DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Tsunami simulation in the ExaHyPE framework

Lisa Scheller

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Tsunami simulation in the ExaHyPE framework

Tsunamisimulation innerhalb des ExaHyPE-Frameworks

Author:	Lisa Scheller
Supervisor:	Prof. Dr. Michael Bader
Advisor:	MSc Leonhard Rannabauer
Submission Date:	15.12.2017

I confirm that this bachelor's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15.12.2017 Lisa Scheller

Acknowledgments

Abstract

This bachelor's thesis gives an overview over hyperbolic partial differential equations, especially the shallow water equations. Several numerical methods for solving these kind of problems are described and compared, focusing on finite volume methods and discontinuous Galerkin methods. Since for those methods numerical flux functions are used, some of them are also explained. An analytic solution for dam break problems is developed for later error computing and convergence analysis. Then the implementation is described in detail, as well as which scenarios were used and what results were computed. This is then used to compare finite volume and discontinuous Galerkin approaches. After that, the error of the methods is computed and the order of convergence of the method determined. The methods are then compared in regard to this.

List of Abbreviations

ODE ordinary differential equation

PDE partial differential equation

Nomenclature

Δx	Cell size, $x_{i+1/2} - x_{i-1/2}$
$f'(q)$	Jacobi matrix of $f(q)$
q	vector of unknowns
q_x	partial derivative $\frac{\delta q}{\delta x}$

Contents

1. Introduction

This thesis gives an overview over the usage of finite volume methods and discontinuous Galerkin methods for solving hyperbolic partial differential equations, especially for the one-dimensional shallow water equations with discontinuous initial conditions, the so called dam break problem. The shallow water equations are an important member of hyperbolic PDEs and are used in a lot of hydrodynamic problems. Also while there are known analytic solutions for many one dimensional initial conditions, which simplifies the error computing process, the solutions also contain the main obstacles that present when trying to solve a hyperbolic PDE numerically. Thus they are well suited to test the ability of a solving method to handle those PDEs. This knowledge can then be used for the approximation of more difficult PDEs with no known analytic solutions or solving higher-dimensional problems.

First an overview over hyperbolic PDEs and especially the shallow water equations is given. Since the solving of Riemann problems is vital to solve a dam break problem, this concept will be explained. The next chapter includes an overview over several numerical methods to solve hyperbolic PDEs and list their advantages and disadvantages. Finite volume and discontinuous Galerkin methods will be covered in more detail. Also, some numerical flux functions will be introduced. They are later used in the implementation. For discontinuous Galerkin methods, the difference between the nodal and modal form will be explained. The implementation uses only a nodal approach. A short overview over some numerical integration methods is given. The remaining part of the chapter is used to explain how a dam break problem can be solved analytically. The following chapter contains the details of the implementation, which scenarios were used, what results were obtained for finite volumes as well as discontinuous Galerkin methods and how the methods compare to each other. Finally the error computing methods will be described and the error analyzed and compared to the theoretically expected error.

2. The shallow water equations

2.1. Hyperbolic partial differential equations

2.1.1. Definition

Partial differential equations (PDEs) are used to describe a multitude of problems. In contrast to ordinary differential equations (ODEs) PDEs contain partial derivatives, whereas ODEs only contain derivatives of one variable. There are a multitude of phenomenons that can be described by PDEs, ranging from fluid mechanics and traffic simulation over thermodynamic problems to quantum mechanics. In this bachelor's thesis we restrict ourselves to a certain subclass of PDEs, the so-called hyperbolic PDEs. The core equations of this thesis, the shallow water equations, which will be introduced in detail in the next section,as well as the standard advection equation, are hyperbolic PDEs.

Definition 2.1.1. "A linear system of the form

$$q_t + Aq_x = 0$$

is called hyperbolic if the $m \times m$ matrix A is diagonizable with real eigenvalues."[Lev04, p. 31]

This is a one-dimensional, homogenous, first-order system of PDEs. q is a vector from $\mathbb{R}^m$ that contains the unknowns of the equations and A a $m \times m$ matrix. The restriction that the matrix has to be diagonizable means that it must be possible to describe A in the following way:

$$A = S \cdot D \cdot S^{-1} \tag{2.1}$$

where D is a diagonal matrix, and S is a basis transformation onto the basis defined by all eigenvectors of A.
Other types of PDEs are elliptic or parabolic PDEs.

2.1.2. Classification

There are several sub-groups of hyperbolic PDEs, in regard to the eigenvalues of A as well as in regard to the structure of the equations.

Eigenvalues

Since a symmetric matrix is always diagonizable with real eigenvalues, all symmetric matrices can be used to describe a hyperbolic problem [Lev04]. These equations are called symmetric hyperbolic.
If the matrix has distinct eigenvalues and is diagonizable, the system of PDEs is called strictly hyperbolic. [Lev04]
In contrast, a matrix that is not diagonizable but has real eigenvalues is called weakly hyperbolic.[Lev04]

Structure of equations

In the simplest case of a hyperbolic PDE A is not a matrix, but a constant scalar. Equation (**??**) simplifies to

$$q_t + aq_x = 0 \tag{2.2}$$

and a has to be real for the system to be hyperbolic.
If the scalar a or the matrix A are time-dependent or depend on another value (e.g. $a(x,t)$, $A(x,p)$), they have to fulfill the criteria for a system of hyperbolic PDEs for each value of t (or in the examples also of x and p).

Conservation laws A special group of hyperbolic PDEs are called conservation laws. In the simplest case (1 dimension) they have the following structure:

$$q(x,t)_t + f(q(x,t))_x = 0 \tag{2.3}$$

or in the quasilinear form

$$q_t + f'(q)q_x = 0 \tag{2.4}$$

[Lev04] $f(q)$ can be linear as well as non-linear in regard to q.

The integral formulation This formulation can also be used to describe a conservation law, in fact it is the more fundamental form to describe one because it also holds across discontinuities.

The integral formulation of a hyperbolic conservation law looks as follows:

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)\,\mathrm{d}x = F(x_2,t) - F(x_1,t) \tag{2.5}$$

$f(x,t)$ represent fluxes going either into or out from the element. The differential formulation ((2.3)), which is used mostly, can be derived from the integral formulation under the assumption that q is smooth. [Lev04]

Weak solutions However, to be able to treat naturally occurring discontinuities, we need to extend the native definition of a solution by the weak formulation. This leads (detailed description [Lev04, p.215]) to the following equation:

$$\int_0^\infty \int_{-\infty}^\infty [q\Phi_t + f(q)\Phi_x]dxdt = -\int_0^\infty q(x,0)\Phi(x,0)dx \tag{2.6}$$

Definition 2.1.2. "The function $q(x,t)$ is a weak solution of the conservation law $q_t + f(q)_x = 0$ with given initial data $q(x,0)$ if (2.6) holds for all functions ϕ in C_0^1."

[Lev04, p.215] Here C_0^1 describes all functions with compact support that are continuously differentiable.

2.1.3. Examples

Only very few hyperbolic PDEs are easy to solve. Their complexity ranges from one-dimensional, linear equations to multidimensional nonlinear PDEs. For most of these problems, especially if they are nonlinear or multidimensional, no analytic solution are known. One example of a very basic hyperbolic PDE is the one-dimensional advection equation with constant speed:

$$q_t + uq_x = 0 \tag{2.7}$$

where u is the constant velocity. This equation is solved by

$$q(x,t) = q_0(x - ut) \tag{2.8}$$

where $q_0(x) = q(x,0)$ are the known initial conditions.

Even though this equation is very simple, its structure still resembles that of more complex problems. So its understanding plays a role at solving more difficult problems.[Lev04]
Another, more complex example are the Maxwell equations

$$\begin{aligned} E_t &= \nabla \times B \\ B_t &= -\nabla \times E \end{aligned} \tag{2.9}$$

([Stu13]), which are used to describe electromagnetic fields.
In general, most problems which are solved by waves or some superposition of waves are described by hyperbolic PDEs. Of course this also includes the description of water waves.

2.1.4. Numerical difficulties

The main problem with solving hyperbolic PDEs is to correctly handle the discontinuities that may arise in the solutions. While weak solutions can be analytically found with the criteria listed above, not all weak solutions solve the problem physically correct [Lev04, p.217]. To dismiss those incorrect solutions, the numerical methods often make use of entropy conditions. Also it is usually not possible to simply use the differential formulation and assume it will be sufficiently smooth. Another problem is that the solutions often can be described as several waves of different types. The composition of types depends on the problem and has to be computed first. If the waves interact which each other, it complicates the solution further.

2.2. The shallow water equations

This thesis focuses on solving the shallow water equations, which describe water height and momentum of a fluid, in one dimension. For the shallow water equations to be applicable the vertical scale has to be much smaller than the horizontal scale. This is true for ocean waves (period, small wavelength) and even tsunamis (caused by singular event, long wavelength, small amplitude while in in deep water, high altitude in shallow water), which are also described by the shallow water equations.

2.2.1. Equations in one dimension

The one-dimensional shallow water equations consist of two equations. The first one is:

$$h_t + (uh)_x = 0 \tag{2.10}$$

and the second one is:

$$(hu)_t + (hu^2 + \frac{1}{2}gh^2)_x = 0 \tag{2.11}$$

If $q(\vec{x}, t) = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} h \\ hu \end{pmatrix}$ is introduced and $f(q) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix} = \begin{bmatrix} q_2 \\ (q_2)^2/q_1 + \frac{1}{2}gq_1 \end{bmatrix}$ is defined, then the equations can be written as

$$q_t + f(q)_x = 0 \tag{2.12}$$

For smoot solutions, the formulation

$$q_t + f'(q)q_x = 0 \tag{2.13}$$

is also valid.[Lev04]

2.2.2. Properties

Since the eigenvalues and eigenvectors of $f'(q)$ are defining the entire system of equations, they are also basic properties of the problem. They have to be known to be able to compute a numeric or exact solution to any problem. They are given by:

$$\lambda_1 = u - \sqrt{gh} \tag{2.14}$$

and

$$\lambda_2 = u + \sqrt{gh} \tag{2.15}$$

This leads to the eigenvectors

$$\vec{v_1} = \begin{pmatrix} 1 \\ \lambda_1 \end{pmatrix} \tag{2.16}$$

and

$$\vec{v_2} = \begin{pmatrix} 1 \\ \lambda_2 \end{pmatrix} \tag{2.17}$$

[Lev04].
If h is zero, both eigenvalues and thus the eigenvectors have the same value. Because the eigenvalues (and thus the eigenvectors) depend on h and u, which are the unknowns of the equation system and can be different for every point in the domain, every of these points may have different eigenvalues. It is not sufficient to compute one set of eigenvalues at the beginning and then never change it again, like with the constant velocity advection equation. The eigenvalues have to be computed for each point or cell in the domain and for each time step, otherwise the numerical solution disregards the distribution in space as well as the time-development.

2.2.3. Example problems

We will look at two well known problems for the Shallow Water Equations. The first one has continuous initial conditions and is the already described Gaussian water hump. The second one is the dam break problem, a so called Riemann problem with initial velocities zero and a difference in initial height.

Riemann problems

Definition 2.2.1. A problem with inital conditions of the form

$$q(x,0) = \begin{cases} q_l & x \leq 0 \\ q_r & 0 \leq x \end{cases}$$

is called a Riemann problem[Lev04].

