Discontinuous Galerkin Methods for Numerical Weather Prediction

Literature Study

C. Caljouw

TU Delft

Discontinuous Galerkin Methods for Numerical Weather Prediction

Literature Study

by

C. Caljouw

in partial fulfilment of the requirements for the degree of
Master of Science in Applied Mathematics
at the Delft University of Technology,
to be defended publicly on Tuesday June 13th, 2017.

Daily supervisors:	Dr. ir. D. Den Ouden	TU Delft
	Prof. dr. A. P. Siebesma	The Royal Netherlands Meteorological Institute
Responsible professor:	Prof. dr. ir. C. Vuik	TU Delft

TU Delft

List of Figures

List of Tables

Nomenclature

Mathematical Notation

$\ell_j^k(x)$	Basis function of element I_k of the nodal form of DG
$\hat{a}_j^k(x_j^k, t)$	Nodal coefficient of polynomial order j and element I_k at grid point x_j^k and time t
$\hat{a}_j^k(t)$	Modal coefficient of polynomial order j and element I_k at time t
Ω	Domain of computation
$\otimes$	Outer product
$\psi_j(x)$	Basis function of the modal form of DG
φ	Quantity of interest $\varphi(x,t)$ of the advection equation
φ_0	Initial condition $\varphi_0(x)$ of the quantity of interest $\varphi(x,t)$
F_1^k	Flux matrix of element I_k depending on element I_k
F_2^k	Flux matrix of element I_k depending on element I_{k-1}
I_k	Element of the computational domain $I_k \subset \Omega$
K	Number of non-overlapping elements in Ω
M^k	Mass matrix of element I_k
N	Polynomial degree of the basis functions
S^k	Stiffness matrix of element I_k
V	Vandermonde matrix

Physical Scalars and Constants

α	Specific volume	$\mathrm{m^3\,kg^{-1}}$
$\boldsymbol{\Omega}$	Coriolis force	
$\boldsymbol{g}$	Effective gravity	$\mathrm{m\,s^{-2}}$
$\boldsymbol{u}$	Velocity field	$\mathrm{m\,s^{-1}}$
ρ	Density	$\mathrm{kg\,m^{-3}}$
c_p	Heat capacity at constant pressure for dry air	$1004.7\ \mathrm{J\,kg^{-1}\,K^{-1}}$
e	Specific internal energy	$\mathrm{J\,kg^{-1}}$
K_h	Eddy diffusivity coefficient	
K_m	Eddy viscosity coefficient	
L	Latent heat of vaporization	$2.5 \times 10^6\ \mathrm{J\,kg^{-1}}$
m	Mass	kg
m_d	Mass of dry air	kg

m_l	Mass of liquid water	kg
m_v	Mass of water vapour	kg
p	Pressure	$\mathrm{N\,m^{-2}}$
q_c	Cloud liquid water specific humidity	$\mathrm{kg\,kg^{-1}}$
q_t	Total water specific humidity	$\mathrm{kg\,kg^{-1}}$
q_v	Water vapour specific humidity	$\mathrm{kg\,kg^{-1}}$
R_d	Gas constant for dry air	$287\ \mathrm{J\,kg^{-1}\,K^{-1}}$
T	Temperature	K

Contents

1 Introduction

Mathematically modelling atmospheric phenomena is one of the two main challenges of numerical weather prediction (NWP) and climate models. The most important atmospheric processes are turbulence, convection and cloud formation. However, the coarse grid of the models resolve these processes that are smaller than the grid size. Therefore, these sub-grid processes are parametrized in the NWP and climate models. For parametrization development, the processes are simulated in a model with a higher resolution. At the Royal Netherlands Meteorological Institute (KNMI), the Dutch Atmospheric Large-Eddy Simulation model, also known as DALES, is used. On top of that, DALES can be used to predict weather on a smaller domain with a higher resolution, for example to provide short-range forecasts for near surface wind and solar power for the renewable energy sector.

Nevertheless, DALES can still be improved, especially the implemented advection schemes. Each finite difference advection scheme of DALES has its own favourable properties, like computational time or accuracy. However, the implemented high accuracy methods are still too diffusive and/or dispersive when steep gradients in temperature, moisture and momentum are present.

