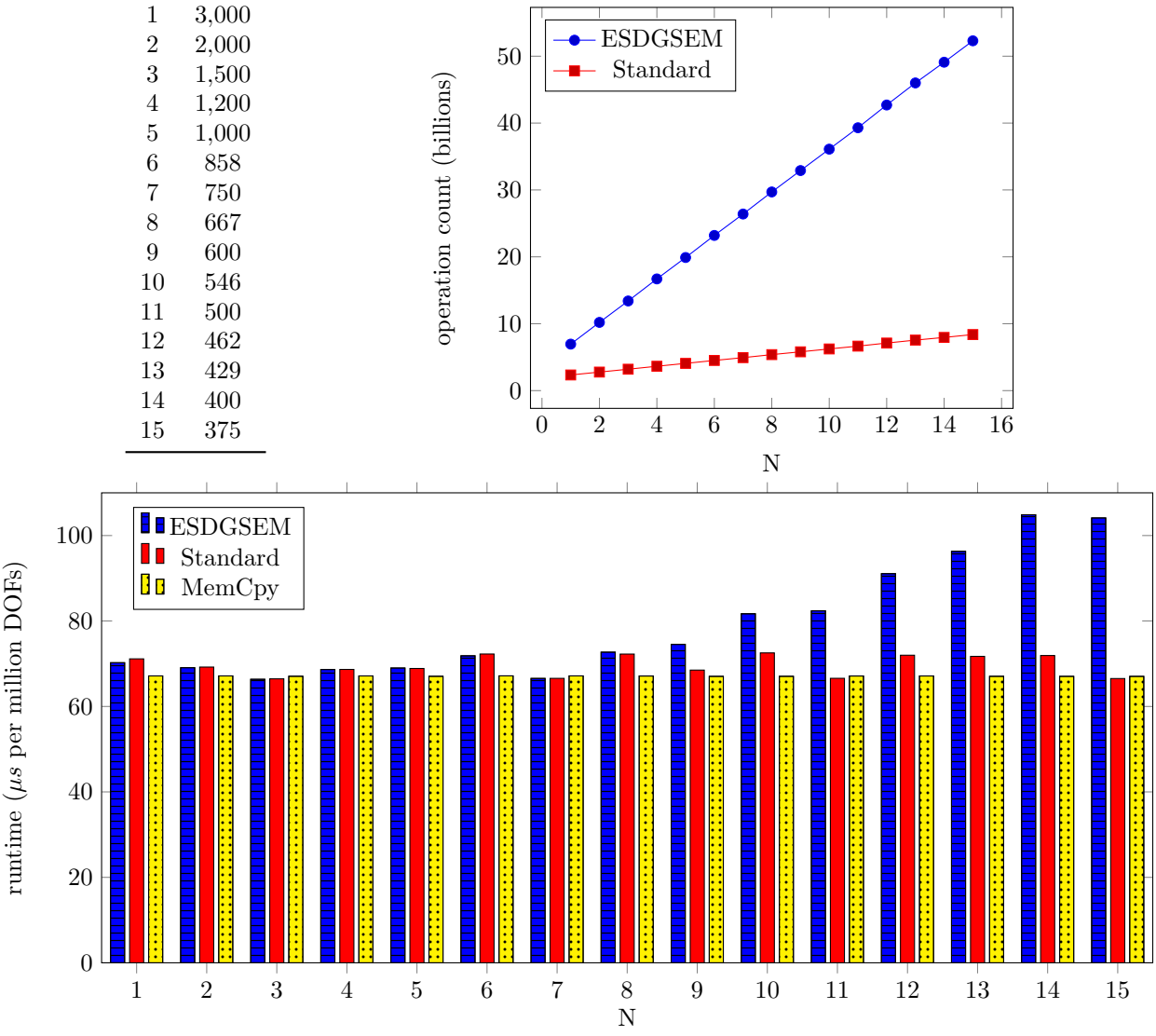


Figure 3: Number of operations and runtime comparison for one kernel execution between the split form volume integral computation of the ESDGSEM and the volume integral of the standard DG method for similar number of degrees of freedom (DOFs) on a NVIDIA GTX 1080 in single precisions. Polynomial order N and number of elements K per spatial direction are listed in the top left table.

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

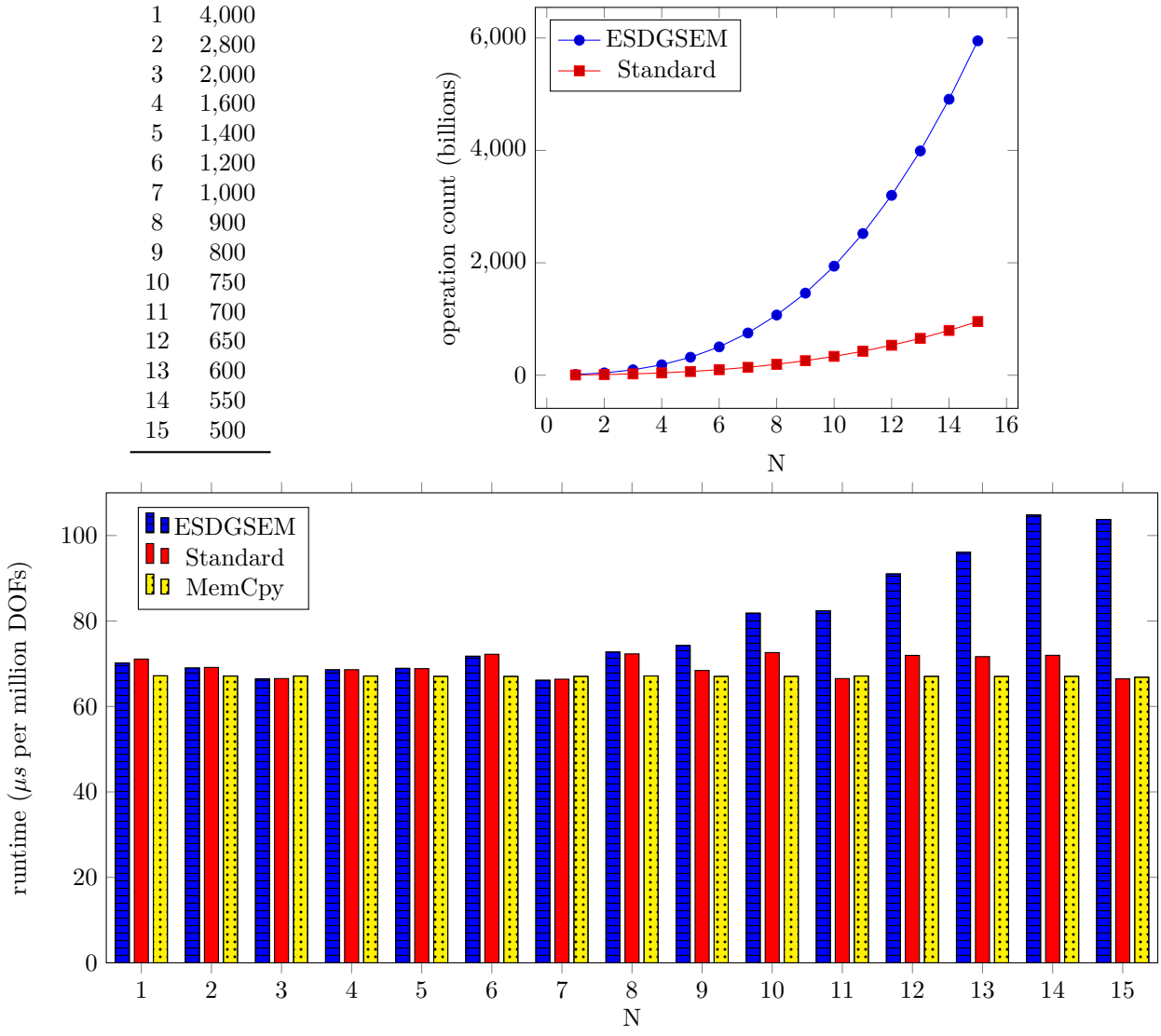


Figure 4: Number of operations and runtime comparison for one kernel execution between the split form volume integral computation of the ESDGSEM and the volume integral of the standard DG method for increasing computational complexity on a NVIDIA GTX 1080 in single precision. Polynomial order N and number of elements K per spatial direction are listed in the top left table.

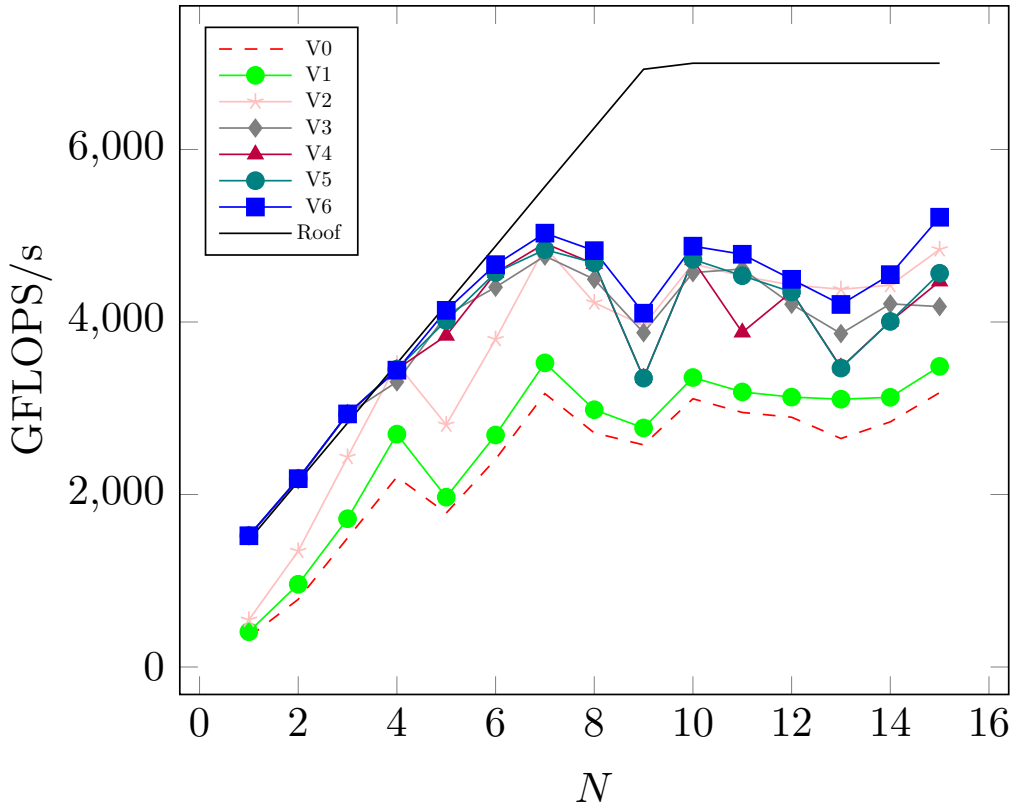


Figure 5: Comparison of all ESDGSEM volume kernel versions and the memory roofline on a Tesla V100 in double precision.

5.1. Double Precision Results

We use a different test system with a Tesla V100 to run the ESDGSEM volume kernel in double precision. We use the same kernels as for the single precision results. We show the performance of the different split form kernel versions in Figure 5 as well as for the standard kernel versions in Figure 6. We, again, compare runtimes for the most optimized kernels in Figure 7. The elements per work block were found by optimization for the GTX 1080 so it may be possible to find better values for the Tesla V100. This could explain why the most optimized kernel is already compute bound for $N = 6$ to $N = 7$. The significant drops in performance for $N = 9$ in Figure 5 and for $N = 15$ for some (sub-optimal) kernel versions in Figure 6 might be related to shared memory bank conflicts and could possibly be removed by a different choice of elements per work block. Another strategy to avoid shared memory bank conflicts is to add padding if $N + 1$ is a multiple of 4. In the optimization steps this is including starting at version 5, and shows a considerable performance improvement in Figure 6. We do however see very satisfactory performance for lower order N where, once again, the split form kernel performance is close to optimal.