The main characteristic of a Riemann problem is the discontinuity. To solve such a problem analytically, it has to be fragmented into several states. Two of them are the outer left and right states, where the solution takes the value q_l or q_r respectively. The middle states depend on the exact composition of the problem, the possibilities for the cases viewed here are either 2 shock waves, 2 rarefaction waves or one of each. Also a contact discontinuity often exists. In every case a middle value q_m has to be computed. The exact way to solve a Riemann problem as well as an explanation of the different types of waves will be given in chapter 3. The analytic solution for a dam-break problem will also be computed.

2.2.4. Two-dimensional equations

In two dimension the shallow water equations can be written as

$$\begin{aligned} h_t + (hu)_x + (hv)_y &= 0 \\ (hu)_t + (hu^2 + \frac{1}{2}gh^2)_x + (huv)_y &= 0 \\ (hv)_t + (huv)_x + (hv^2 + \frac{1}{2}gh^2)_y &= 0 \end{aligned} \tag{2.18}$$

[Lev04] This differs from the one-dimensional form by the addition of a additional dimension y. Also the velocity is now described by a vector, $\vec{u} = \begin{pmatrix} u \\ v \end{pmatrix}$. The one-dimensional equations can be derived from the two-dimensional case by setting one of the velocities to zero. The two-dimensional equations can be solved using similar techniques as for the one-dimensional case, however for the sake of complexity this thesis will only cover the one-dimensional equations.

3. Methods for solving hyperbolic partial differential equations

There are a multitude of numerical methods to solve PDEs. However, some of them are more suited to the task of solving hyperbolic PDEs than others. As pointed out earlier some methods become unstable along discontinuities. So it's important to choose a method that handles these discontinuities well if a hyperbolic PDE has to be solved numerically.

3.1. Finite Differences

3.1.1. General description

The finite differences method is one of the better known numerical schemes. It approximates derivatives by replacing them with a difference quotient of the form

$$\frac{u(x+h) - u(x)}{h} \tag{3.1}$$

and solving the resulting equation for $u(x+h)$. The result can then be integrated by using an integration scheme such as explicit Euler etc. The domain is divided into intervals of length h, for which the equation is evaluated. Higher accuracy can be achieved by choosing a smaller h or using a higher order integration scheme such as Runge-Kutta, however this will also impact the definition of the difference quotient.

The computational domain is divided so that the method can be evaluated at certain predefined grid points.

3.1.2. Advantages and disadvantages

Advantages The method is easy to understand and since only one value is computed per interval it can also be very efficient[JSH08]. Also it is possible to modify the scheme to get high order accuracy.

Disadvantages However since the finite difference becomes quite nonsensical at a discontinuity it is not well suited for solving hyperbolic PDEs. Also it doesn't simply allow more complex geometry because it only supports equidistant intervals in its standard form [JSH08]. While schemes with non-equidistant intervals exist, they are quite convoluted. This becomes even more of a problem for solving higher dimensional problems.

3.1.3. Example

3.2. Finite Volumes

3.2.1. General description

The main difference between the method of finite volumes and the method of finite differences is that the first is based on the integral form, whereas the latter is based on the differential form. Using the integral form makes it possible to handle discontinuities. The computational domain is divided in cells, and for each cell the integral over this cell is approximated. For each cell, the cell average is computed, which is the quotient of the approximated integral and the cell volume [Lev04]. This means that each cell stores only this average and not an approximation polynomial or an interpolation polynomial.

To evaluate the change over time now also the fluxes from the neighbor cells have to be considered. There are several possibilities to compute those fluxes, some of which will be explained later.

From now on, the description will be restricted to only one dimension.
The basic idea is to describe the ith grid cell as

$$C_i = (x_{i-1/2}, x_{i+1/2}) \tag{3.2}$$

[Lev04] The cell average of the ith cell in the nth iteration can now be approximated as:

$$Q_i^n = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} q(x, t_n) dx = \frac{1}{\Delta x} \int_{C_i} q(x, t_n) dx \tag{3.3}$$

[Lev04], where Δx is the cell size.
If this is applied to the integral formulation of a conservation law, it leads to the following equation:

$$\frac{d}{dt} \int_{C_i} q(x,t) dx = f(q(x_{i-1/2}, t)) - f(q(x_{i+1/2}, t)) \tag{3.4}$$

Since the goal is to propagate the degrees of freedom in time and thus compute the values Q_i^{n+1} at time t^{n+1}, the above equation has to be time-integrated. If the result of this is then divided by Δx to get the cell average it leads to an approximation of exactly this.

$$\frac{1}{\Delta x}\int_{C_i} q(x,t^{n+1})dx = \frac{1}{\Delta x}\int_{C_i} q(x,t^{n})dx - \frac{1}{\Delta x}[\int_{t^n}^{t^{n+1}} f(q(x_{i+1/2},t))dt - \int_{t^n}^{t^{n+1}} f(q(x_{i-1/2},t))dt] \tag{3.5}$$

[Lev04]. While this equation would compute exactly what we want it to compute, it can't normally be used because the time integrals on the right side are usually not known. Usually the time integrals over Δt are approximated by some numerical flux in the time interval Δt, which leads to schemes of the following form:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(F_{i+1/2}^n - F_{i-1/2}^n) \tag{3.6}$$

F is a approximation of the time integrals in (3.5). The choice of this flux approximation heavily impacts the quality of the approximated solution. $F_{i-1/2} = F((Q_{i-1}, Q_i))$ is called a numerical flux function and it can now be seen that the approximation of Q_i^{n+1} depends on Q_{i-1}, Q_i and Q_{i+1}, a so called three point stencil [Lev04].

3.2.2. Advantages and disadvantages

Advantages One of the main benefits of the method of finite volumes is that it is able to handle problems with discontinuities. Also it is possible to use varying cell sizes, which is useful for more complex geometries. Since the solution depends only on the inital conditions and the fluxes one of the main tasks is to use the flux functions most well-suited for the specific problem. This depends on required accuracy, stability and also on the properties of the problem itself.

Disadvantages Since only the cell average is stored for each cell, the approximation inside of one cell is quite rudimentary. It would be more accurate to give the cell more information, e.g. a value at the left boundary and the right boundary of the cell as well as a interpolation function to link those values. The inaccuracy of the cell average can of course be lessened by making the cells smaller, but it is still not as accurate as for example the discontinuous Galerkin method, which will be covered later.
Also if a higher order of accuracy shall be reached, the restrictions on the grid come back again, as described in [JSH08]. This means the method loses one of it's main advantages. If these disadvantages are a problem for the specific problem that should

be solved, and a finite difference scheme is also not the right solution, it might be necessary to take a look at finite elements methods.

3.2.3. The CFL condition

The stability of a finite volume method depends heavily on the chosen time step. If it is to big, the changed information has traveled too far and so the solution depends not only on the neighboring cells but on their neighbors etc. as well. Because those are not part of the numerical scheme the results will become increasingly wrong with time. So a condition to determine the time step size is necessary.
The Courant-Friedrichs-Lewy-condition (CFL-condition) is such a condition.

Definition 3.2.1. "A numerical method can be convergent only if its numerical domain of dependence contains the true domain of dependence of the PDE, at least in the limit as Δt and Δx go to zero"[Lev04]

The CFL condition is a necessary condition for stability and convergence. It leads to the conclusion that the Courant number

$$v = \frac{\Delta t}{\Delta x} max|\lambda^p| \tag{3.7}$$

, where the λ^p are the wave speeds of the system of equations, has to stay below a certain bound, usually 1 for the finite volume methods we use. From this equation the biggest time step that still leads to a stable solution can be computed by solving it for Δt. Depending on the problem (for problems with variable wave speeds) it is not possible to use a stable time step, since it would become bigger than the maximum allowed time step over time and so the time step for the next time iteration has to computed from the wave speeds at the respective time.

3.2.4. Flux approximations

The choice of the flux function is a major part of the finite volumes scheme. The following section will give short overview over some flux functions. Most of these fluxes where also implemented in the program.

Unstable Flux This method follows a very simple approach. The flux function is approximated by

$$F^n_{i-1/2} = \frac{1}{2}[f(Q^n_{i-1}) + f(Q^n_i)] \tag{3.8}$$

This would lead to

$$Q^{n+1}_i = Q^n_i - \frac{\Delta t}{2\Delta x}[f(Q^n_{i+1}) - f(Q^n_{i-1})] \tag{3.9}$$

However, this approach is unusable because it is unstable for hyperbolic problems and all timestep sizes [Lev04].

Lax-Friedrichs flux This flux looks very similar to the unstable flux, but it is different enough that it leads to a stable result. It has the advantage that it is still a very simple method that is easy to implement. The flux function is

$$F_{i-1/2} = \frac{1}{2}[f(Q_{i-1}^n) + f(Q_i^n)] - \frac{\Delta x}{2\Delta t}(Q_i^n - Q_{i-1}^n) \tag{3.10}$$

This can then be used in (3.6) to compute the value of Q_i at the next time step.
While the Lax-Friedrichs flux leads to a stable scheme the order of accuracy still isn't too good, as will be outlined later when the results of the Shallow Water Equations computed with this flux will be compared to the analytic solution.

Richtmyer Two-Step Lax–Wendroff Method A better order of accuracy can be achieved by evaluating q at time $t + \frac{1}{2}\Delta t$ and using this value to compute the fluxes at time $t + \Delta t$. However, this method is less needs more computational power than the one-step methods, because for each time step two values of Q have to be computed.

The local Lax-Friedrichs-Flux The main difference between the local Lax-Friedrichs method is that the factor $\frac{\Delta x}{\Delta t}$ in the computation of the fluxes is now replaced with another value. In particular this value is not fixed, but changes for every cell and every time. The method for determining the fluxes becomes

$$F_{i-1/2} = \frac{1}{2}(f(Q_{i-1}) + f(Q_i) - a_{i-1/2}(Q_i - Q_{i-1})) \tag{3.11}$$

and the factor a is determined by

$$a_{i-1/2} = max(|f(Q_{i-1}|, |f(Q_i)|) \tag{3.12}$$

For the shallow water equations this leads to the biggest eigenvalue (which are $u \pm \sqrt{gh}$) at both indexes (i and i-1 for the left side and i and i+1 for the right side). Using the local Lax-Friedrichs flux leads to better results especially concerning rarefaction waves [Lev04]. This method is also known as Rusanov's method. The name "local Lax-Friedrichs method" is based on the similarity to the Lax Friedrichs method and the choice of different factors for each cell, which makes the method local.

Upwind methods

3.3. Finite Elements

3.3.1. General description

The method of finite elements divides the domain in several elements. Even if the general geometry is very complex, the geometry of the elements is a lot simpler and makes it possible to approximate the solution inside of this elements. The solution inside the elements is approximated by local basis functions [JSH08]. The approximated solution is easier to compute for a lower degree of these basis functions, but the accuracy increases with the degree of those functions. A popular choice for the basis functions are the Lagrange polynomials,

$$\ell_i^k(x) = \frac{x - x^{k+1-i}}{x^{k+1} - x^{k+1-i}} \tag{3.13}$$

It is of course possible to choose other interpolation polynomials.
The global formulation is retrieved by choosing a set of test functions that are orthogonal to the residual of the problem. For a Galerkin scheme the test functions span the same spaces as the basis functions. Other choices of test functions are possible. For the standard advection equation this leads to a scheme of the form

$$M\frac{du_h}{dt} + Sf_h = 0 \tag{3.14}$$

where

$$M_{ij} = \int_\Omega N^i(x)N^j(x)dx \tag{3.15}$$

is the globally defined mass matrices (N^i are the basis functions) and

$$S_{ij} = \int_\Omega N^i\frac{dN^j}{dx}dx \tag{3.16}$$

is the globally defined stiffness matrix [JSH08].