The other main challenge of NWP and climate models is to evaluate these models as accurate and efficiently as possible. This can be done by using all the available computational resources. The architecture of a finite difference scheme does not take full advantage of the architecture of the modern-day computers and is thus not the most computational efficient method.

To solve these two problems, the discontinuous Galerkin (DG) method is suggested. DG is an attractive method, because it allows discontinuities, it has a geometric flexibility and has a high parallel-scalability due to a compact stencil. However, it is known that non-physical oscillations are generated with DG. Therefore, a limiter has to be added.

The question of this thesis project is if DG has a faster computation or a higher accuracy, without increasing the running time too much, than the already implemented schemes in DALES. In this literature study, the DG method is investigated for a simple one-dimensional advection equation and compared to several of the implemented finite difference schemes of DALES. Moreover, the dispersion errors are removed by adding a moment limiter.

1.1. Outline

In Chapter 2, the background of DALES is given by explaining the origin of the model equations. Thereafter, in Chapter 3 the numerical methods of DALES are explained and the shortcomings of the implemented advection schemes are shown with numerical results. Chapter 4 describes the discontinuous Galerkin method for a one-dimensional advection equation and the moment limiter. On top of that, the numerical results are shown for both DG with and without limiter, including convergence and computational time. Then in Chapter 5 the open problems and issues are discussed that need to be reflected on before DG can be implemented in DALES. The literature study is concluded in Chapter 6 with a

summary and conclusion.

2

Atmospheric Modelling

In this chapter some general information is given on the atmosphere. Thereafter, the general equations for atmospheric modelling are shown and explained. Finally, an explanation is given on the adjustments for the governing equations of DALES and its prognostic variables.

2.1. The Atmosphere

The atmosphere is a layer of multiple gasses, known as air, around the Earth that is kept in place by the Earth's gravity. The atmosphere consists of a number of layers:

- Troposphere,
- Stratosphere,
- Mesosphere,
- Thermosphere,
- Exosphere.

Each layer has different properties like the composition and temperature. In Figure 2.1 [6], the temperature and air pressure of the layers are given as a function of height.

Most weather takes place in the troposphere. Many different processes that take place in the air cause all the different weather types. The most important ones are turbulence, convection and particularly, cloud formation. These three processes are results of the movement of air in the atmosphere.

2.1.1. Air movement

As the Earth's surface warms up, water evaporates and also the air near the surface warms up. Warm air and moist air are lighter than cold air and dry air, therefore, the air parcel with water vapour travels higher into the sky. How far the air parcel rises, depends on the temperature of the surrounding air and the amount of water vapour the air parcel holds.

During the upward motion of the air parcel, the air pressure around the air parcel descends, as a result the air parcel expands. Due to the expansion the air parcel loses energy and consequently, the air parcel cools down a bit. At a certain point, the air pressure and the temperature are so low that the air parcel cannot hold the amount of water vapour any more; the air parcel is saturated. Further cooling leads to condensation, however, the air parcel keeps rising by the release of latent heat during condensation. For this reason, the water droplets can get higher in the sky which even can become ice crystals.

The tiny water droplets and ice crystals are so small and light that they are able to stay up in the air. When there is a visible amount of tiny water droplets, ice crystals or a mixture of both, it forms a cloud