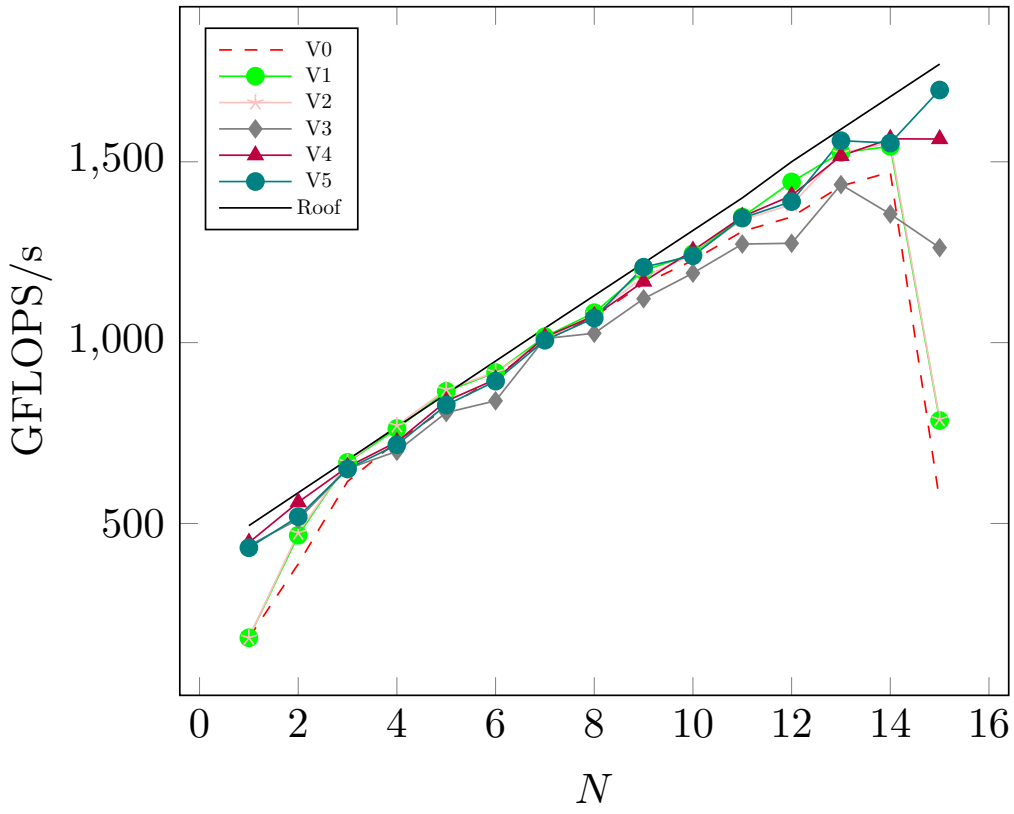


Figure 6: Comparison of all standard DG volume kernel versions and the memory roofline on a Tesla V100 in double precision.

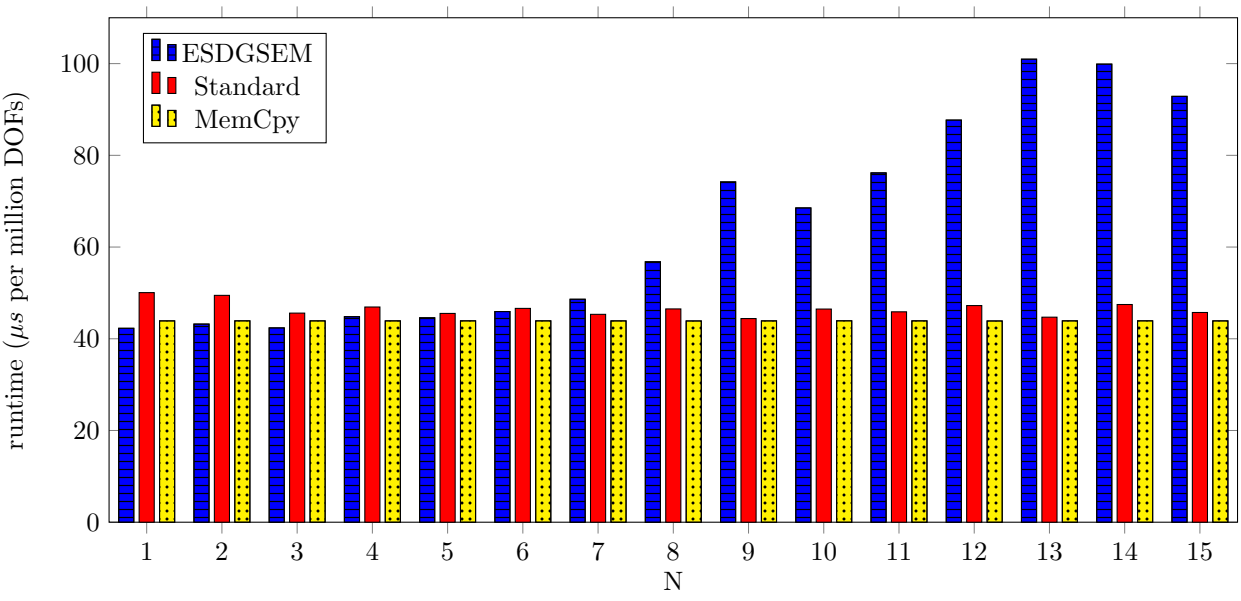


Figure 7: Number of operations and runtime comparison for one kernel execution between the split form volume integral computation of the ESDGSEM and the volume integral of the standard DG method for increasing computational complexity in double precision on the NVIDIA Tesla V100. The number of elements per polynomial order are identical to the single precision case in Figure 4.

5.2. Runtime overview

To get a better grasp on the runtimes of the various kernels that are part 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 1. We note that while the volume kernel is the most thoroughly optimized, similar steps, listed in Appendix C, 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 1: Runtimes in relation to the total code runtime for a dam break problem on a 200×200 element mesh for polynomial degrees $N = 5, 10, 15$.

6. Numerical Results

In this section we first numerically verify and demonstrate the theoretical properties of the scheme in Subsection 6.1. In particular, we present the difference in the numerical solution of the entropy stable scheme with artificial viscosity presented in this paper, the ESDGSEM without artifical viscosity, and a standard DGSEM of the same polynomial order. To fully explore the solution quality of the new scheme, we apply the ESDGSEM with artificial viscosity to several well-known test cases which require shock capturing and positivity preservation. Namely, the oscillating lake [7, 12, 28, 44, 48], the three mound dam break on a closed channel [5, 7, 12, 28, 49] and a solitary wave run-up [2, 28, 36]. Lastly, to test the properties on a curvilinear mesh, we modify the partial dam break from [46] to feature a dry area on the shallow side of the dam.

We choose the time step based on a typical CFL condition. The additional time step restrictions of the positivity limiter (4.2) are sufficient but not necessary. If we detect that a smaller time step is needed, we adjust it accordingly. A more detailed discussion on choosing an appropriate time step for positivity preserving schemes is given by Xing and Zhang [49].

All the examples have been computed on two NVIDIA GTX 1080 GPUs, where we use MPI parallelization such that each MPI rank hosts one GPU via OCCA [29, 30].

6.1. Theory Validation

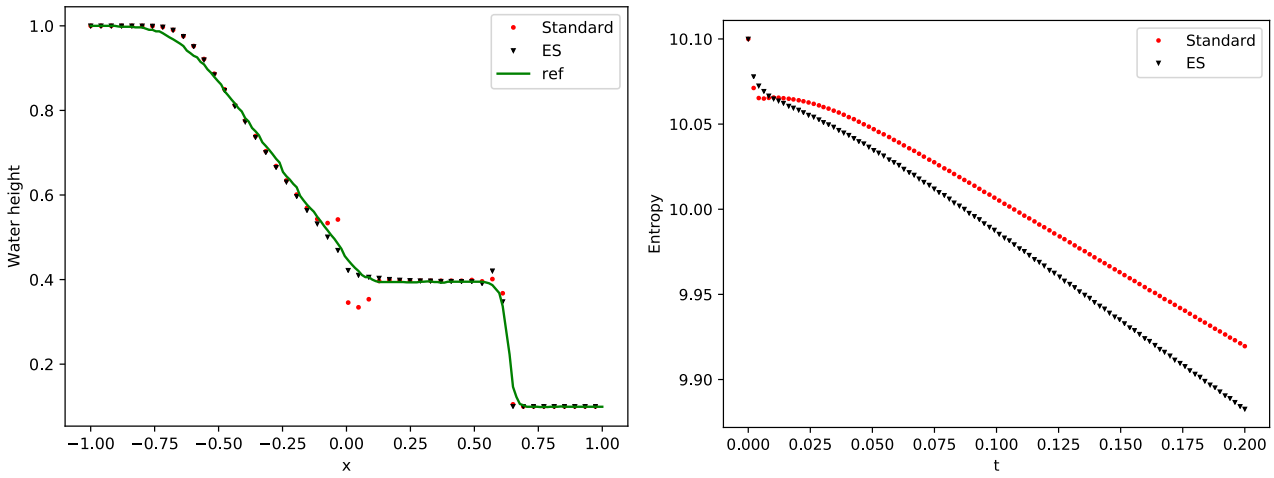
We first verify the theoretical entropy stable properties of the approximation described in this work. The ESDGSEM was previously shown to be high-order accurate but unphysical overshoots remained near discontinuities in Wintermeyer et al. [46]. Therefore, we demonstrate in Sec. 6.1.1 that applying the artificial viscosity from Thm. 1 can remove spurious oscillations and remain entropy stable. We then consider a shocktube test in Sec. 6.1.2 that requires the positivity preserving limiter to model a flow with dry regions. Further, we numerically verify the result of Lemma 2 that the limited DG solution is entropy stable. We choose a positivity tolerance level of $h^{\mathrm{TOL}} = 10^{-4}$ below which velocities are set to zero in the positivity preserving stage.

6.1.1. Entropy Glitch

We use a specific dam break problem to demonstrate the necessity of entropy stability to guarantee correct solutions. We set up a two-dimensional version of the one-dimensional test case previously done in [27]. We use a uniform Cartesian mesh on the domain $\Omega = [-1, 1]^2$ with 100×100 elements. There is a shock at $x = 0$ with water heights $h_L = 1$ and $h_R = 0.1$ and the velocities are zero, i.e.,

$$\begin{aligned} h(x, y, 0) &= \begin{cases} 1.0, & \text{if } x < 0 \\ 0.1, & \text{otherwise} \end{cases}, \\ u &= v = 0. \end{aligned} \tag{6.1}$$

For simplicity, the gravity constant is set to $g = 10$ for this example and the test is run up to $T = 0.2$. We show results for $N = 1$ as the entropy glitch even occurs for this most robust case. Also the standard DGSEM quickly becomes unstable for higher polynomial orders due to the oscillations introduced by the entropy glitch. We still use the positivity limiter to catch some minor overshoots due to the oscillations. We compare a slice of the solution at $y = 0$ with the 1D solution provided in [27]. Solutions obtained by the standard DG method with a local Lax-Friedrichs numerical interface flux show an unphysical discontinuity, an "entropy glitch," at $x = 0$, see Figure 8. The unphysical shock does not appear in the solutions obtained by the ESDGSEM. There are however oscillations at the shock front for both schemes. We also plot the evolution of the total entropy for the both cases and observe that the total entropy builds up at around $t = 0.02$ for the standard DGSEM whereas the total entropy is strictly decreasing for the ESDGSEM. We conclude that even limited entropy build up may cause unphysical phenomena to appear and destroy the correctness of the solution.