3.3.2. Advantages and disadvantages

Advantages This approach has the advantage that it is possible to choose different orders of approximation in different cells [JSH08] and so adapt to the specifications of the problem. Also the choice of the elements size and geometry is very flexible.

Disadvantages However, the above choice of basis functions and the requirements they must fulfill means that the matrix M has to be inverted and the scheme becomes implicit [JSH08]. Also the choice of symmetric basis functions is not helpful for problems which prefer a certain direction, e.g. wave problems [JSH08].

3.4. Discontinuous Galerkin methods

3.4.1. General description

As the previous sections show, both finite volume methods and finite element methods have some strengths, but also some weaknesses. The discontinuous Galerkin method is a method that tries to combine these strengths and eliminate the weaknesses as good as possible.
It uses the same elements as a finite elements method, but the inner nodes are duplicated [JSH08]. The cells/elements are then (in the one-dimensional case) limited to the left by u_i and to the right by u_{i+1}. For example the first cell stretches from u_0 to u_1, while the second cell goes from u_1 to u_2. This shows why the inner nodes have to be duplicated: The value at the right end of cell 1 doesn't have to be the same value as the left end of cell 2. This means the solution can be discontinuos at the cell interfaces. Inside an element the solution is approximated as

$$u_h^k(x) = \sum_{i=0}^{1} u^{k+i} \ell_i^k(x) \tag{3.17}$$

as well as the flux, as explained in [JSH08].
To determine the test functions, the local residual (for a scalar conservation law)

$$R_h(x,t) = \frac{\delta u_h^k}{\delta t} + \frac{\delta f_h^k}{\delta x} \tag{3.18}$$

has to be orthogonal to the test functions, this means the test functions have to fulfill the following equation:

$$\int_{D^k} R_h(x,t) \ell_j^k(x) dx = 0 \tag{3.19}$$

Applying Gauss' theorem and introducing a numerical flux f^* leads to the equation

$$\int_{D^k} R_h(x,t) \ell_j^k(x) dx = [(f_h^k - f^*) \ell_j^k]_{x^k}^{x^{k+1}} \tag{3.20}$$

Like with finite volumes the choice of the numerical flux is an important part of the method. It is also possible to convert this equation so that it mimics the matrix formulation of finite element methods. The mass and stiffness matrices are computed analogous to this.

3.4.2. Nodal and modal formulation

There a two ways to express the solution when using a discontinuous Galerkin method. These two ways are called the nodal and the modal form.

Nodal form The nodal form uses the values at the grid points and interpolates the function between them by using some form of interpolation polynomial [JSH08]:

$$u_h^k = \sum_{i=1}^{N_p} u_h^n(x_i^k, t)\ell_i^k(x) \tag{3.21}$$

In this case the interpolation polynomials are the Lagrange polynomials of order $N_p + 1$.

"The global solution u(x, t) is then assumed to be approximated by the piecewise N-th order polynomial approximation $u_h(x, t)$,

$$u(x, t) \simeq u_h(x, t) = \oplus_{k=1}^{K} u_h^k(x, t) \tag{3.22}$$

defined as the direct sum of the K local polynomial solutions $u_h^k(x, t)$."[JSH08, p.21] Since the residual (here for the constant coefficient advection equation)

$$R_h = \frac{\delta u_h}{\delta t} + \frac{\delta a u_h}{\delta x} \tag{3.23}$$

has to be orthogonal to all test functions

$$\phi_h^k(x) = \sum_{n=1}^{N_p} \Phi_n^k \psi_n(x) \tag{3.24}$$

the following statement has to be fulfilled for all n $\in [0, N_p]$:

$$\int_{D^k} R_h(x, t)\psi_n(x)dx = 0 \tag{3.25}$$

However since this is only a local statement, it doesn't help in finding the global solution. Spacial integration and the choice of a sensible numerical flux $(au_h)^*$ (see [JSH08, p.22] leads to the weak form

$$\int_{D^k} (\frac{\delta u_h^k}{\delta t}\psi_n - a u_h^k \frac{\delta \psi_n}{\delta x}))dx = -\int_{\delta D^k} \hat{n} \cdot (au_h)^* \psi_n dx \tag{3.26}$$

and the strong form

$$\int_{D^k} R_h(x, t)\psi_n(x)dx = \int_{\delta D^k} \hat{n} \cdot (au_h^k - (au_h)^*)\psi_n dx \tag{3.27}$$

where n̂ is the outward pointing normal (+1 and -1 for one dimension)[JSH08] and $(au_h)^*$ is a chosen numerical flux function.
For a mass matrix

$$\hat{M}_{ij}^k = \int_{D^k} \ell_i \ell_j dx \tag{3.28}$$

and stiffness matrix

$$\hat{S}_{ij}^k = \int_{D^k} \ell_i \frac{d\ell_j}{dx} dx \tag{3.29}$$

the weak form can be written as

$$\hat{M}^k \frac{d}{dt} u_h^k - (\hat{S})^T a u_h^k = -(a u_h)^* \psi(x_r^k) + (a u_h)^* \psi(x_l^k) \tag{3.30}$$

and the strong form as

$$\hat{M}^k \frac{d}{dt} u_h^k + \hat{S}^k a u_h^k = (a u_h^k - (a u_h)^*) \psi(x_r^k) - (a u_h^k - (a u_h)^*) \psi(x_l^k) \tag{3.31}$$

[JSH08]

Modal form The modal form uses a local polynomial basis [JSH08]:

$$u_h^k(x,t) = \sum_{n=1}^{N_p} \hat{u}_n^k(t) \psi_n(x) \tag{3.32}$$

where $\psi_n(x)$ is the polynomial basis function and N_p is the polynomial order plus one. Following the same approach as above to determine the weak and strong forms for test functions that have the same space as the solution space (called a Galerkin approach) [JSH08, p.23] leads to

$$\hat{M}^k \frac{d}{dt} \hat{u}_h^k - (\hat{S})^T a \hat{u}_h^k = -(a u_h)^* \psi(x_r^k) + (a u_h)^* \psi(x_l^k) \tag{3.33}$$

with local mass matrix

$$\hat{M}_{ij}^k = \int_{D^k} \psi_i \psi_j dx \tag{3.34}$$

and local stiffness matrix

$$\hat{S}_{ij}^k = \int_{D^k} \psi_i \frac{d\psi_j}{dx} dx \tag{3.35}$$

The strong form is given by

$$\hat{M}^k \frac{d}{dt} \hat{u}_h^k + \hat{S}^k a \hat{u}_h^k = (a u_h^k - (a u_h)^*) \psi(x_r^k) - (a u_h^k - (a u_h)^*) \psi(x_l^k) \tag{3.36}$$

3.4.3. Handling more complex equations

The explanations until now only used a very basic linear flux function au and only one unknown. If we assume that now the flux is given by some flux function $f(u)$, we can use the same equations ((3.27) and (3.26)) for the weak and strong formulation, only that now au_h^k gets replaced by $f_h^k(u_h^k)$ (which is also defined as a sum of coefficents and basis functions) and $(au_h)^*$ by f^*. So they become

$$\int_{D^k}(\frac{\delta u_h^k}{\delta t}\psi_n - f_h^k(u_h^k\frac{\delta \psi_n}{\delta x}))dx = -\int_{\delta D^k}\hat{n}\cdot f^*\psi_n dx \tag{3.37}$$

and

$$\int_{D^k}(\frac{\delta u_h^k}{\delta t} + \frac{\delta f_h^k(u_h^k)}{\delta x})\psi_h^k dx = \int_{\delta D^k}\hat{n}\cdot(f_h^k(u_h^k) - f^*)\psi_h^k dx \tag{3.38}$$

[JSH08, p.31] The generalization into more than one dimension leaves this structure intact, the scalars for u, $\hat{n}$ and f get replaced by their vector equivalents and the space derivative gets replaced by the gradient (weak form) or the divergence (strong form).

3.4.4. Advantages and disadvantages

Advantages The discontinuous Galerkin method makes it possible to achieve higher order accuracy even on unstructured grids. Also it can make use of different flux functions, making it possible to choose a flux tailored to the specific problem that shall be solved [JSH08]. Choosing the right flux makes it also possible to reduce oscillations in comparison with finite element methods [JSH08].

Disadvantages The computational work is a lot higher than in the simpler methods. This is caused by the doubling of the nodes but also by the more complex numerical computation. Especially with high-order interpolation and thus bigger matrices to compute this is an issue. Also, of course, for problems that don't need the advantages of the discontinuous Galerkin method and are not hit especially hard by the problems of other methods, this computational effort is often not worth it.

3.4.5. Comparison between the methods

The following figure gives an overview over the strengths and weaknesses of each of the methods we analyzed so far. It can be seen that the discontinuous Galerkin method is a good combination of the strengths of finite volume methods and finite element methods.

Figure 3.1.: Comparison of the advantages and disadvantes of different methods to solve hyperbolic PDEs, source [JSH08]

	Complex geometries	High-order accuracy and *hp*-adaptivity	Explicit semi-discrete form	Conservation laws	Elliptic problems
FDM	×	✓	✓	✓	✓
FVM	✓	×	✓	✓	(✓)
FEM	✓	✓	×	(✓)	✓
DG-FEM	✓	✓	✓	✓	(✓)

3.5. Overview over the integration methods

The following section aims to give a short overview over the most common time integration methods and their usage without going to much into detail.

3.5.1. Explicit Euler

For a given $\frac{d}{dt}x = f(t,x)$ with initial value $x(t_0) = x_0$ the explicit Euler step is

$$x_{n+1} = x_n + hf(t_n, x_n) \tag{3.39}$$

where h is the time step size.
While this method is simple to use, it doesn't suit all kinds of numerical problems. Especially problems with high stiffness prefer an implicit method. Also more accurate schemes exist. When using an explicit euler step the time step size is also limited by the CFL-condition (has to be lower than 1).

Improved explicit Euler - midpoint method A simple way to improve explicit Euler methods is by first computing a middle step for x

$$x_{n+1/2} = x_n + \frac{h}{2}f(t_n, x_n) \tag{3.40}$$

and then using this to compute the corresponding value of $f(x,t)$ and use it to get the final value of the full time step

$$x_{n+1} = x_n + hf(t_{n+1/2}, x_{n+1/2}) \tag{3.41}$$

3.5.2. Implicit Euler

Used especially for problems of high stiffness, the implicit euler step is defined as

$$x_{n+1} = x_n + hf(x_{n+1}, t_{n+1}) \tag{3.42}$$

The new computation of f for all time steps needs more computational effort, but since the implicit euler method is not restricted in its time step size by the CFL condition, this may be worth it for some problems.

3.5.3. Heun's method

To get a higher rate of convergence (the euler methods have 1, Heun's method has 2) a temporary value $\tilde{x}_{k+1}$ is computated. The complete method is then given by

$$\begin{aligned} \tilde{x}_{n+1} &= x_n + hf(x_n, t_n) \\ x_{n+1} &= x_n + \frac{1}{2}h(f(t_n, x_n) + f(t_{n+1}, \tilde{x}_{n+1})) \end{aligned}$$

The disadvantage of this method is that for each time step two values have to be computed.

3.5.4. Runge-Kutta method

Runge-Kutta methods are a family of methods with varying number of middle values per timestep and explicit as well as implicit forms. The most common Runge-Kutta method has 4 middle values and is explicit. It has the form

$$x_{k+1} = x_k + h \cdot \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.43}$$

with

$$\begin{aligned} k_1 &= f(x_n, t_n) \\ k_2 &= f(x_n + \frac{h}{2}k_1, t_n + \frac{h}{2}) \\ k_3 &= f(x_n + \frac{h}{2}k_2, t_n + \frac{h}{2}) \\ k_4 &= f(x_n + hk_3, t_n + h) \end{aligned}$$

With this method a rate of convergence of 4 is possible, but at the cost of having to compute 4 extra values per time step.