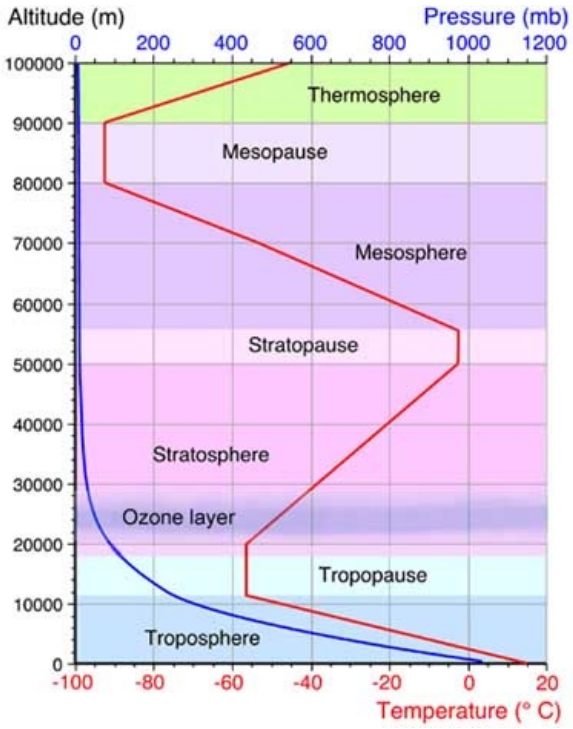


Figure 2.1: Temperature and pressure of the atmospheric layers. Image taken from [6].

in the sky.

Ultimately the sun is the reason behind all rising motion. Other factors causing air to rise or cool, are [19]:

- Orography - Air parcels are forced to rise because of physical obstacles like mountains.
- Large scale convergence - Streams of air flowing from different directions are forced to rise where they meet.
  - For example frontal systems - When warm air and cold air meet each other, the warm air mass rises up over the cold air mass.
- Turbulence - Sudden changes in wind speed with height create turbulent eddies in the air. This is sometimes notable in an air plane.

Another important factor for cloud formation are the condensation nuclei, like salt, dust and smoke particles. These particles are needed for the water vapour to condense onto. This can particularly be seen as the cloud lines that air planes leave behind in the sky.

2.2. Model equations

For the movement of air, three conservation laws of momentum, mass and energy must hold. These laws must also hold in the absence of air movement. With the three laws, the time evolution of momentum, density, temperature and humidity can be described.

Conservation of momentum

First, for a small package of air, the momentum must be conserved. The *Navier-Stokes equations* describe the differential equations that the motion of air in the atmospheric boundary layer must satisfy:

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p - 2\rho(\boldsymbol{\Omega} \times \boldsymbol{u}) - \rho \boldsymbol{g}, \tag{2.1}$$

where ρ denotes the density, $\boldsymbol{u}$ the velocity field, p the pressure, $\boldsymbol{\Omega}$ the Coriolis force, which is the force that rotates the Earth, and $\boldsymbol{g}$ the effective gravity, which is the sum of the true gravity and the centrifugal force. The equation is derived from Newton's second law for motion relative to a rotating coordinate frame and says that the acceleration that follows the relative motion in the rotating frame is equal to the sum of the effective gravity, the pressure gradient and the Coriolis force. More information of the derivation of the equation can be found in [11].

Conservation of mass
Second, conservation of mass must hold which is known as the *continuity equation*:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0. \tag{2.2}$$

This means that the local rate of change of density equals the mass divergence.

Conservation of energy
Third, the energy is conserved. By all means the change in specific internal energy of the system e is equal to the difference between the heat added to the system dQ and the work done by the system dw, known as the first law of thermodynamics (per unit mass):

$$de = dQ - dw. \tag{2.3}$$

To create prognostic equations for thermodynamic variables like temperature, the ideal gas law is needed to rewrite the first law of thermodynamics:

$$p = \rho R_d T = \frac{R_d T}{\alpha}, \tag{2.4}$$

where R_d is the gas constant, T the temperature and α is the specific volume.
Using Legendre transformations, the ideal gas law (2.4) and other state functions, especially with the use of the enthalpy h (which is defined as $h = e + p\alpha$), equation (2.3) can be rewritten:

$$dh = c_p dT = Tds + \alpha dp, \tag{2.5}$$

where c_p is the heat capacity at constant pressure for dry air and ds is the change in entropy.

The heating of air changes both the temperature and the pressure of the air parcel. Therefore, the "potential" temperature of the air parcel is defined.