(a) Water height H, $N = 1$

(b) Total Entropy, $N = 1$

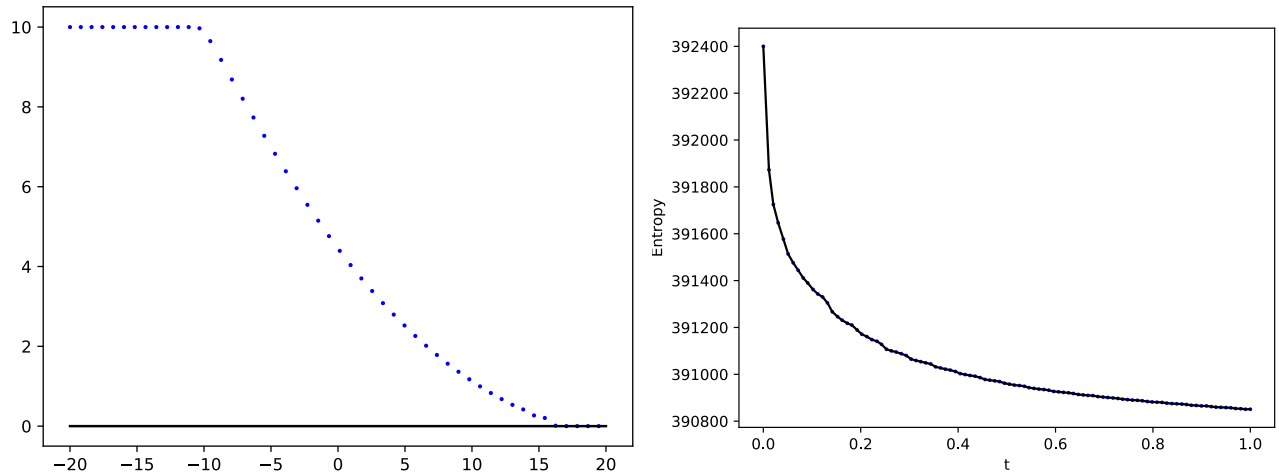
Figure 8: Entropy glitch test case for ESDGSEM and standard DGSEM for $N = 1$ at $T = 0.2$ sliced at $y = 0$ compared to a 1D reference solution from [27]. The standard DGSEM produces an incorrect shock due to the unphysical entropy production.

6.1.2. Necessity of Positivity Limiter

First, we numerically verify the mass conservation and entropy stability of the ESDGSEM with artificial viscosity and positivity preservation in the presence of dry areas. The test case is a dam break problem with periodic boundary conditions at $y = \pm 20$ and solid walls at $x = \pm 20$ on the domain $\Omega = [-20, 20]^2$ with initial conditions

$$\begin{aligned} h(x, y, 0) &= \begin{cases} 10.0, & \text{if } x < 0 \\ 0.0, & \text{otherwise} \end{cases}, \\ u &= v = 0. \end{aligned} \tag{6.2}$$

We use a polynomial order of $N = 3$ and a uniform Cartesian mesh with 50×50 elements. We set the viscosity coefficient to be $\epsilon_0 = 0.1$ and use a gravitational constant of $g = 9.81$. From the results in Figure 9 we see that the entropy is monotonically decreasing as expected. The mass is conserved up to machine precision. This test case crashes immediately without the use of a positivity limiter.



(a) Water height H at $y = 0$, $T = 1.0$

(b) Total Entropy over time

Figure 9: 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 $N = 3$.

6.2. Oscillating Lake

The oscillating lake is a parabolic bowl partly covered with water that is moving around the center of the domain $\Omega = [-2, 2] \times [-2, 2]$ and defined by

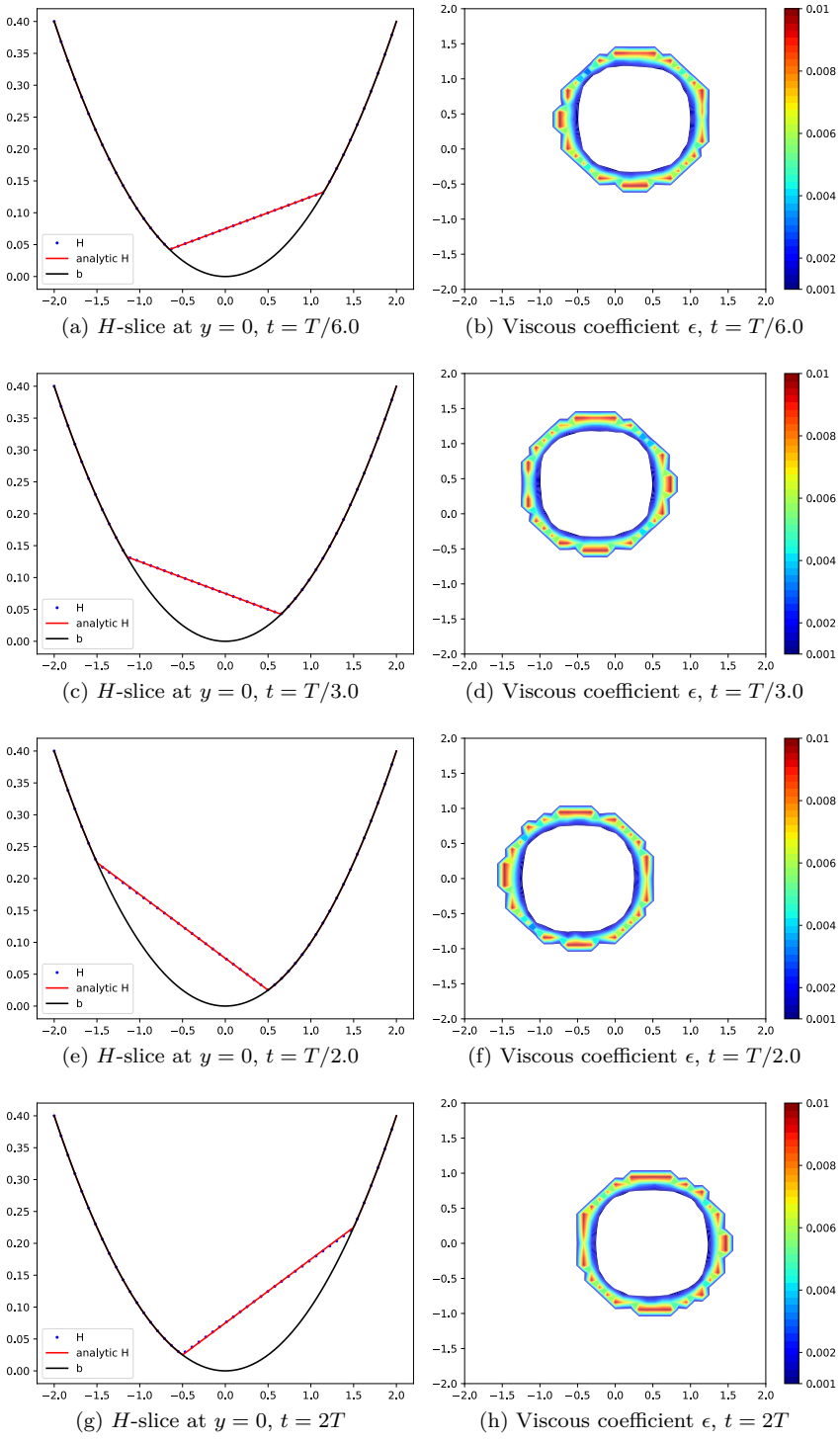
$$\begin{aligned} h(x, y, 0) &= \max\left(0, \sigma \frac{h_0}{a^2} \left(2x \cos \omega t + 2y \sin \omega t - \sigma\right) + h_0 - b\right), \\ u(x, y, 0) &= -\sigma\omega \sin \omega t, \\ v(x, y, 0) &= \sigma\omega \cos \omega t. \end{aligned} \tag{6.3}$$

with parabolic bottom topography

$$b(x, y) = h_0 \frac{x^2 + y^2}{a^2}, \tag{6.4}$$

with parameters $h_0 = 0.1$, $a = 1$ $\sigma = 0.5$ and $\omega = \frac{\sqrt{2gh_0}}{a}$. The boundary conditions can be set to solid walls as the water flow never reaches the domain boundaries. The base viscosity parameter is set to $\epsilon_0 = 0.01$. The gravitational constant is set to $g = 9.81$. We use a uniform Cartesian mesh with 200×200 elements for the computation.

The oscillating lake test case tests how well a numerical method is able to handle wetting and drying as the fluid evolves. There are no strong shocks so the water pond should travel smoothly around the center of the domain. We plot a slice through $y = 0$ as well as the dynamic viscosity coefficient for the domain in Figure 10. We see that viscosity is only applied on the edges of the wet circle with small magnitudes, as not much viscosity is necessary for this case.



(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, $t = T/3.0$

(e) H-slice at $y = 0$, $t = T/2.0$ (f) Viscous coefficient ϵ, $t = T/2.0$

(g) H-slice at $y = 0$, $t = 2T$ (h) Viscous coefficient ϵ, $t = 2T$

Figure 10: ESDGSEM approximation with artificial viscosity and positivity limiter for the 2D oscillating lake at $N = 3$.

6.3. Three Mound Dam Break

We proceed with a more challenging test case to thoroughly test the shock capturing capabilities as well as the positivity preservation. A dam break is set up on the domain $\Omega = [0, 75] \times [0, 45]$ at $x = 16$ with a water height of $h = 1.875$ on the top and zero at the bottom. On the dry side of the dam are three mounds which will be partially flooded during the computation. The water is initialized at rest

$$\begin{aligned} h(x, y, 0) &= \begin{cases} 1.875, & \text{if } x < 16 \\ 0, & \text{otherwise} \end{cases}, \\ u &= v = 0. \end{aligned} \tag{6.5}$$

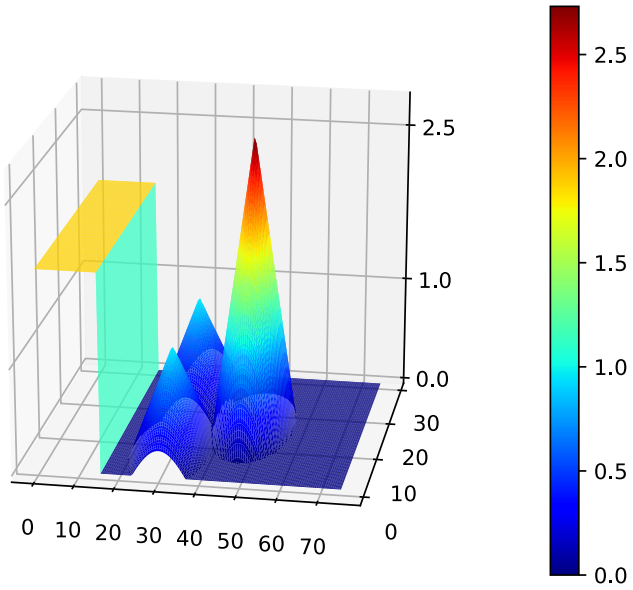
The three mounds on the down hill side are defined by

$$\begin{aligned} M_1(x, y) &= 1 - 0.1\sqrt{(x - 30)^2 + (y - 22.5)^2}, \\ M_2(x, y) &= 1 - 0.1\sqrt{(x - 30)^2 + (y - 7.5)^2}, \\ M_3(x, y) &= 2.8 - 0.28\sqrt{(x - 47.5)^2 + (y - 15)^2}, \end{aligned} \tag{6.6}$$

and the bottom topography is taken as the maximum elevation level

$$b(x, y) = \max(0, M_1(x, y), M_2(x, y), M_3(x, y)) \tag{6.7}$$

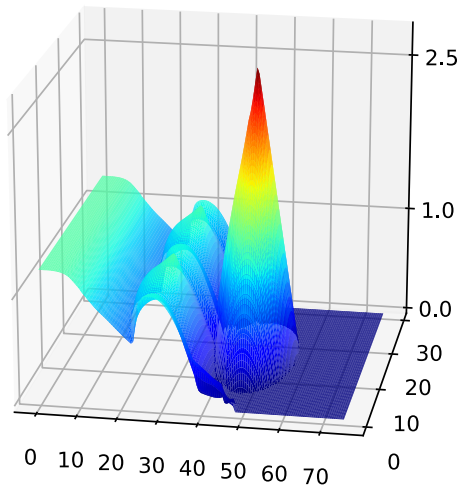
The set up uses solid wall boundary conditions on all four sides. We use a Cartesian mesh with 150×100