3.6. Analytic solution of the dam-break problem

To evaluate how well the numerical methods used for solving the dam-break problem in the implementation work and how they compare to each other, we need to know the exact solution for this problem. The following section will explain how to solve a given dam-break problem using the shallow water equations. The specific solution for the problem used in the implementation will be computed based on this knowledge in the next chapter.

3.6.1. Introduction

As defined earlier, a dam-break problem is a special set of initial conditions, where the initial velocities are zero over the whole domain and the height is piece-wise constant, with a left and a right starting height, as in (2.2.1).
The solution can be divided into several areas with identical behavior. Two of those sections are the areas where the water has still the height before the waves reach it (so the smaller of h_l and h_r and another behind the waves where the height is still the bigger one. What happens between those boundaries depends on the properties of the problem, more precisely on which types of of waves occur in which order. For the dam-break problem as defined above, the solution always contains one shock wave and one rarefaction wave [Lev04, p.260].

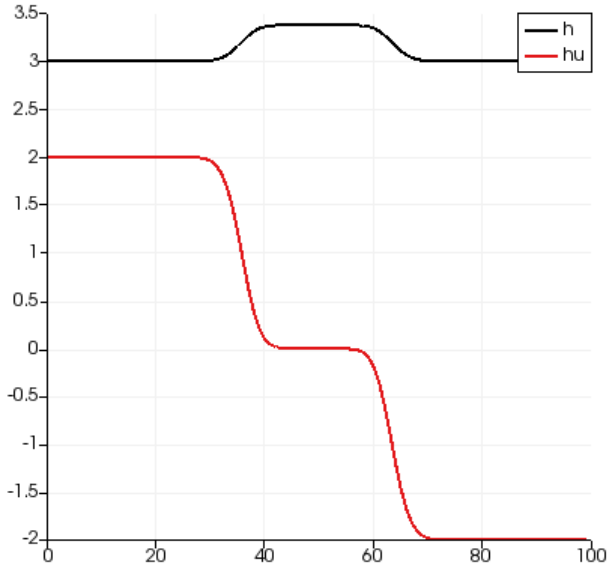
Shock waves Physically a shock wave is a wave traveling faster than the speed of sound of the fluid it travels in. It is characterized by a sudden change of value of the examined variables. A well known example is the sonic boom observed when aircraft travel faster than the speed of sound in air. A shock wave also occurs at the end of a traffic jam, where cars' velocity changes abruptly from some positive value to zero. In the case of the shallow water equations, shock waves are discontinuities in height and/or velocity that travel through the water.

Rarefaction waves Rarefaction waves are defined by a gradual increase or decline of (in our case) height and/or velocity, in contrast to the sudden changes associated with shock waves. Classically, they occur behind a traveling shock wave. They usually widen and spread out over time.

3.6.2. Different types of wave composition

Two shocks If the initial height is uniform over the whole domain and the velocity is piece-wise constant with $u_r = -u_l$, the solution consists of two shock waves. The

Figure 3.2.: Two shocks - Paraview visualization of a numerical approximation to a problem with $u_l = 2$, $u_r = -2$ and $h_0 = 3$ at time $t \approx 2.65$ for a computational domain of size 100



approximated shock waves can be seen, especially in hu. The left shock travels to the left and the right shock to the right.

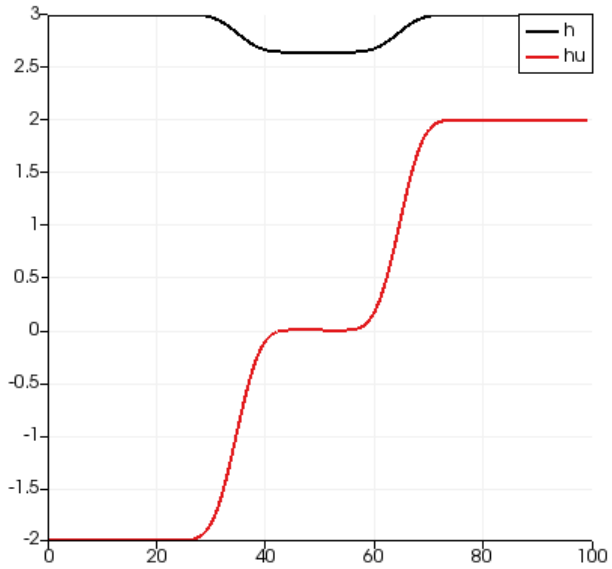
Two rarefaction waves While the example for two shock waves can be explained as two bodies of water colliding, two bodies of water that travel away from each other lead to a solution consisting of two rarefaction waves. To achieve this, the sign of the velocities used in 3.2 have to be reversed. Since the graphics were made from a numerical approximation using a finite volume scheme the difference between the shocks and rarefaction waves is not as strong as in an exact solution.

One rarefaction wave and one shock The problem that is used in the implementation section is solved by a combination of a shock and a rarefaction wave. The shock travels to the right and the abrupt change in height is clearly visible.

3.6.3. Solving the problem

In all types of wave composition the solution can be split in five sections. The first is, as already discussed earlier, the area behind the wave. In 3.4 this is the left section with

Figure 3.3.: Two rarefaction waves - Paraview visualization of a numerical approximation to a problem with $u_l = -2$, $u_r = 2$ and $h_0 = 3$ at time $t \approx 2.65$ for a computational domain of size 100

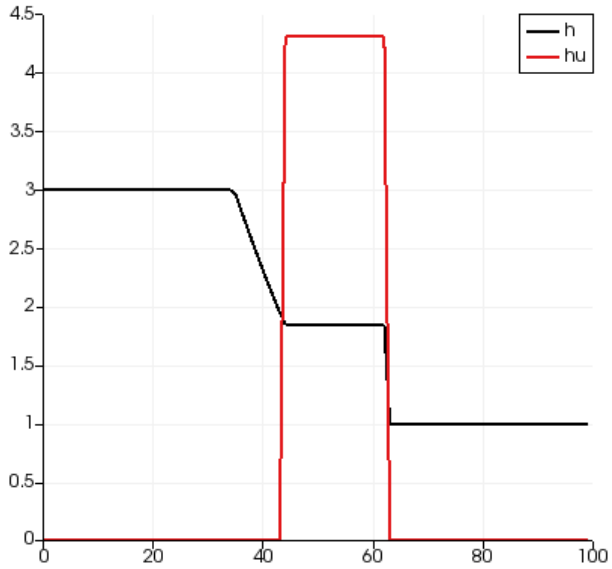


$h = h_l$. There is also an area in front of the wave, in 3.4 this is the section with $h = h_r$. Between those two areas the two waves impact the variables. Each of them defines the behavior in one section. Between the waves there is some middle state with properties that depend on the initial problem.

To find an exact solution, the first step is to find out what types of waves occur in the given scenario. To do this, an entropy condition like the Lax condition, which gives requirements for a shock wave, can be used [Lev04, p.285]. Also a way to define which x value corresponds to which part of the solution is needed. The third step is then to compute the middle state q_m between the two waves.

Determining the boundaries of each section Since one of the main characteristics of a Riemann problem is that its solution is constant along rays of x/t, this can be used to determine the different parts of the solution. Since one of the areas not affected by the waves might be the one behind a rarefaction wave, this means that for these x values the wave speed of the neighbouring rarefaction is higher than x/t. The same consideration can be used for the areas in front of shocks. For those areas x/t has to be faster than than the shock speed. If the solution consists of two shocks, these

Figure 3.4.: 1 rarefaction, 1 shock - Paraview visualization of the exact solution to a problem with $u_l = u_r = 0$, $h_l = 3$ and $h_r = 1$ at time $t \approx 2.65$ for a computational domain of size 100



boundaries are sufficient to solve the problem, since the shock waves are discontinuities and thus occur at exactly one x. If the solution has rarefaction waves, if x/t is between the outer wave speed and the middle state wave speed, it is in the area of the rarefaction wave. If x/t is faster than the speed of the left wave, but slower than the right wave (or the other way around for solutions traveling to the right) it lies within the section of the middle state.

Getting the middle state The height and velocity of the middle state q_m depends on the heights and velocities of the left and right state. If a shock is present, the system of equations arising from the Rankine-Hugoniot condition can be used to compute the middle state [Lev04, p.265]. For a solution consisting of two shocks, this is sufficient and gives the physically correct solution.

Computing the shock speed The shock speed is given by the Rankine-Hugoniot condition [Lev04, p.265] and depends on the values of the two states in front and behind the shock. This is described by the equation

$$s = \frac{h_* u_* - hu}{h_* - h} \tag{3.44}$$

and if assume, for example, that the shock is between the middle and the right state $h_* = h_m$ and $h = h_r$.

Determining the behaviour inside a rarefaction wave The behaviour of a rarefaction wave can be described by a centered wave. These are a type of simple waves, which means they can be described by scalar nonlinear equations [Lev04, p.269]. They are of the following form:

$$q(x,t) = \begin{cases} q_l & x/t \leq \xi_1 \\ \tilde{q}(x,t) & \xi_1 \leq x/t \leq \xi_2 \\ q_r & x/t \geq \xi_2 \end{cases} \tag{3.45}$$

q_l and q_r are some states separated by a rarefaction wave, where q_l has a lower wavespeed than q_r [Lev04, p.276]. The exact process to compute $\tilde{h}$ is given in [Lev04, p.277]. A detailed explanation is also given in [Sto57], chapter 10, especially 10.8. It uses the fact that q_m has to lie on specific integral curves. It results in

$$\tilde{h} = \frac{1}{9g}(A - \xi)^2 \tag{3.46}$$

with

$$A = u_l + 2\sqrt{gh_l} = u_r + 2\sqrt{gh_r} \tag{3.47}$$

This means the evolution of h inside the rarefaction wave is described by a parabolic curve.

To compute u the value for $\tilde{h}$ has to be put in the following equation

$$u = u_* - 2(\sqrt{gh_*} - \sqrt{gh}) \tag{3.48}$$

[Lev04, p.271]

Putting the solution together Now all values and all boundaries of areas are known. The final solution is then given by defining a piecewise defined function that takes all the information computed before into consideration. [al.16] gives a compact overview over all kinds of analytic solutions for the shallow water equations in one and two dimensions. For a problem with initial conditions similar to 3.4 they give the solution, consistent with the solving process described above, as

$$h(x,t) = \begin{cases} h_l & x/t \leq \lambda_{1l} \\ \frac{4}{9g}(\sqrt{gh_l} - \frac{1}{2}(x/t))^2 & \lambda_{1l} \leq x/t \leq \lambda_{1m} \\ h_m & \lambda_{1m} \leq x/t \leq s \\ h_r & x/t > s \end{cases} \tag{3.49}$$

$$u(x,t) = \begin{cases} u_l & x/t \leq \lambda_{1l} \\ \frac{2}{3}((x/t) + \sqrt{gh_l}) & \lambda_{1l} \leq x/t \leq \lambda_{1m} \\ u_m & \lambda_{1m} \leq x/t \leq s \\ u_r & x/t > s \end{cases} \tag{3.50}$$

The equations given in [Sto57, chapter 10.8] also lead to the same piece-wise defined function. For a solution consisting only of shocks or only of rarefaction waves, the values and boundaries would be different, according to the above explanations.

4. Implementation

4.1. Computed scenarios

Two types of initial conditions were used, one of them continuous and one discontinuous. The same initial conditions were used for the finite volumes solver as well as for the discontinuous Galerkin solver to be able to compare the results.