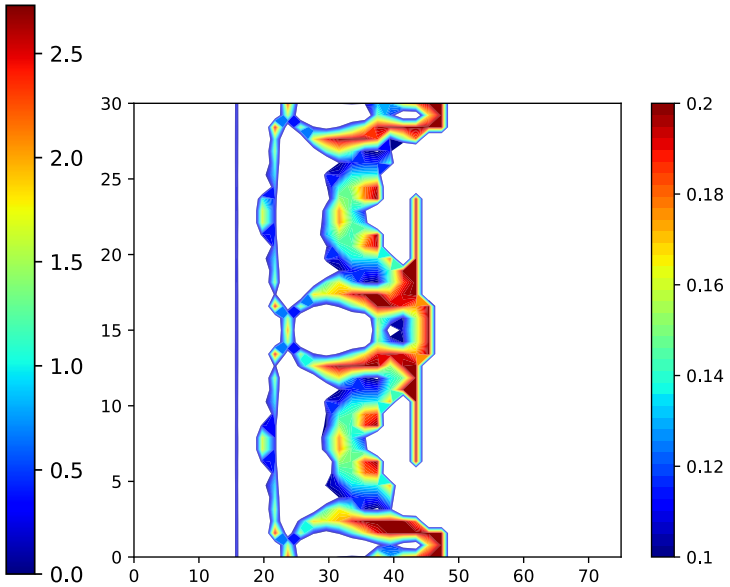
(a) H, $T = 0.0$

Figure 11: Initial condition for the dam break over three mounds $N = 3$.

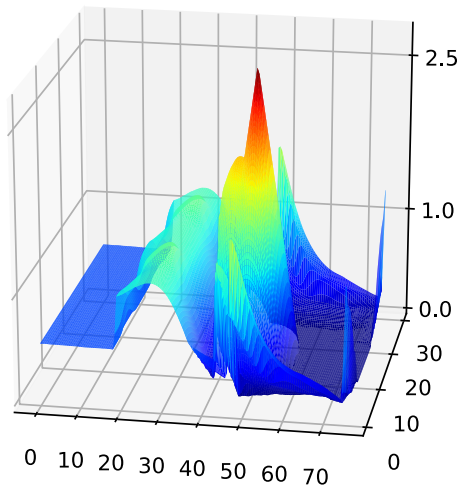
elements. Due to the combination of a strong shock and a dry area including varying bottom elevations, the base viscosity parameter is set to be $\epsilon_0 = 0.2$. The gravitational constant is set to $g = 9.81$. We show the total water height at various times in Figure 11 and Figure 12. Also included in these figures is the dynamic viscous coefficient ϵ. These plots show that the shock capturing mechanism accurately tracks the shock fronts across the domain and around the three mounds.



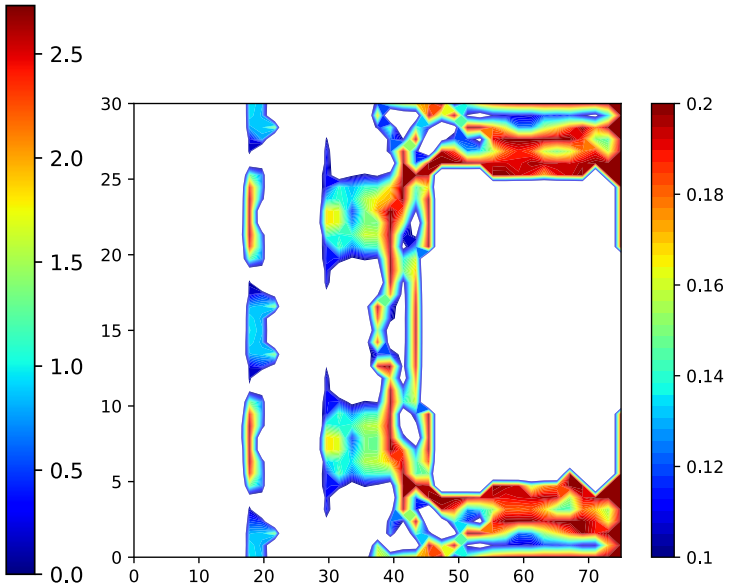
(a) H, $T = 5.0$



(b) Viscous coefficient ϵ, $T = 5.0$

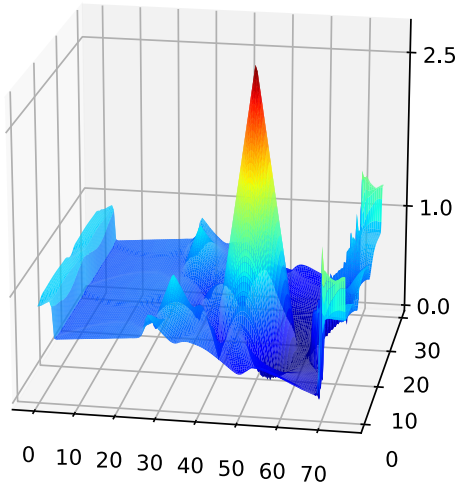


(c) H, $T = 10.0$

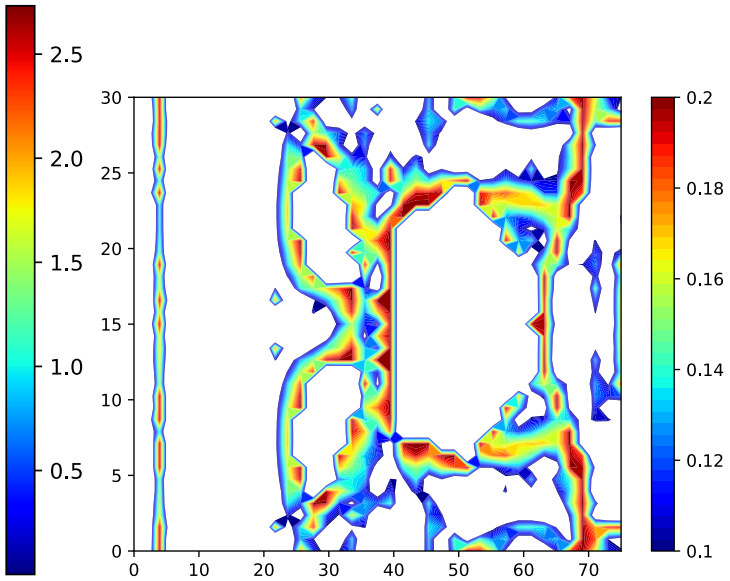


(d) Viscous coefficient ϵ, $T = 10.0$

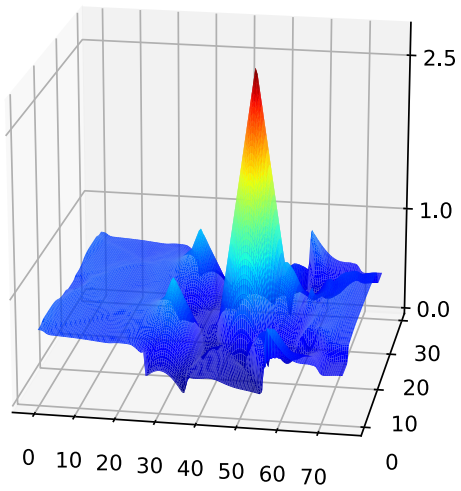
Figure 12: ESDGSEM approximation with artificial viscosity and positivity limiter for the dam break over three mounds $N = 3$.



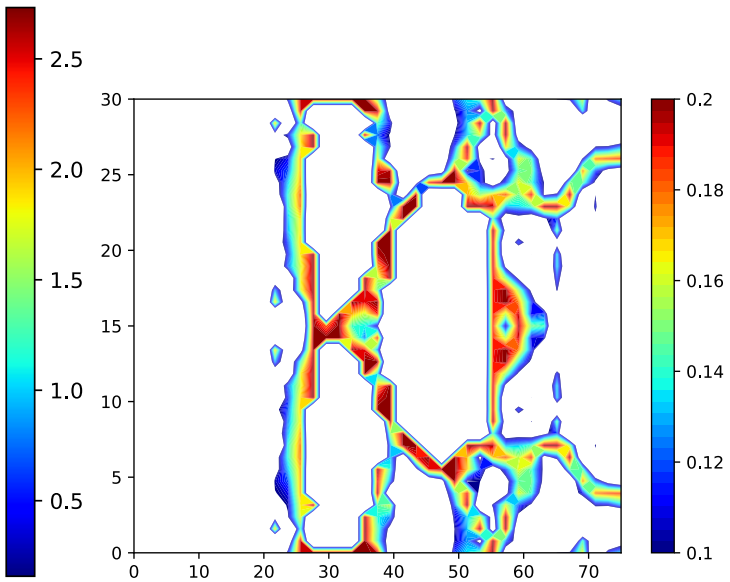
(a) H, $T = 20.0$



(b) Viscous coefficient ϵ, $T = 20.0$

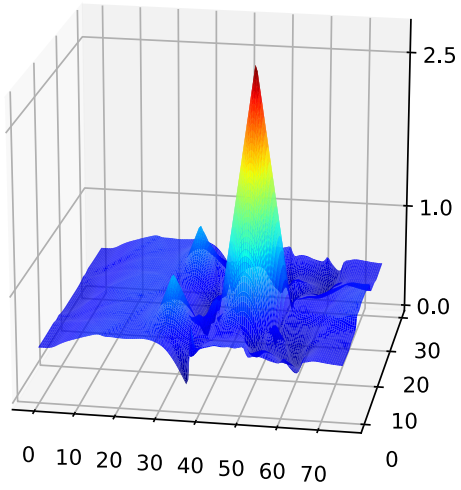


(c) H, $T = 30.0$

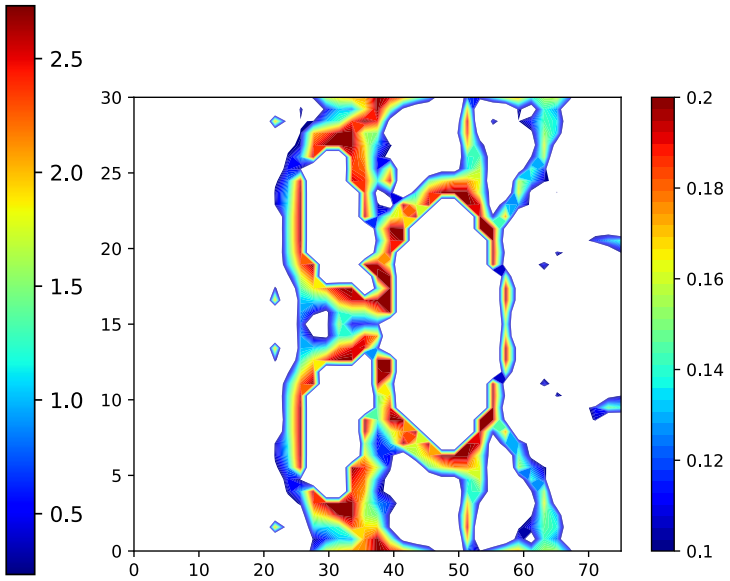


(d) Viscous coefficient ϵ, $T = 30.0$

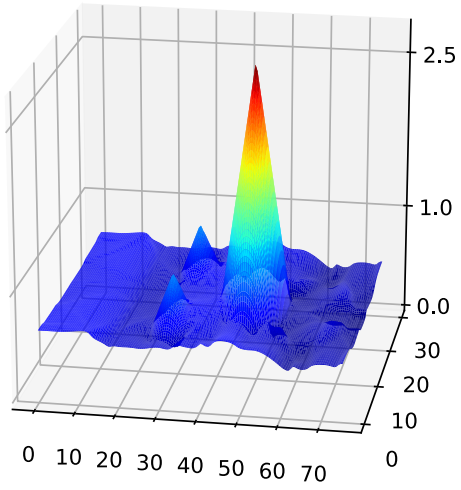
Figure 13: ESDGSEM approximation with artificial viscosity and positivity limiter for the dam break over three mounds $N = 3$.



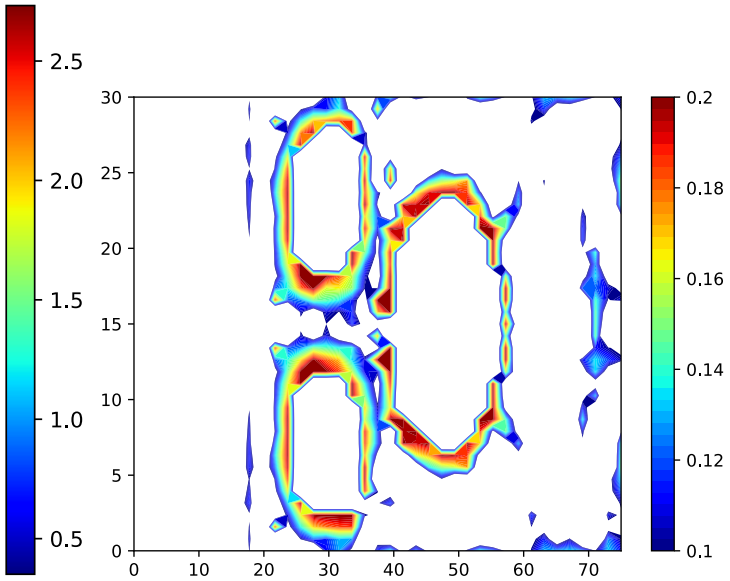
(a) H, $T = 40.0$



(b) Viscous coefficient ϵ, $T = 40.0$



(c) H, $T = 50.0$



(d) Viscous coefficient ϵ, $T = 50.0$

Figure 14: ESDGSEM approximation with artificial viscosity and positivity limiter for the dam break over three mounds $N = 3$.

6.4. Solitary wave runup on a conical island

We examine the runup of a wave on the domain $\Omega = [0, 25] \times [0, 30]$ with a partly dry conical island in the center of the domain. The wave flows around the island, is reflected at the far end and flows back around it. This test case was previously studied numerically in [28, 41] and experimentally in [4]. The initial wave η defined by

$$\eta(x, y, 0) = \frac{A}{h_0}\text{sech}^2(\gamma(x - x_c)), \tag{6.8}$$

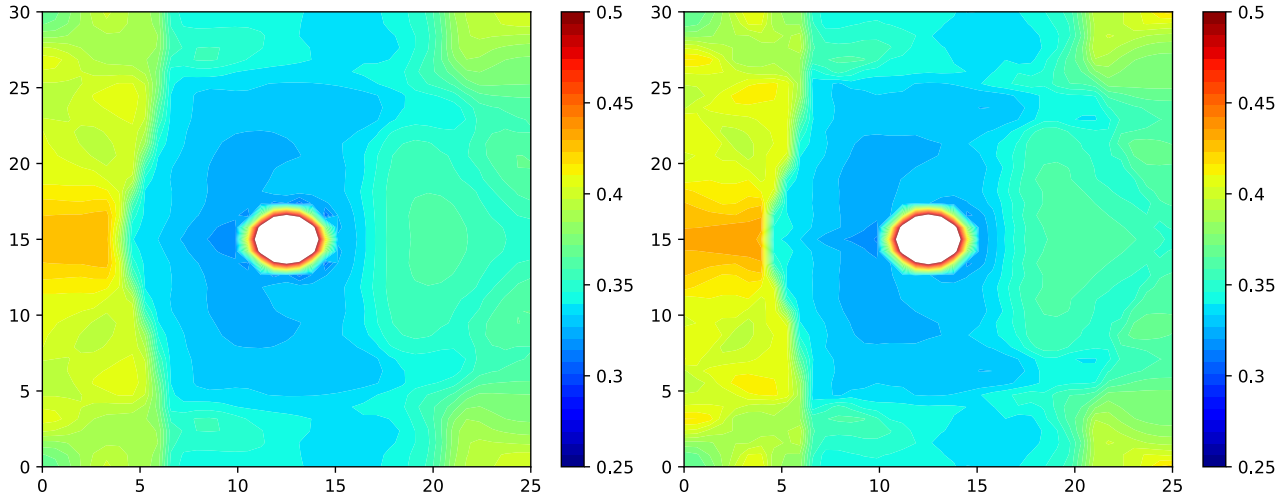
is set on top a flat water level of $h_0 = 0.32$, leading to initial conditions of

$$\begin{aligned} h(x, y, 0) &= \max\left(0, h_0 + \eta(x, y, 0) - b(x, y)\right), \\ u(x, y, 0) &= \eta(x, y, 0)\sqrt{\frac{g}{h_0}} \\ v &= 0, \end{aligned} \tag{6.9}$$

where the parameters are set to $A = 0.064m$, $x_c = 2.5m$, $\gamma = \sqrt{\frac{3A}{4h_0}}$. The bottom topography is a cone and defined by

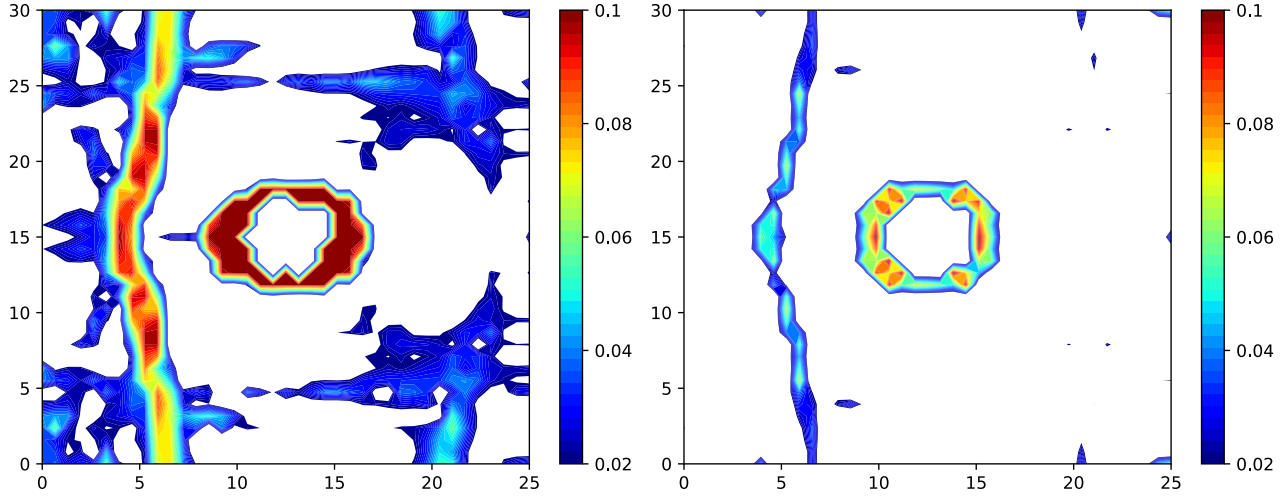
$$b(x, y) = 0.93\left(1 - \frac{r}{r_c}\right) \qquad \text{if } r \leq r_c \tag{6.10}$$

with $r = \sqrt{(x - x_c)^2 + (y - y_c)}$, $r_c = 3.6m$ and $(x_c, y_c) = (12.5, 15)$. The domain Ω is bounded by solid walls everywhere. The test is run up to a final time of $T = 50$. We use a uniform Cartesian mesh with varying spatial resolutions and show the results in Figure 15. The base viscosity parameter is set to be $\epsilon_0 = 0.1$.



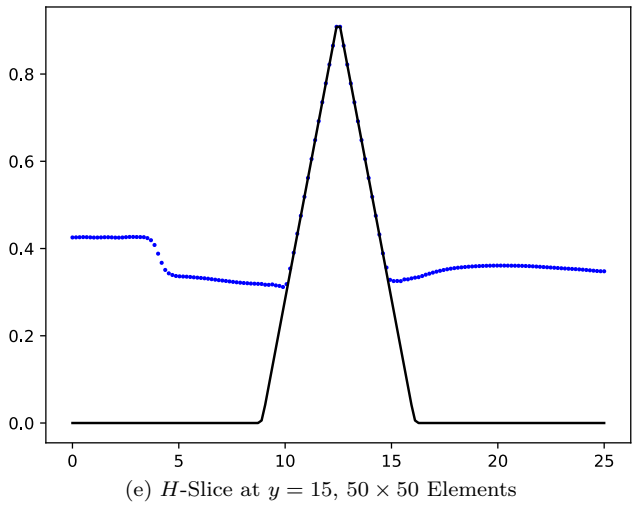
(a) Total water height H, 50×50 Elements

(b) Total water height H, 200×200 Elements

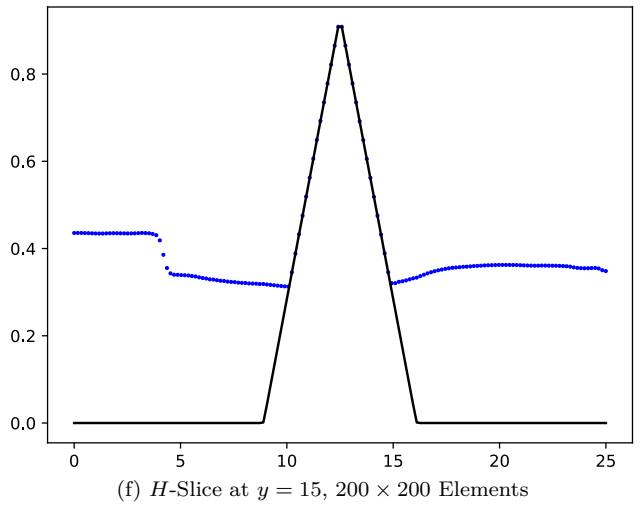


(c) Viscous coefficient ϵ, 50×50 Elements

(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 15: ESDGSEM approximation with artificial viscosity and positivity limiter for the solitary wave runup for different grid resolutions with $N = 3$ at $T = 50$.