4.1.1. Continuous initial conditions - Gaussian distribution

As an example of continuous initial conditions a water hump with gaussian height distribution was used. The water height (for the advection equation it might also be some density or similar) is described by a gaussian distribution

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.1}$$

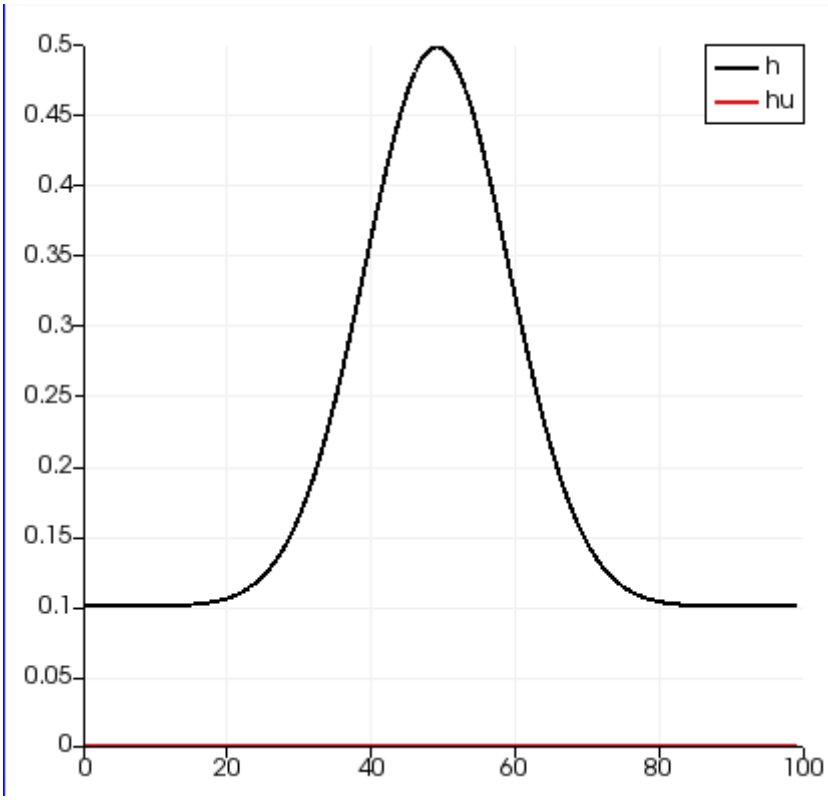
with $\sigma = 10$, $\mu = 0$ and $x = pos - x_0$, where x_0 is the middle of the computational domain. So the height distribution is given by

$$h(x) = \frac{1}{\sqrt{2\pi 100}} e^{-\frac{x^2}{200}} \tag{4.2}$$

This height is then multiplied by 10 and added to a basic height of 0.1 to prevent the water height becoming zero and disturbing the calculations by using negative values, which might lead to uncontrolled oscillations. The water hump then has its maximum at the middle of the domain with a corresponding height of ≈ 0.5.

Boundary conditions For the advection equation periodic boundary conditions were used. Since the wave reaches its original position again, it is possible to use this fact to compute the error over time (for the time where the maximum reaches the middle for the n-th time) by simply comparing it with the starting conditions.
For the shallow water equations the boundary elements were simply estimated by their direct neighbour and thus the boundary conditions were not periodic.

Figure 4.1.: Paraview visualization of the initial conditions following a Gaussian distribution for a computational domain of size 100



4.1.2. Discontinuous initial conditions - dam-break problem

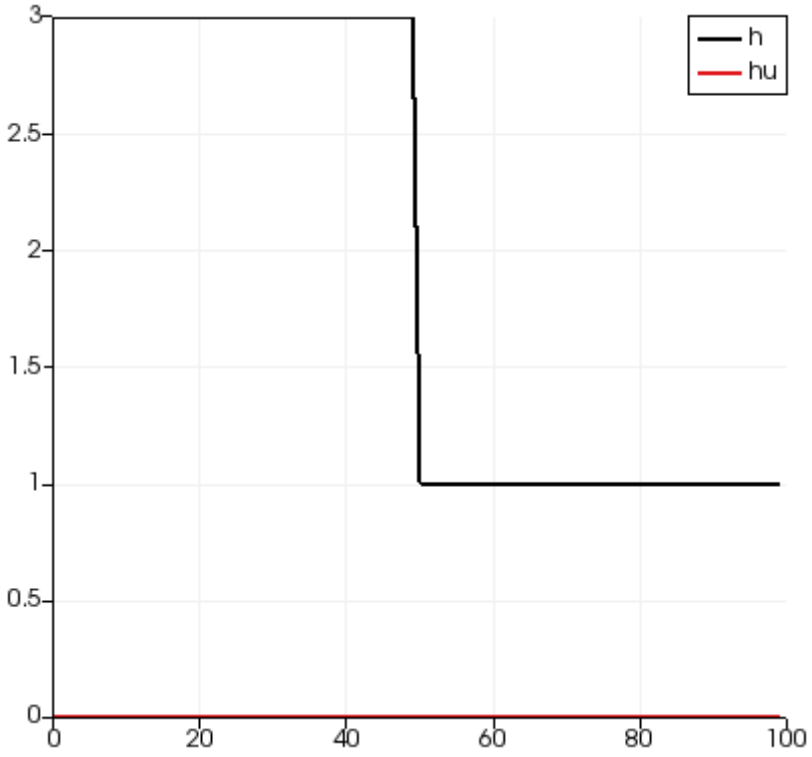
To test the numerical solution's ability to handle discontinuities a discontinuos test case was used. This was a classical dam-break problem with initial values

$$h(x,0) = \begin{cases} 3 & x \leq x_0 \\ 1 & x_0 \leq x \end{cases} \tag{4.3}$$

$$hu(x,0) = 0 \tag{4.4}$$

with x_0 defining the middle of the computational domain.
This defines a non-moving area of water, where the left half has height 3 and the left half has height 1.
For this dam-break problem an analytic solution can be found, as given in the next section. This can then be used to compute an error to test the accuracy of the used numerical methods.

Figure 4.2.: Paraview visualization of the initial conditions of the used dam-break problem for a computational domain of size 100



Boundary conditions Like for the shallow water solution for the Gaussian bump, non-periodic boundary conditions were used that simply extrapolate from their direct neighbour.

4.1.3. Exact solution of the used dam-break problem

The exact solution for the used dambreak problem 4.3 can be computed by using the equations (3.49) and (3.50) and using the given values for h and u. The necessary wavespeeds are

$$\lambda_{1l} = u_l - \sqrt{gh_l} = -5.423991150$$
$$\lambda_{1m} = u_m - \sqrt{gh_m} = -1.92517691$$
$$s = (hu_m - hu_r)/(h_m - h_r) = 5.081313902$$

The middle state has the following properties:

$$h_m = 1.848576603$$
$$u_m = 2.332542824$$

All equations were solved with and the values computed with maple.
This leads to the following analytic solution:

$$h(x,t) = \begin{cases} 3 & x/t \leq -5.423991150 \\ \frac{4}{9g}(\sqrt{3g} - (x/2t))^2 & -5.423991150 \leq x/t \leq -1.92517691 \\ 1.848576603 & -1.92517691 \leq x/t \leq 5.081313902 \\ 1 & x/t > 5.081313902 \end{cases} \tag{4.5}$$

$$u(x,t) = \begin{cases} 0 & x/t \leq -5.423991150 \\ \frac{2}{3}((x/t) + \sqrt{3g}) & -5.423991150 \leq x/t \leq -1.92517691 \\ 2.332542824 & -1.92517691 \leq x/t \leq 5.081313902 \\ 0 & x/t > 5.081313902 \end{cases} \tag{4.6}$$

4.2. Details of implementation

4.2.1. General information

The program is based on the SWE1D program by Sebastian Rettenberger [Ret]. While none of the original solvers are still used, the basic of the main program remains unchanged and the basic vtk file writer is still used (also as a base for more complex vtk and vtu writers). All implementation was done in C++. The program can be built either with CMake or with scons. The output from the numerical computations can be visualized using ParaView, using either the "Plot Data" filter (for all finite volume solvers) or "Plot Over Line" for discontinuous Galerkin solvers.
While it can be run using command line parameters for number of intervals and numer of time steps (e.g. -s 1000 -t 300 for 1000 space intervals and 300 time steps), it is also possible to manipulate these parameters inside args.h. The very beginning of main.cpp contains some boolean variables that function as switches for the different solvers and scenarios.

4.2.2. Switches to use finite volume methods

bool SWE This boolean is set to true if some scenario shall be computed using the shallow water equations, using a finite volume method.

bool Adv This boolean is used to set the used equations to the advection equation (simplest form with constant wave speed).

bool dambreak This boolean is set additionally to SWE for a dam break scenario that shall be solved using finite volume methods.

bool gauss This boolean can be used either with SWE or with Adv, depending if the evolution of the Gaussian hump given in (4.1) shall be solved for the shallow water equations or for the advection equation. Used only for finite volume methods.

4.2.3. Switches to use discontinuous Galerkin methods

bool AdvDG This is set to true to solve the advection equation with a discontinuous Galerkin scheme.

bool SWEDG This boolean is used to solve the shallow water equation for the dam break problem given earlier.

bool SWEDGGAuss If this is set to true, the shallow water equations will be solved for a Gaussian water hump.

4.2.4. Other switches

bool errorComputing If this is set to true, the error of the used method will be computed. For the advection equation, the initial condition will be saved and compared to the numerical solution after the maximum of the Gaussian distribution reaches its origin again. For the shallow water equations, the error will only be computed for a dam break scenario. At a predefined time (currently set to 0.5s), the exact solution will be computed using the equations (4.5) and (4.6). This can then be compared to the numerical solution. If error computing is active, no vtk output will be written. The time steps needed to reach the point where an error can be computed will also be computed and overwrite the given number of time steps. This is especially important for a very high number of intervals or very small CFL numbers.

4.3. Used methods

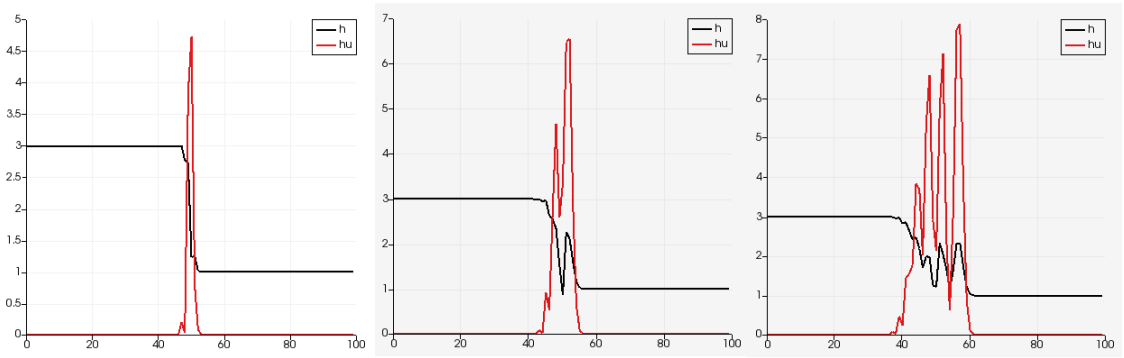
4.3.1. Finite Volumes

All finite volume methods use the same process. First the boundary values are set (either to periodic or neighbour values, depending on the scenario). Then for each cell the numerical fluxes from the left and right neighbour are computed. In this step, the new time step is also computed from the CFL condition. After that, the values of the fluxes are used to compute the new values of q. Afterwards, the results are given to the vtk writer and written to a file.