6.5. Parabolic Partial Dam Break

In [46] the authors examined the ESDGSEM on curved meshes with a parabolic partial dam break test case. The mesh is shaped such that it aligns with the parabolic dam. We show the initial condition and the mesh in Figure 16. While the results showed increased stability for the entropy stable scheme compared

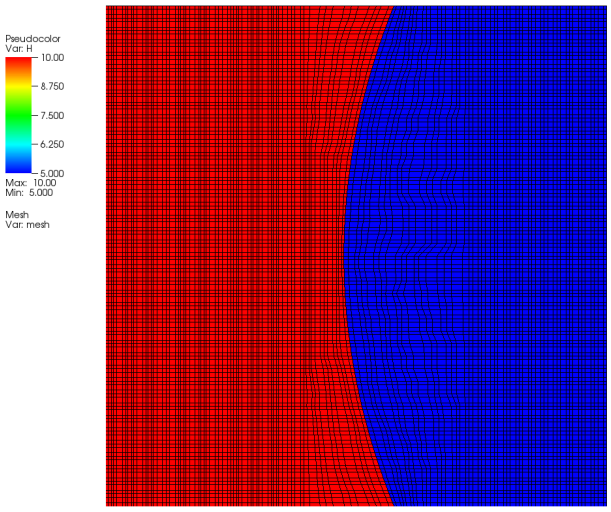
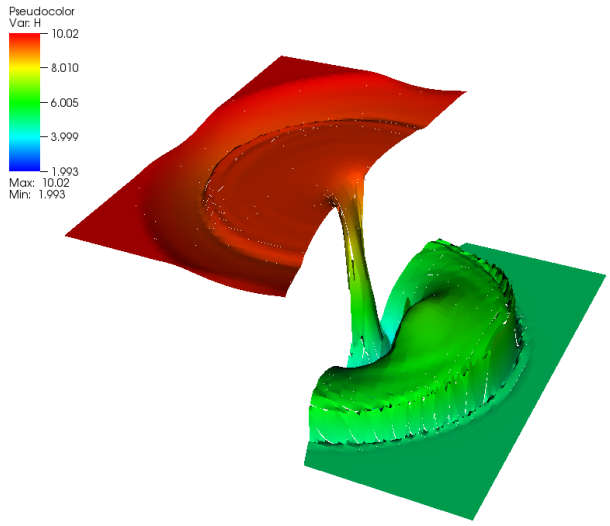


Figure 16: Initial condition and mesh for the parabolic dam break test case.

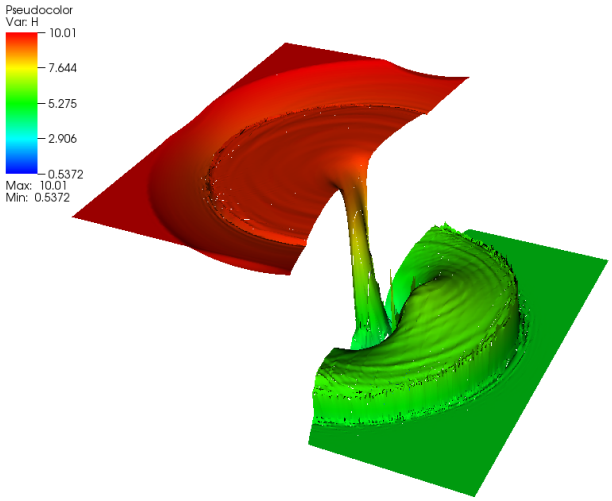
to a standard DGSEM, it also showed that the method suffered under oscillations in the shock region. We repeat this test with the additional dynamic artifical viscosity. The initial setup is given by

$$h = \begin{cases} 10, & \text{if } x < \frac{1}{25}y^2 - 0.25 \\ 5, & \text{otherwise} \end{cases}, \qquad (6.11)$$
$$u = v = 0.$$

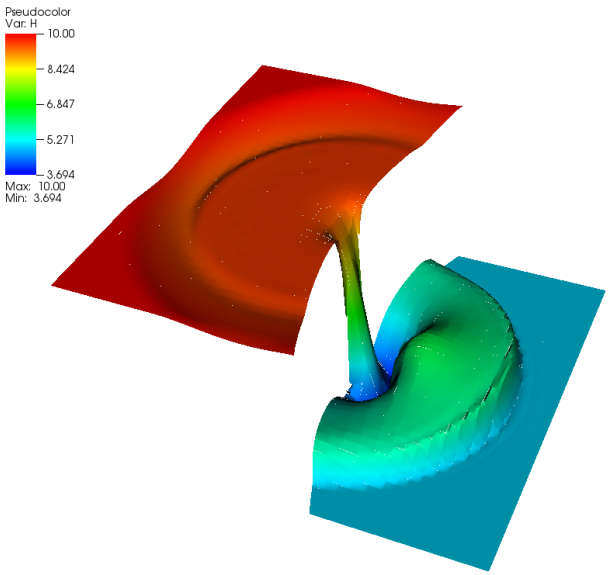
For simplicity, the gravitational constant is set to $g = 1.0$ here. We also remove the discontinuous bottom topography from [46]. While the scheme still works for discontinuous bottoms, the multitude of different effects makes it hard to observe the impact of the artificial viscosity alone. We compare the results for $N = 3$ and $N = 7$ with and without added stabilization and show the the calculated dynamic viscosity coefficients in Figure 17. The base viscosity coefficient is set to $\epsilon_0 = 0.025$. 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 the finer the discretization is. For $N = 7$ not only less elements are affected, but the absolute amount of viscosity is also reduced.



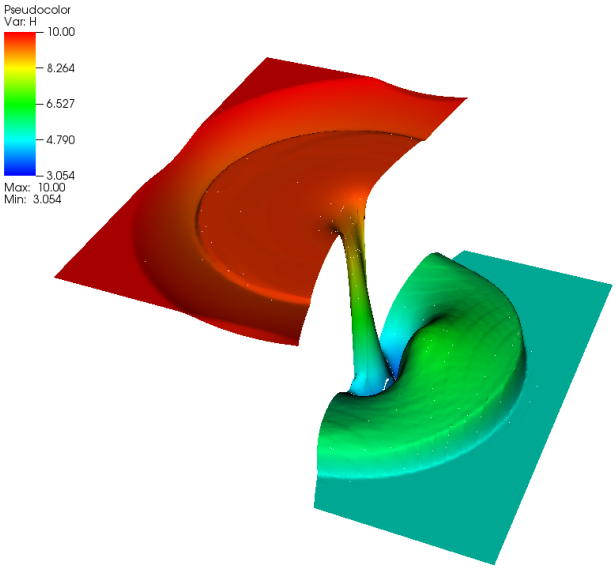
(a) Total water height H without AV, $N = 3$



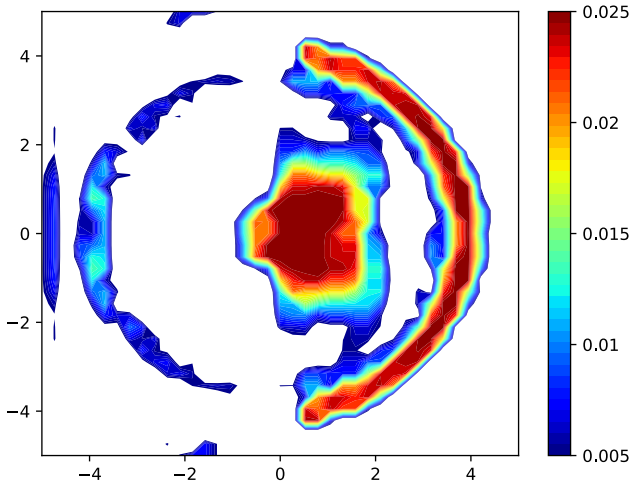
(b) Total water height H without AV, $N = 7$



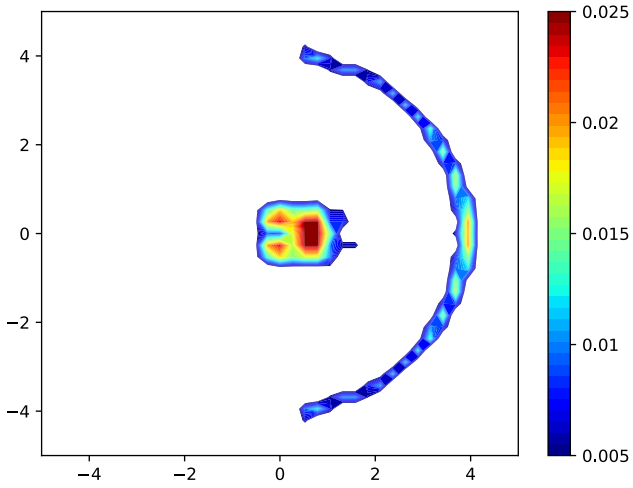
(c) Total water height H with AV, $N = 3$



(d) Total water height H with AV, $N = 7$



(e) Viscous coefficient ϵ, $N = 3$



(f) Viscous coefficient ϵ, $N = 7$

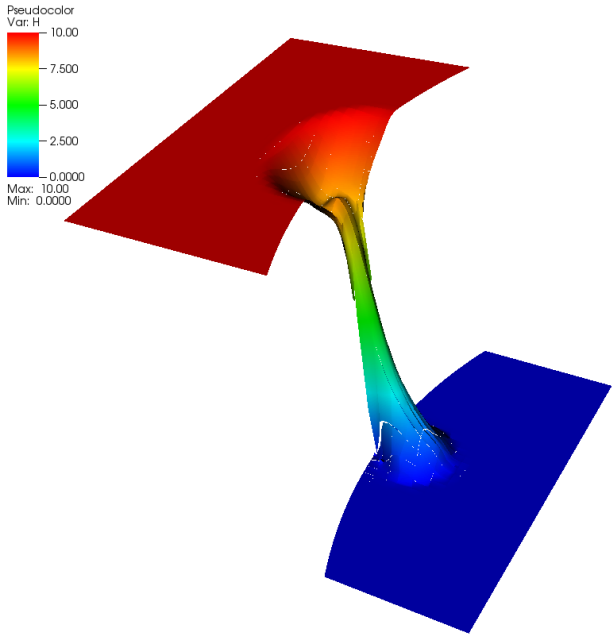
Figure 17: ESDGSEM approximation for the curved dam break with and without artificial viscosity (AV) at $N = 3$ and $N = 7$ on a 40×40 curved mesh at $T = 1.5$.

6.6. Wet/Dry Parabolic Partial Dam Break

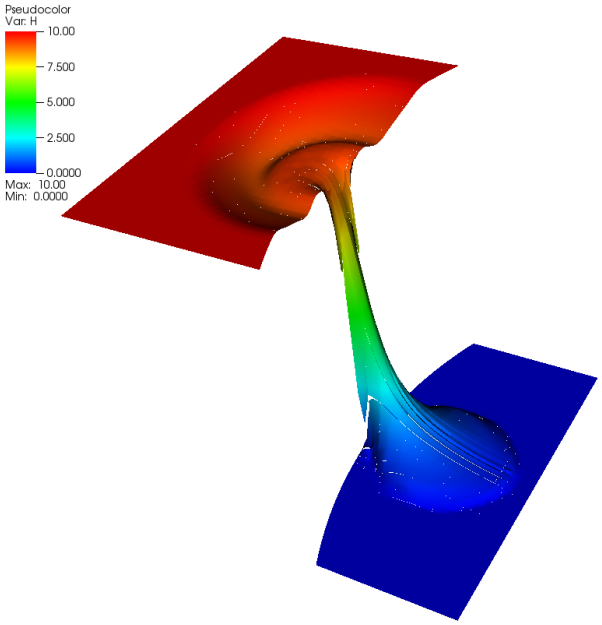
We modify the parabolic dam break problem to feature a dry area on the downstream side. We use the same mesh from Figure 16. It is very challenging as it is a massive shock and thus requires the artificial viscosity as well as the positivity preserving limiter. The initial setup is given by

$$\begin{aligned} h &= \begin{cases} 10, & \text{if } x < \frac{1}{25}y^2 - 0.25 \\ 0, & \text{otherwise} \end{cases}, \\ u &= v = 0. \end{aligned} \tag{6.12}$$

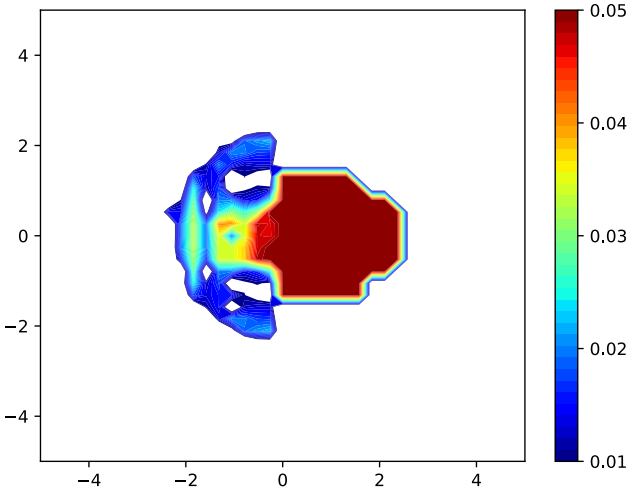
The gravitational constant is set $g = 1.0$ again and the viscosity parameter is set to $\epsilon_0 = 0.05$ for $N = 3$ and to $\epsilon_0 = 0.025$ for $N = 7$. We plot the solution as well as the dynamic viscosity parameter for $N = 3$ in Figure 18 and for $N = 7$ in Figure 19. As the dam break is steeper than in the completely wet case from Section 6.5, the final time is set to $T = 1.0$ so the water does not hit the back wall. The viscous parameter plots again show that only the critical regions are treated with the added artificial viscosity.