Numerical fluxes

Unstable flux The method described in (3.8) and (3.9) is implemented using those equations, with Courant number 0.4. 4.3 shows the increasing oscillations over time (soon after this they start to amplify with every time step and completely dominate the results) and thus this numerical flux function is not suitable for hyperbolic PDEs, as already said earlier.

Figure 4.3.: Evolution of the solution to the dam break problem using an unstable flux, t=0.5, t=1.6 and t=3.65



Lax-Friedrichs flux Equations (3.10) and (3.6) are used to implement the classical Lax-Friedrichs flux. While the results, visible in 4.4 and 4.5, still show oscillations, these oscillations are more stable and don't start to dominate the solution completely.

Local Lax-Friedrichs flux The local Lax-Friedrichs flux is implemented using (3.11) for a Courant number of 0.4. This method clearly shows the best results, as shown in

Figure 4.4.: Evolution of the solution of the dam break problem using Lax-Friedrichs flux, t1=1.4, t2=2.7, t3=5.2

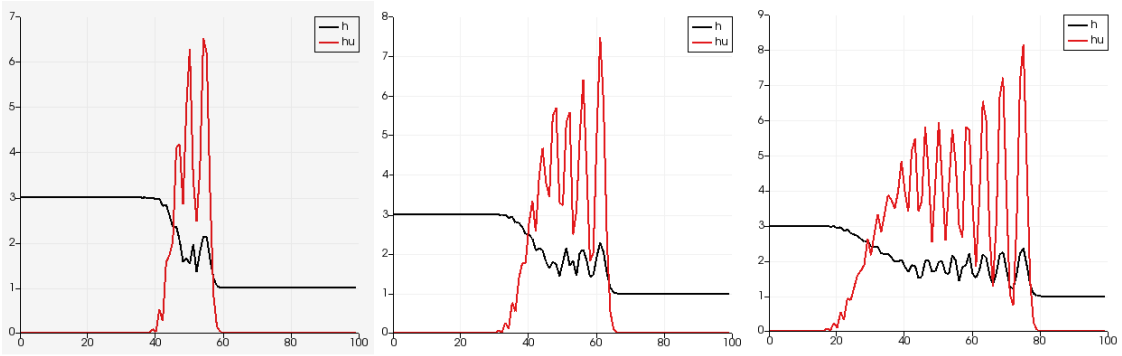
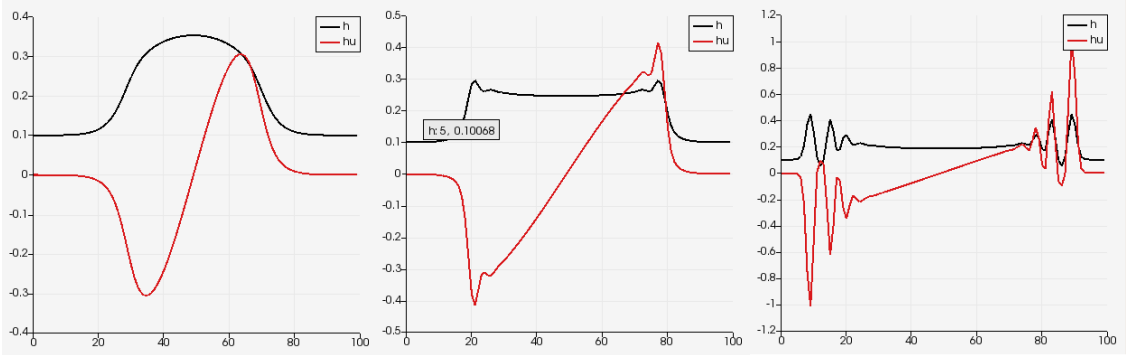


Figure 4.5.: Evolution of the solution of the gaussian water hump using Lax-Friedrichs flux, t1=5.4, t2=10.1, t3=15.3



4.6 and 4.7. The solution doesn't show any kind of unwanted oscillations and is also able to handle the discontinuity in the dam break problem well.

4.3.2. Discontinuous Galerkin

The process for the discontinuous Galerkin methods is similar to the one for finite volume methods. First the boundary conditions are set, then the numerical fluxes and the next time step are computed. Then the time derivative of q is computed from the fluxes. Finally, the new values for q (for each node) are computed using the time derivative by using a time-integration method (in this case an explicit Euler step). After this, the values of q are given to a vtu writer and written to a file.

Figure 4.6.: Evolution of the solution of the dam break problem using a local Lax-Friedrichs flux, t1=1.4, t2=2.7, t3=4.5

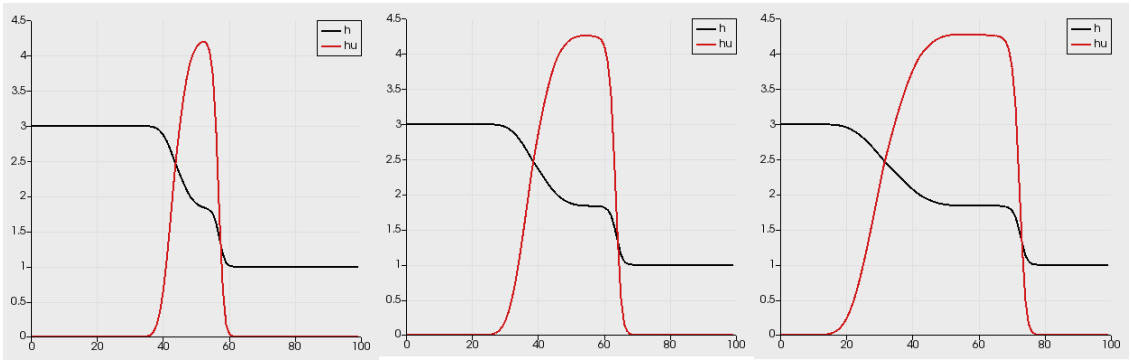
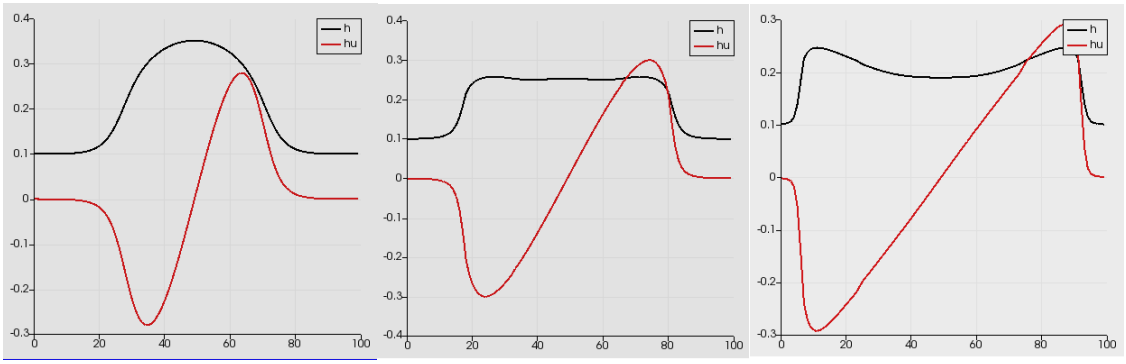


Figure 4.7.: Evolution of the solution of the gaussian water hump using a local Lax-Friedrichs flux, t1=5.4, t2=10.1, t3=15.3



The numerical flux used is a local Lax-Friedrichs flux.

The local test functions, as well as the basis functions, are the Lagrange polynomials of degree 1 over the basis interval from 0 to 1. So $\varphi_0(x) = 1 - x$ and $\varphi_1(x) = x$.

Advection equation For the simplest case of the advection equation (one dimension, fixed speed), the following equation has to be solved:

$$u_t + au_x = 0 \tag{4.7}$$

It is important to remember that u is in fact a vector, containing two nodes for each cell ($u = \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}$). For each interval (the finite elements of the scheme), this means the

following has to be true:

$$\int_{I_i} \varphi_i(x)(u_t + au_x) = 0 \tag{4.8}$$

Sine we use a nodal representation, this leads to the following:

$$\int_{I_i} \varphi_i(\sum_j (\varphi_j u_j)_t + a\sum_j (\varphi_j u_j)_x)dx = 0 \tag{4.9}$$

The goal is to get to a expression for u_t, so the above expression has to be solved for this, using the fact that φ does not depend on t and u_j doesn't depend on x. Partial integration is also used.

$$\begin{aligned}
\sum_j \int_{I_i} \varphi_i((\varphi_j u_j)_t + a(\varphi_j u_j)_x)dx &= 0 \\
\sum_j \int_{I_i} \varphi_i(\varphi_j(u_j)_t + a(\varphi_j)_x u_j)dx &= 0 \\
\sum_j \int_{I_i} \varphi_i\varphi_j(u_j)_t + a\varphi_i(\varphi_j)_x u_j dx &= 0 \\
\sum_j \int_{I_i} \varphi_i\varphi_j(u_j)_t dx + a\int_{I_i} \varphi_i(\varphi_j)_x u_j dx &= 0 \\
\sum_j \int_{I_i} \varphi_i\varphi_j(u_j)_t dx - a\int_{I_i} \varphi_j u_j(\varphi_i)_x dx + \int_{\delta I_i} \varphi_i\varphi_j u_j \vec{n} dx &= 0 \\
\sum_j \int_{I_i} \varphi_i\varphi_j(u_j)_t dx - a\int_{I_i} \varphi_j u_j(\varphi_i)_x dx + (\varphi_i\varphi_j a u_j)|_{x_l} - (\varphi_i\varphi_j a u_j)|_{x_r} &= 0 \\
\sum_j \int_{I_i} \varphi_i\varphi_j(u_j)_t dx - a\int_{I_i} \varphi_j u_j(\varphi_i)_x dx + (\varphi_i\varphi_j)|_{x_l}((au)^*)_l - (\varphi_i\varphi_j)|_{x_r}((au)^*)_r &= 0
\end{aligned}$$

This can be put into the matrix form known from chapter 3.

$$M\Delta x u_t - S\Delta x(au) + N_0 f_l - N_1 f_r = 0 \tag{4.10}$$

This leads to the wanted equation for u_t:

$$u_t = (M^{-1} \cdot (S(au) + N_0 f_l - N_1 f_r))/\Delta x \tag{4.11}$$

f_l and f_r are the fluxes from left or right, computed with a local Lax-Friedrichs approximation. The matrices are (instead of evaluating the specific interval, the transformation

theorem can be used to use the basis interval from 0 to 1 instead):

$$M = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix} M^{-1} = \begin{pmatrix} 4 & -2 \\ -2 & 4 \end{pmatrix}$$

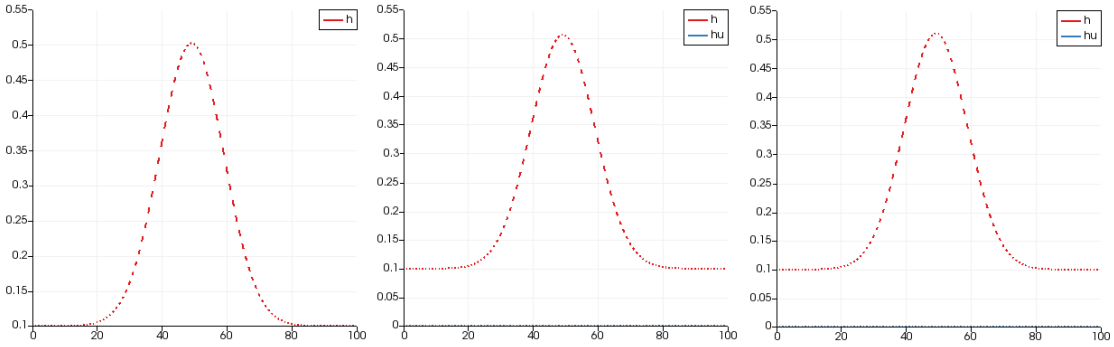
$$S = \begin{pmatrix} -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} N_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} N_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

The explicit Euler step is then given by

$$u^{n+1} = u^n + \Delta t u_t \tag{4.12}$$

Visualization of results 4.8 shows the results for the advection equation after one, two and three turnarounds.

Figure 4.8.: DG approximation (advection equation) after 1, 2 and 3 full "circles"



Shallow water equations Since the shallow water equations are a system of two hyperbolic PDEs, the derivation of u_t, as done above for the advection equation, has to be done twice, once for each equation. While the advection equation used the very simple flux au, now the equations use a flux function $f(q)$, as defined in 2.2.1. This doesn't change the structure of the result, but $S(au)$ becomes $Sf(h)$ for the equation for h and $Sf(hu)$ for the equation for hu. Of course, the left and right fluxes also use the corresponding flux function and not the one from the advection equation.

Visualization of results 4.9 shows the results for the shallow water equations after one, two and three turnarounds for a dam break problem and 4.10 shows the results for a Gaussian water hump.

Figure 4.9.: DG approximation (SWE, dam break) t1=1.4, t2=2.7, t3=4.5

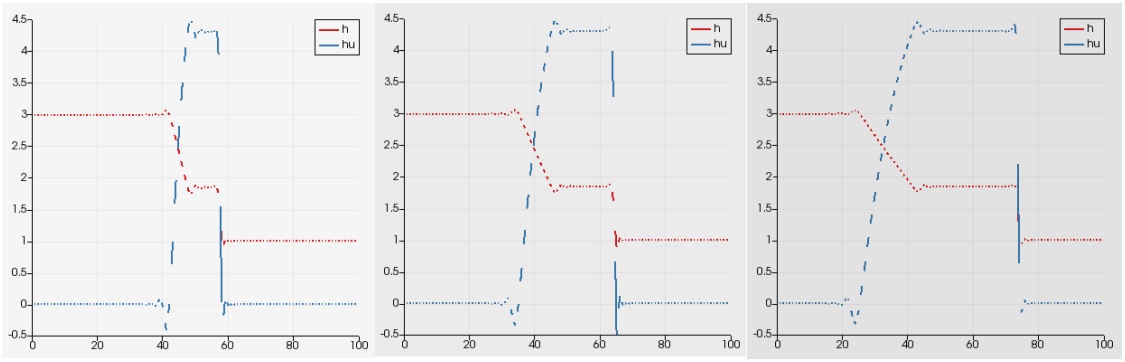
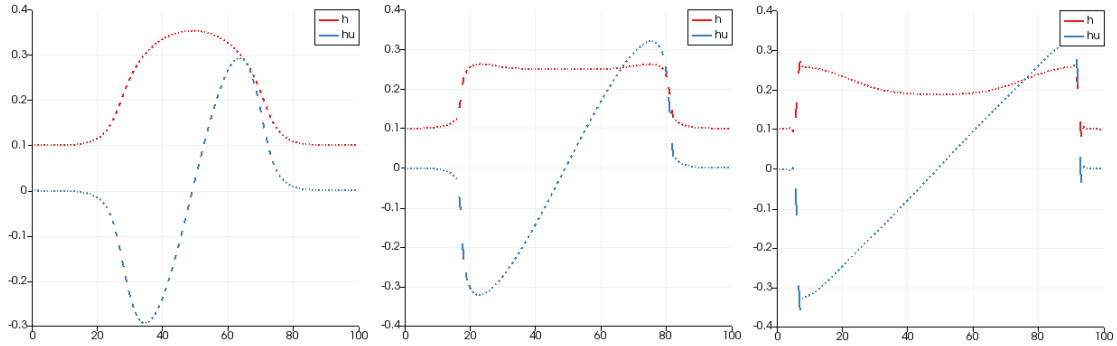


Figure 4.10.: DG approximation (SWE, gaussian) t1=5.4, t2=10.1, t3=15.3



4.4. Comparison of finite volume and discontinuous Galerkin

4.4.1. Advection equation

If the time evolution of the advection solution is computed using a finite volume scheme, it contains too much dissipation. The expected result would be a wave of unchanging shape traveling at the given wave speed, since the model contains no friction losses. However, the wave maximum decreases noticeably over time and the shape also doesn't stay the same (visualized in 4.11).

If the approximation computed by a discontinuous Galerkin solver is evaluated at the same time, these dissipation effects don't occur at all, neither in height nor in shape (see 4.12).

A direct comparison between the two methods (same size of domain and same

Figure 4.11.: Finite volume approximation (advection equation) after 1, 2 and 3 full "circles"

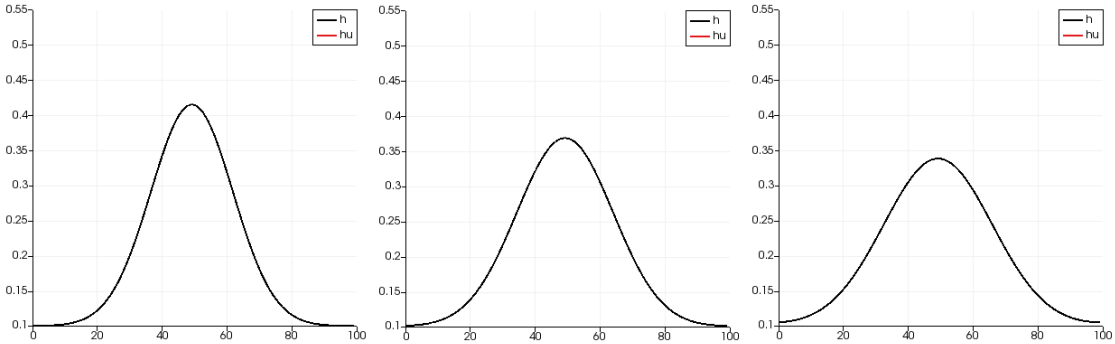
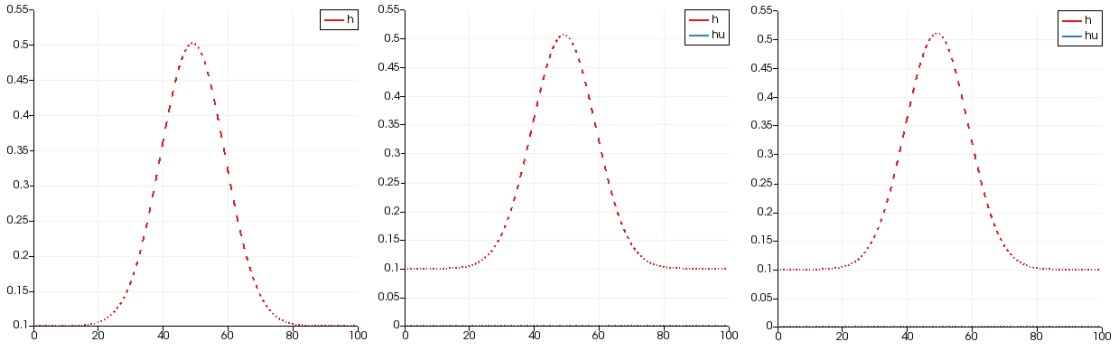


Figure 4.12.: Discontinuous Galerkin approximation (advection equation) after 1, 2 and 3 full "circles"



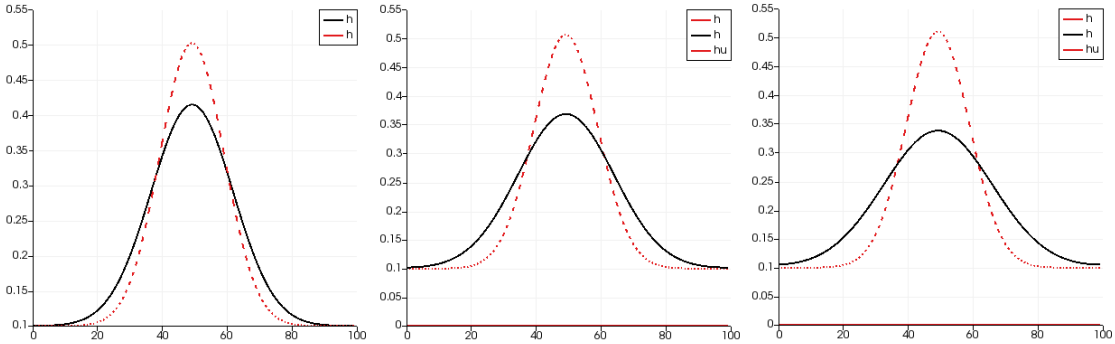
Courant number) is given in 4.13. The benefits for the discontinuous Galerkin method increase with increasing time.

4.4.2. Shallow Water Equations

For the shallow water equations, the same effects occur in principle. Also it is clearly visible that discontinuous Galerkin methods are able to compute results closer to the analytic solution, especially at the location of the shock wave. The comparison is visualized in 4.14.

For Gaussian hump starting conditions, the observed effects are mainly the same. Again, the dissipation in the finite volume method is to high and gets worse over time (see 4.15).

Figure 4.13.: Comparison between finite volume method and discontinuous Galerkin method for the advection equation after 1, 2 and 3 full "circles"



4.5. Error estimation

4.5.1. Advection equation

The error is computed using the p-norm for p=2.

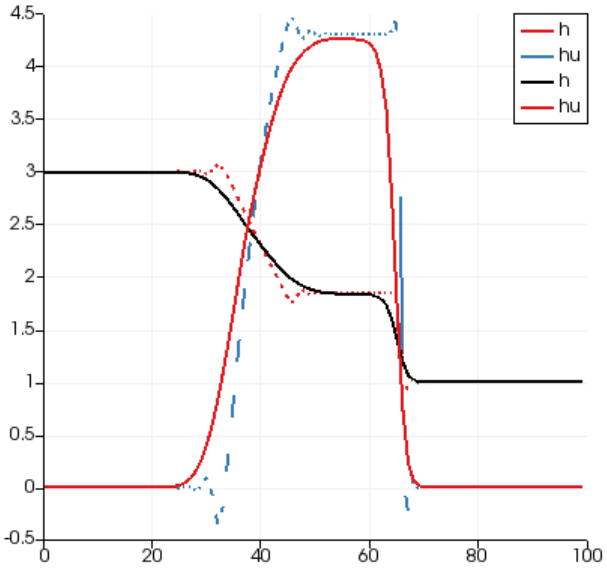
$$error = \sqrt{\Delta x \sum_{i=1}^{I} (u_i^{num} - u_i^{exc})^2} \tag{4.13}$$

where I is the number of intervals in the domain. The domain size is 100.
For finite volumes the expected result is linear convergence, at least until accuracy is limited by machine accuracy. Diagram 4.16 (remark: excel plot will be replaced by better plot soon) plots the error against the number of intervals. Until around 1200 intervals, linear convergence can be observed. After that, the error converges against machine accuracy.

For the shallow water equations convergence is limited by the choice of integration method. While the absolute value of the error (computed with the same method as for finite volumes) is smaller by a factor of more than 10 than it's finite volumes counterpart, the rate of convergence is still linear. This is caused by using an explicit Euler step, which has a rate of convergence of one, so the rate of convergence of the overall method can not be better than this. The results are visualized in 4.17.