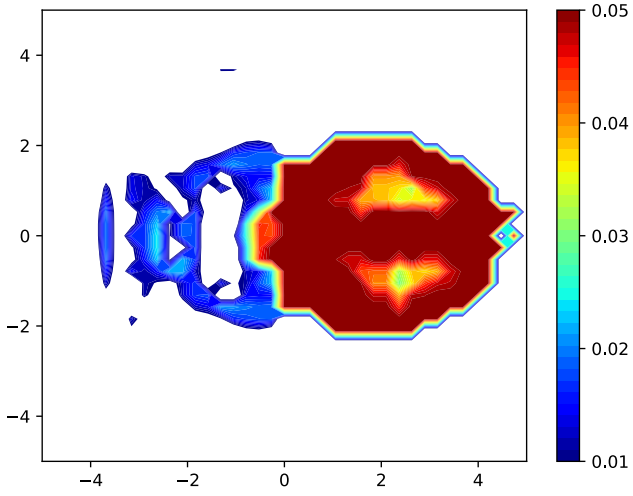
(a) Water height H, $T = 0.5$



(b) Water height H, $T = 1.0$

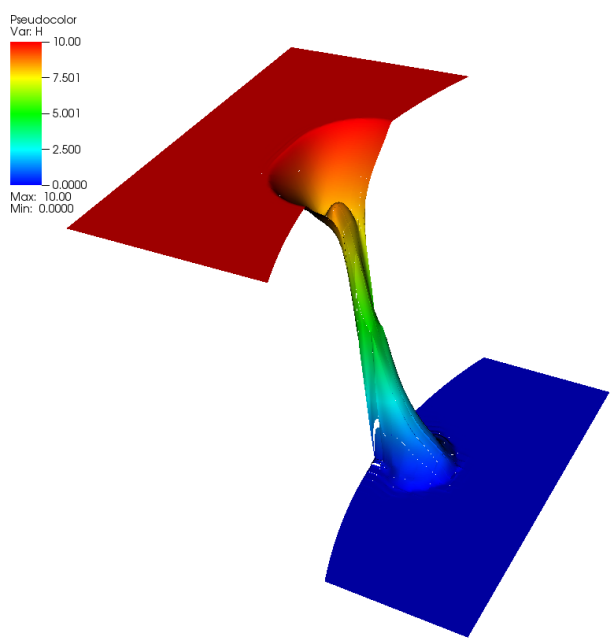


(c) Viscous coefficient ϵ, $T = 0.5$

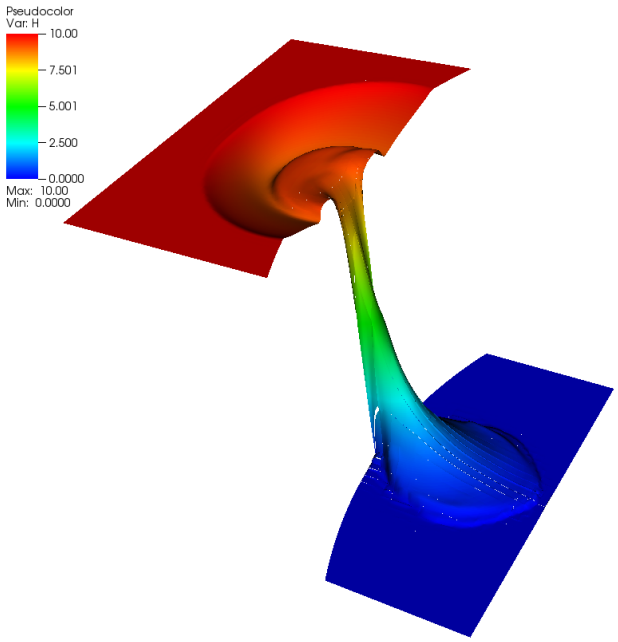


(d) Viscous coefficient ϵ, $T = 1.0$

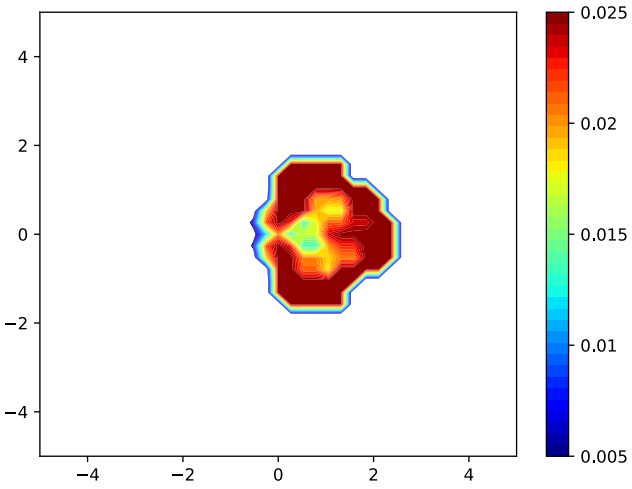
Figure 18: ESDGSEM approximation with artificial viscosity for the curved dam break with zero water height on the downstream side at $N = 3$ on a 40×40 curved mesh.



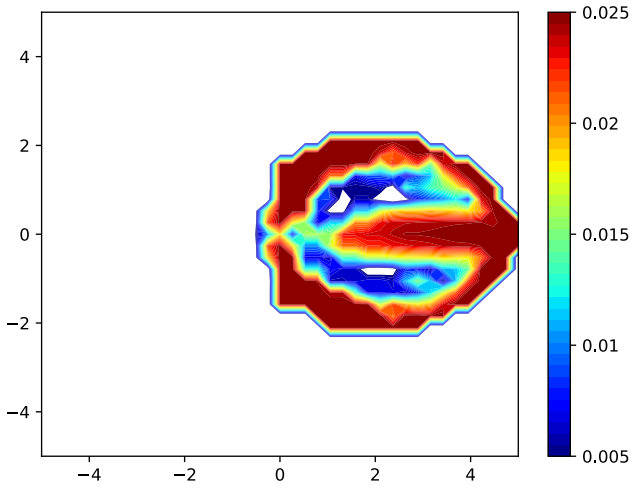
(a) Water height H, $T = 0.5$



(b) Water height H, $T = 1.0$



(c) Viscous coefficient ϵ, $T = 0.5$



(d) Viscous coefficient ϵ, $T = 1.0$

Figure 19: 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.

7. Conclusions

In this work we extended the entropy stable discontinuous Galerkin (DG) spectral element approximation of Wintermeyer et al. [46] to include shock capturing capabilities as well as positivity preservation of the water height such that the numerical scheme can handle wet/dry regions. We demonstrated that these new features, necessary for applications in, e.g., oceanography, did not alter the entropy stable nature of the approximation. Further, we demonstrated that the entropy stable DG discretization for the shallow water equations is well suited for simulations on GPUs. In fact, we found that for polynomial orders of $N \leq 7$ the two methods remained memory bound on GPUs and had nearly identical runtimes.

We then verified the properties of the scheme numerically. Specifically, we found that the entropy stable DG approximation remained conservative and entropy stable even with the additional shock capturing and positivity preserving methods. We also demonstrated that a numerical method which takes the entropy into account is useful to avoid unphysical solutions with an "entropy glitch" test case. Next, we provided five numerical examples to show the utility of the entropy stable, shock capturing, positive water height preserving DG method for problems that feature, among other things, smooth solutions with wet/dry regions, complex multi-shock interactions with bottom topographies or curvilinear element meshes.

Acknowledgements

Gregor Gassner thanks the European Research Council for funding through the ERC Starting Grant "An Exascale aware and Un-crashable Space-Time-Adaptive Discontinuous Spectral Element Solver for Non-Linear Conservation Laws" (Extreme), ERC grant agreement no. 714487.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications.* Springer Verlag, New York, 2008.

[20] 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.

[21] 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.

[22] Andreas Klöckner, Timothy Warburton, and Jan S Hesthaven. Solving wave equations on unstructured geometries. *GPU Computing Gems*, 2:225, 2012.

[23] 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.

[24] David A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers.* Springer Publishing Company, Incorporated, 1st edition, 2009.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] David Medina. *OKL: A unified language for parallel architectures*. PhD thesis, Rice University, 2015.

[30] David S Medina, Amik St-Cyr, and Timothy Warburton. OCCA: A unified approach to multi-threading languages. *arXiv preprint arXiv:1403.0968*, 2014.

[31] 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.

[32] 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.

[33] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. *AIAA Journal*, 112, 2006.

[34] 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.

[35] Benoıt Perthame and Chi-Wang Shu. On positivity preserving finite volume schemes for Euler equations. *Numerische Mathematik*, 73(1):119–130, 1996.

[36] Dang Hieu Phung. Numerical study of long wave runup on a conical island. *VNU Journal of Science, Earth Sciences*, 24:79–86, 2008.

[37] 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.

[38] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988.

[39] 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.

[40] 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.

[41] Costas Emmanuel Synolakis. The runup of solitary waves. *Journal of Fluid Mechanics*, 185:523–545, 1987.

[42] Eitan Tadmor. Numerical viscosity and the entropy condition for conservative difference schemes. *Mathematics of Computation*, 43:369–381, 1984.

[43] Eitan Tadmor. Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems. *Acta Numerica*, 12:451–512, 5 2003.

[44] 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.