If instead of an explicit Euler step the midpoint method is used, the expected result is quadratic convergence. The visualization in 4.18 shows that the error for the used implementation shows this behaviour. Using the midpoint method slows the computation

Figure 4.14.: Comparison between finite volume method and discontinuous Galerkin method for the shallow water equations at t=3



of results and error noticeably. This is caused by needing twice as much computations per time step.

4.5.2. Shallow water equations

For the shallow water equations the expected results are the same as for the advection equation. Error computing is now done by comparing the numerical approximation with the analytic solution at a predetermined time $t = 0.5$. The results are visualized in 4.19, 4.20 and 4.21. For the finite volume method, the error of h behaves as expected, however, the error of hu doesn't show linear convergence, but decreases too slow and finally stagnates if number of intervals is higher or equal than 800. This occurs also for the discontinuous Galerkin solver. If only h is evaluated, both the finite volume solver and the discontinuous Galerkin solver show linear convergence.

Figure 4.15.: Comparison between finite volume method and discontinuous Galerkin method for SWE, gaussian hump, at t=12.5

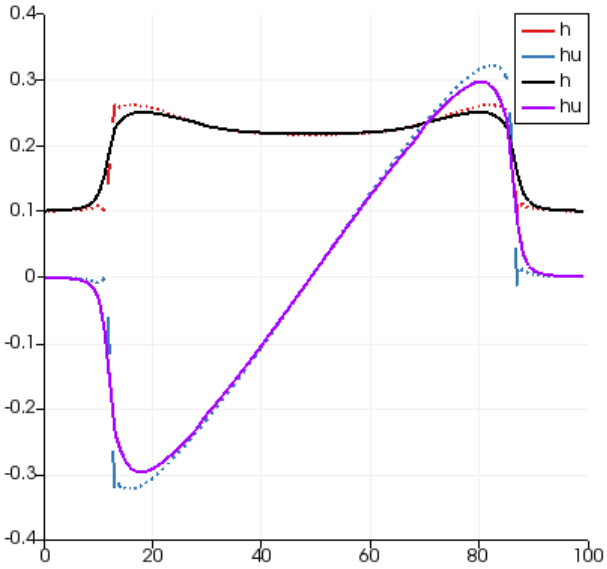


Figure 4.16.: Diagram plotting error (p=2 norm) against number of intervals for a finite volume solver, CFL=0.04, advection equation. Logarithmic scale

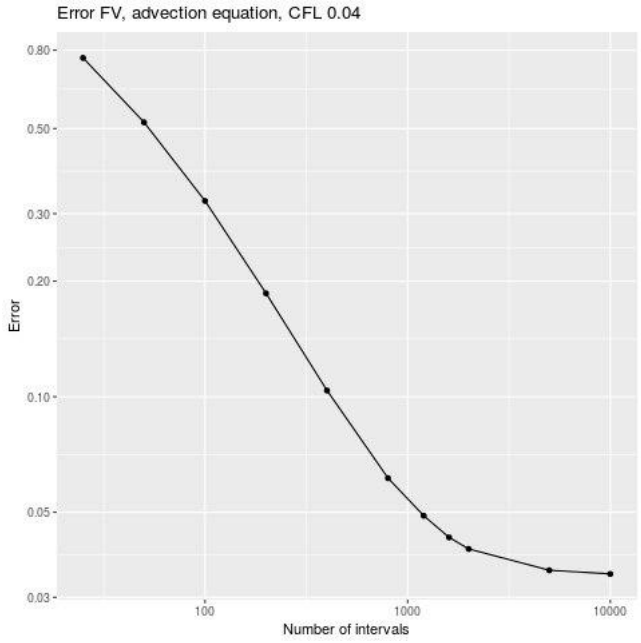


Figure 4.17.: Diagram plotting error (p-norm, p=2) against number of intervals for a discontinuous Galerkin solver, CFL=0.04, advection equation. Logarithmic scale

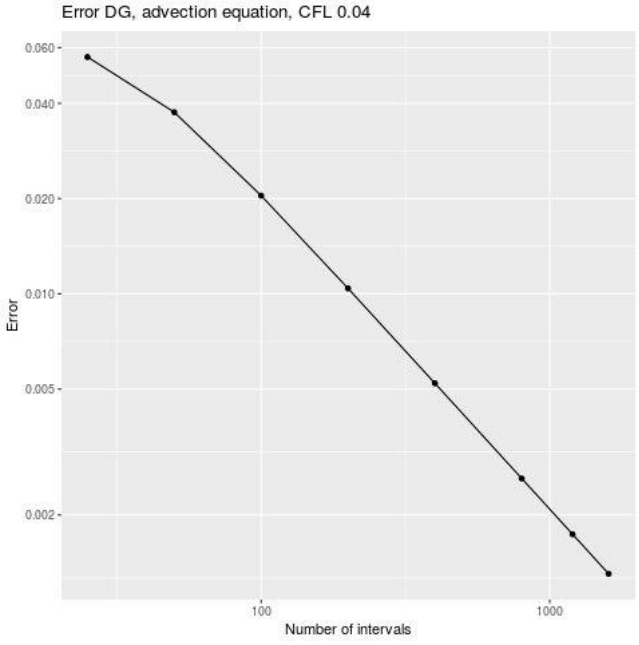


Figure 4.18.: Diagram plotting error (p-norm, p=2) against number of intervals for a discontinuous Galerkin solver, midpoint method, CFL=0.04, advection equation. Logarithmic scale

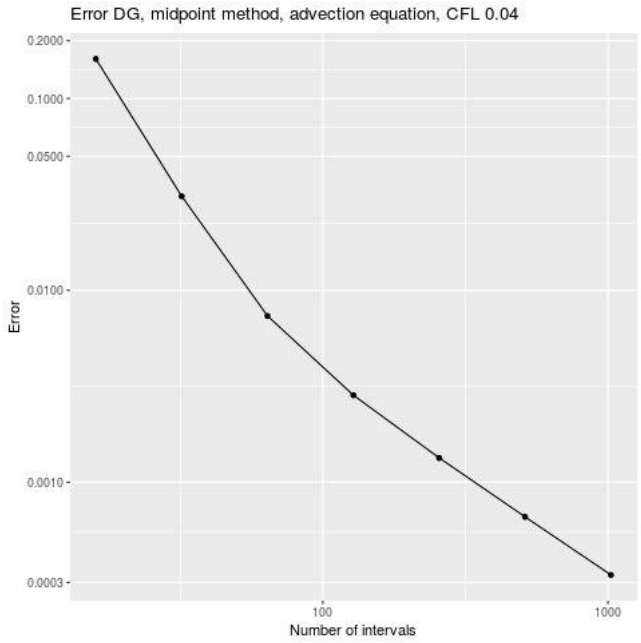


Figure 4.19.: Diagram plotting error (p-norm, p=2) against number of intervals for a finite volume solver, CFL=0.04, shallow water equations. Logarithmic scale

Figure 4.20.: Diagram plotting error (p-norm, p=2) against number of intervals for a discontinuous Galerkin solver, CFL=0.04, shallow water equations. Logarithmic scale

Figure 4.21.: Diagram plotting error (p-norm, p=2) against number of intervals for a discontinuous Galerkin solver, midpoint method, CFL=0.04, advection equation. Logarithmic scale

5. Conclusion

While the tested finite volume approach solves problem with continuous initial conditions reasonably well, at least when a local Lax-Friedrichs flux is used and the simulation doesn't run for too long, and make the usage of discontinuous Galerkin method unnecessary for those easier cases, the results differ vastly for a dam break problem. The comparison clearly shows that a discontinuous Galerkin solver can handle discontinuities a lot better and also doesn't have the problem of too much dissipation. Especially the numerical solution of the advection equation shows too much dissipation when solved with finite volumes. So even if a problem is continuous and otherwise very simple, using a discontinuous Galerkin solver can be beneficial when later times shall be evaluated. Since discontinuous Galerkin methods were designed especially for solving hyperbolic PDEs and conservation laws, they are the preferable choice of numerical method for most of these equation systems. These methods also have the advantage that if two different systems of PDEs shall be solved and the used basis and test functions stay the same, then all used matrices (stiffness, mass and the ones applied to right and left fluxes) stay the same and can be reused without computing them again. This means that those pre-computed matrices can be used as a framework for solving several different equation systems.

The theory and methods presented in this thesis can be adapted to more-dimensional problems or other systems of hyperbolic PDEs. Analyzing the error and its convergence made it clear that for a higher order of accuracy for a discontinuous Galerkin method it would be necessary to use higher order integration methods such as standard Runge-Kutta.

A. Appendix

A.1. Overview over computed errors

A.1.1. Advection equation - finite volumes

No. of intervals	25	50	100	200	400	800
Error	0,764036	0,519036	0,323876	0,186017	0,103836	0,061355
No. of intervals	1200	1600	2000	5000	10000	
Error	0,0489753	0,0430092	0,0401233	0,035304	0,0345383	

Table A.1.: Computed errors for Courant number = 0.04, advection equation, finite volumes method

A.1.2. Advection equation - discontinuous Galerkin

No. of intervals	25	50	100	200	400	800
Error	0,056056	0,0374915	0,0204375	0,0104037	0,00521661	0,00260811
No. of intervals	1200	1600				
Error	0,00173832	0,00130353				

Table A.2.: Computed errors for Courant number = 0.04, advection equation, discontinuous Galerkin method

A.1.3. Advection equation - discontinuous Galerkin and midpoint method

A.1.4. Shallow water equations - finite volumes

A.1.5. Shallow water equations - discontinuous Galerkin

A.1.6. Shallow water equations - discontinuous Galerkin and midpoint method

No. of intervals	16	32	64	128	256	512
Error	0,160879	0,0309538	0,0073533	0,00283959	0,00133784	0,000659513
No. of intervals	1024					
Error	0,000328546					

Table A.3.: Computed errors for Courant number = 0.04, advection equation, discontinuous Galerkin method, midpoint method

No. of intervals	25	50	100	200	400	800
Error (h)	0,63981	0,910011	0,743865	0,513795	0,387709	0,283215
No. of intervals	1200	1600	2000	5000	10000	
Error (h)	0,234891	0,206025	0,187382	0,104746	0,0712021	

Table A.4.: Computed errors for Courant number = 0.4, SWE, finite volumes method

No. of intervals	25	50	100	200	400	800
Error (h)	x	x	x	x	x	x
No. of intervals	1200	1600	2000	5000		
Error (h)	x	x	x	x		

Table A.5.: Computed errors for Courant number = 0.4, SWE, discontinuous Galerkin method

No. of intervals	25	50	100	200	400	800
Error (h)	x	x	x	x	x	x
No. of intervals	1200	1600	2000	5000		
Error (h)	x	x	x	x		

Table A.6.: Computed errors for Courant number = 0.4, SWE, discontinuous Galerkin method, midpoint method

List of Figures

List of Tables

Bibliography

[al.16] O. D. et al. *SWASHES: a compilation of Shallow Water Analytic Solutions for Hydraulic and Environmental Studies*. 2016.

[JSH08] T. W. Jan S. Hesthaven. *Nodal Discontinuos Galerkin Methods - Algorithms, Analysis, and Applications*. Texts in Applied Mathematics. Springer, 2008. ISBN: blub.

[Lev04] R. J. Leveque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2004. ISBN: bla.

[Ret] S. Rettenberger.

[Sto57] J. J. Stoker. *Water Waves - The Mathematical Theory with Applications*. Pure and Applied Mathematics - A Series of Texts and Monographs. Interscience Publishers, Inc., New York, 1957. ISBN: ...

[Stu13] U. Stuttgart. *Skript Partielle Differentialgleichungen II*. 2013.