[45] G. B. Whitham. *Linear and Nonlinear Waves*. John Wiley and Sons, New York, 1974.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

Appendix A. The viscous parameter ϵ

The viscous parameter ϵ in (3.1) is chosen dynamically for each element dependent on the smoothness of the solution. To get an estimate for the smoothness we transform our nodal DG solution Q to modal space $\hat{Q}$ by

$$\hat{Q}_{ij} = \sum_{i=0}^{N} \sum_{j=0}^{N} V_{ij}^{-1} Q_{ij} V_{ji}^{-1}, \tag{A.1}$$

with Vandermonde matrix $\mathbf{V}$ defined by

$$\begin{aligned} V_{ij} &= L_j(\xi_i^{GL}) \sqrt{j+0.5}, \\ V_{ij}^{-1} &= \left(\ell_j, \tilde{L}_i\right)_{L^2} \approx \sum_{l=0}^{N} L_i(\xi_l^G)\, \ell_j^{GL}(x_i^G)\, \omega_l^G \ \sqrt{i+0.5}, \end{aligned} \tag{A.2}$$

where L_i is the i-the Legendre polynomial, ℓ_i^{GL} the i-th Lagrange polynomial based on Legendre-Gauss-Lobatto nodes and ξ_i^G 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 nodes. We compute shock indicators similar to [33] by

$$\sigma_{dof} = \log_{10}\left(\max\left(\frac{(Q-\tilde{Q}, Q-\tilde{Q})_{L^2}}{(Q,Q)_{L^2}}, \frac{(\tilde{Q}-\tilde{\tilde{Q}}, \tilde{Q}-\tilde{\tilde{Q}})_{L^2}}{(\tilde{Q},\tilde{Q})_{L^2}}\right)\right), \tag{A.3}$$

with

$$\begin{aligned} \tilde{Q} &:= \sum_{i,j=0}^{N-1} \hat{Q}_{ij} \tilde{L}_i \tilde{L}_j, \\ \tilde{\tilde{Q}} &:= \sum_{i,j=0}^{N-2} \hat{Q}_{ij} \tilde{L}_i \tilde{L}_j. \end{aligned} \tag{A.4}$$

With these definitions (A.3) can be simplified to

$$\sigma_{dof} = \log_{10}\left(\max\left(\frac{\sum_{i=0}^{N-1}\left(\hat{Q}_{iN}^2+\hat{Q}_{Ni}^2\right)+\hat{Q}_{NN}^2}{\sum_{i,j=0}^{N}\hat{Q}_{ij}^2}, \frac{\sum_{i=0}^{N-2}\left(\hat{Q}_{i(N-1)}^2+\hat{Q}_{(N-1)i}^2\right)+\hat{Q}_{(N-1)(N-1)}^2}{\sum_{i,j=0}^{N-1}\hat{Q}_{ij}^2}\right)\right). \tag{A.5}$$

This σ_{dof} is used to determine the amount of viscosity applied in every element individually by

$$\epsilon = \begin{cases} 0, & \text{if } \sigma_{dof} \leq \sigma_{min}, \\ \frac{1}{2}\epsilon_0\Delta, & \text{if } \sigma_{dof} \leq \sigma_{min}\ , \\ \epsilon_0, & \text{else.} \end{cases} \tag{A.6}$$

and

$$\Delta := 1.0 + \sin\left(\frac{\pi(\sigma_{dof} - \frac{1}{2}(\sigma_{max}+\sigma_{min}))}{\sigma_{max}-\sigma_{min}}\right). \tag{A.7}$$

Appendix B. Simplification of entropy stable normal numerical h flux

We aim to find a compact expression for the first entry of the numerical flux in normal direction, $\tilde{\vec{F}}^{*,es}$, given by

$$\tilde{F}_1^{*,es}(\vec{W}^+,\vec{W}^-,\vec{n}) = \left(F_1^{*,es}(\vec{W}^+,\vec{W}^-), G_1^{*,es}(\vec{W}^+,\vec{W}^-)\right)\cdot\vec{n}. \tag{B.1}$$

with numerical fluxes in physical x and y direction given by

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

The shallow water equations are rotationally invariant and we can compute the numerical flux in normal direction by rotating the velocities into the new coordinate system and then evaluating the numerical flux in x-direction, $\vec{F}^{*,es}$, with the rotated velocities

$$\begin{aligned} \tilde{u} &:= n_x u + n_y v, \\ \tilde{v} &:= t_x u + t_y v = -n_y u + n_x v. \end{aligned} \tag{B.3}$$

We denote the rotated conservative variables by $\tilde{\vec{W}}$. After computing the x-direction numerical flux we then rotate back into the original coordinate system. Taking everything into account we obtain the following formulas for the numerical fluxes in normal direction

$$\begin{aligned} \tilde{F}_1^{*,es}(\vec{W}^+,\vec{W}^-,\vec{n}) &= F_1^{*,es}(\tilde{\vec{W}}^+,\tilde{\vec{W}}^-) \\ \tilde{F}_2^{*,es}(\vec{W}^+,\vec{W}^-,\vec{n}) &= n_x F_2^{*,es}(\tilde{\vec{W}}^+,\tilde{\vec{W}}^-) + t_x F_3^{*,es}(\tilde{\vec{W}}^+,\tilde{\vec{W}}^-) \\ \tilde{F}_3^{*,es}(\vec{W}^+,\vec{W}^-,\vec{n}) &= n_y F_2^{*,es}(\tilde{\vec{W}}^+,\tilde{\vec{W}}^-) + t_y F_3^{*,es}(\tilde{\vec{W}}^+,\tilde{\vec{W}}^-). \end{aligned} \tag{B.4}$$

To find the numerical flux in the normal direction, we now evaluate the numerical flux in x-direction, $\vec{F}^{*,es}$, using rotated rotated conservative variables, $\tilde{\vec{W}}$,

$$\vec{F}^{*,es}(\tilde{\vec{W}}^+,\tilde{\vec{W}}^-) = \begin{pmatrix} \{\{h\}\}\,\{\{\tilde{u}\}\} \\ \{\{h\}\}\,\{\{\tilde{u}\}\}^2 + \frac{1}{2}\,g\,\{\{h^2\}\} \\ \{\{h\}\}\,\{\{\tilde{u}\}\}\,\{\{\tilde{v}\}\} \end{pmatrix} - \frac{1}{2}\tilde{\mathbf{R}}\left|\tilde{\mathbf{\Lambda}}\right|\tilde{\mathbf{R}}^T\left[\!\left[\,\tilde{\vec{q}}\,\right]\!\right], \tag{B.5}$$

with matrix of right eigenvectors

$$\tilde{\mathbf{R}} = \begin{pmatrix} 1 & 0 & 1 \\ \{\{\tilde{u}\}\} + \{\{c\}\} & 0 & \{\{\tilde{u}\}\} - \{\{c\}\} \\ \{\{\tilde{v}\}\} & 1 & \{\{\tilde{v}\}\} \end{pmatrix}, \tag{B.6}$$

and scaled diagonal eigenvalue matrix

$$|\tilde{\mathbf{\Lambda}}| = \frac{1}{2g} \begin{pmatrix} |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,| & 0 & 0 \\ 0 & 2g|\,\{\{h\}\}\,\{\{\tilde{u}\}\}\,| & 0 \\ 0 & 0 & |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,| \end{pmatrix}, \tag{B.7}$$

with wave speed $c = \sqrt{gh}$. We compute the first row of the matrix product of the dissipation term by multiplying the first row $\tilde{\mathbf{R}}_1$

$$\begin{aligned} &2g\tilde{\mathbf{R}}_1 \,|\tilde{\mathbf{\Lambda}}|\, \tilde{\mathbf{R}}^T \\ &= \begin{pmatrix} 1, & 0, & 1 \end{pmatrix} \begin{pmatrix} |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,| & 0 & 0 \\ 0 & 2g|\,\{\{h\}\}\,\{\{\tilde{u}\}\}\,| & 0 \\ 0 & 0 & |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,| \end{pmatrix} \begin{pmatrix} 1 & \{\{\tilde{u}\}\} + \{\{c\}\} & \{\{\tilde{v}\}\} \\ 0 & 0 & 1 \\ 1 & \{\{\tilde{u}\}\} - \{\{c\}\} & \{\{\tilde{v}\}\} \end{pmatrix} \\ &= \begin{pmatrix} |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,|, & 0, & |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,| \end{pmatrix} \begin{pmatrix} 1 & \{\{\tilde{u}\}\} + \{\{c\}\} & \{\{\tilde{v}\}\} \\ 0 & 0 & 1 \\ 1 & \{\{\tilde{u}\}\} - \{\{c\}\} & \{\{\tilde{v}\}\} \end{pmatrix} \\ &= \begin{pmatrix} A, & \{\{\tilde{u}\}\}\, A + \{\{c\}\}\, B, & \{\{\tilde{v}\}\}\, A \end{pmatrix}, \end{aligned} \tag{B.8}$$

with

$$\begin{aligned} A &:= |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,| + |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,|, \\ B &:= |\,\{\{\tilde{u}\}\} + \{\{c\}\}\,| - |\,\{\{\tilde{u}\}\} - \{\{c\}\}\,|. \end{aligned} \tag{B.9}$$

Multiplying by $[\![\,\vec{\tilde{q}}\,]\!]$ we find the first entry of $\frac{1}{2}\tilde{\mathbf{R}}\,|\tilde{\mathbf{\Lambda}}|\,\tilde{\mathbf{R}}^T\,[\![\,\vec{\tilde{q}}\,]\!]$

$$\begin{aligned} 2g\left(\tilde{\mathbf{R}}\,|\tilde{\mathbf{\Lambda}}|\,\tilde{\mathbf{R}}^T\,[\![\,\vec{\tilde{q}}\,]\!]\right)_1 &= \begin{pmatrix} A, & \{\{\tilde{u}\}\}\, A + \{\{c\}\}\, B, & \{\{\tilde{v}\}\}\, A \end{pmatrix} \begin{pmatrix} g\,[\![h+b]\!] - \frac{1}{2}\,[\![\tilde{u}^2]\!] - \frac{1}{2}\,[\![\tilde{v}^2]\!] \\ [\![\tilde{u}]\!] \\ [\![\tilde{v}]\!] \end{pmatrix} \\ &= A\,(g\,[\![h+b]\!] - \{\{\tilde{u}\}\}\,[\![\tilde{u}]\!] - \{\{\tilde{v}\}\}\,[\![\tilde{v}]\!]) + (\{\{\tilde{u}\}\}\, A + \{\{c\}\}\, B)\,[\![\tilde{u}]\!] + \{\{\tilde{v}\}\}\, A\,[\![\tilde{v}]\!] \\ &= gA\,[\![h+b]\!] + \{\{c\}\}\, B\,[\![\tilde{u}]\!]\,. \end{aligned} \tag{B.10}$$

We can find bounds on A and B by

$$\begin{aligned} 2\lambda_{\max} = 2\max\left(|\tilde{u}| + |c|\right) \geq 2(|\,\{\{\tilde{u}\}\}\,| + |\,\{\{c\}\}\,|) \geq A \geq 2|\,\{\{\tilde{u}\}\}\,| \\ \lambda_{\max} = \max\left(|\tilde{u}| + |c|\right) \geq (|\,\{\{\tilde{u}\}\}\,| + |\,\{\{c\}\}\,|) \geq B \geq -|\,\{\{\tilde{u}\}\} - \{\{c\}\}\,| \geq -\lambda_{\max}. \end{aligned} \tag{B.11}$$

Overall, we can express the first entry of the entropy stable numerical flux in terms of the rotated velocities

$$F_1^{*,es}(\vec{\tilde{W}}^+, \vec{\tilde{W}}^-) = \{\{h\}\}\,\{\{\tilde{u}\}\} - \frac{1}{4g}\,(A\,[\![gh + gb]\!] + \{\{c\}\}\, B\,[\![\tilde{u}]\!])\,. \tag{B.12}$$

Appendix C. Versioning of OCCA Optimized Volume Integral Kernels

- 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 the constant values $\frac{1}{2}g$ and $\frac{1}{4}g$ as kernel infos at the start of the program. We also introduce an additional shared memory array that stores the inverse of the water height $1/h$ which is used in the flux calculations.

- Version 3: Multiple Elements per Block

  One GPU thread block is typically able to handle multiple DG elements in parallel. We aim to make full use of the GPU compute power and reduce the number of idle threads. Thus we introduce a parameter NE_{Block} that sets the number of elements handled in one thread block. This number obviously depends on the polynomial order and typically decreases with N. Unfortunately 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

  We need to make sure that memory access is aligned as much as possible. 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[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.

- Version 5: Avoid Bank Conflicts, add Padding

  To avoid bank conflicts 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. We hope that this potentially hides loads time 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.