When a dry air parcel is adiabatically moved, it means that the system does not lose or gain heat ($dQ = Tds = 0$). Using this information, equation (2.4) and integrating (2.5) from state $T_1 = T$, $p_1 = p$ to the reference state $T_0 = \theta$, $p_0 = 1000\,\mathrm{hPa}$, we obtain:

$$\theta = T\left(\frac{p}{p_0}\right)^{-R_d/c_p} \Leftrightarrow T = \theta\Pi, \tag{2.6}$$

in which Π is the exner function given by $\Pi = \left(\frac{p}{p_0}\right)^{R_d/c_p}$.

The potential temperature θ describes what the temperature that a dry air parcel at a pressure p and temperature T would have if it were compressed or expanded to the standard pressure p_0. Consequently, we have redefined the temperature such that the pressure contribution is removed. Moreover, θ is conserved under dry adiabatic changes.

However, an air parcel can contain water vapour or even little water droplets. For this mixture, another variable must be introduced which is also conserved under phase transition.

Since the air is mixed, the mass of air can be written as the mass of liquid water m_l, water vapour m_v and dry air m_d:

$$m = m_l + m_v + m_d. \tag{2.7}$$

Instead of the masses, the mass fractions q are used in equations.

$$q_c = \frac{m_l}{m}, \quad q_v = \frac{m_v}{m}, \quad q_t = q_v + q_c. \tag{2.8}$$

The total water specific humidity q_t is the sum of the water vapour specific humidity q_v and the cloud liquid water specific humidity q_c. One can understand that q_t is conserved under the phase transitions from liquid water to water vapour and vice versa.

Moreover, a temperature can be constructed which is also conserved under the phase transition. During the phase transition latent heat is released, i.e. $dQ = Tds = Ldq_c$ where L is the latent heat of vaporization. Therefore, when using the same tricks as for (2.6), we get a temperature conserved under the phase transition:

$$\theta_l = \theta \exp\left(\frac{-L}{c_p T} q_c\right) \approx \theta - \frac{L}{c_p \Pi} q_c. \tag{2.9}$$

The liquid water potential temperature θ_l can be linearly approximated, since q_c is usually very small ($\sim 10^{-3}$, a few grams of liquid water per kilogram of air mixture). For more information on the thermodynamics of atmospheric modelling can be found in [11] and [1].

When there is no precipitation or other explicit sources, θ_l and q_t are conserved and for conserved variables $\phi(x, y, z, t)$ it holds that the total derivative $\frac{D\phi}{Dt} = 0$. Writing out the total derivative, we obtain:

$$\begin{aligned} \frac{D\phi}{Dt} &= \frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial \phi}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial \phi}{\partial z}\frac{\partial z}{\partial t} \\ &= \frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \nabla \phi \qquad\qquad = 0, \end{aligned} \tag{2.10}$$

which is the advection equation. If there is an explicit source S_ϕ, for example precipitation, the equation is given by:

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \nabla \phi = S_\phi. \tag{2.11}$$

2.3. Governing equations of DALES

DALES is a Large-Eddy simulation, for this reason the equations are slightly different from equations (2.1), (2.2) and (2.11).

A Large-Eddy simulation is a mathematical model for turbulence. For the simulation of the time evolution of the air flow, all different time and length scales affect the flow field and therefore, all must be resolved. However, the difference between the largest ($\sim 1\,\mathrm{km}$) and smallest scales ($\sim 1\,\mathrm{mm}$) of eddies are huge. LES models reduce the computational cost by ignoring the smallest length scales.

The eddies that are smaller than the grid size, called the sub-grid scales, are filtered out of the numerical solution. This is done by using a low-pass filter, which can be seen as an averaging of the flow quantities in time and space. The sub-grid scale dynamics are subsequently parametrized by the sub-grid model. For more details on LES models, the book of Berselli et al. [2] can be read.

The anelastic approximation[1] is also applied on the filtered equations of DALES. These filtered equations are derived by Böing and can be found in the appendix of his dissertation [4]. Since the advection

[1] The anelastic approximation takes the density differences in the vertical direction of the continuity equation into account, while the Boussinesq approximation, used in the previous version DALES 3.2, takes no density variations into account unless the density is multiplied by the gravitational acceleration.

equation is the centrepiece of the thesis, the filtered equation is stated:

$$\frac{\partial}{\partial t}\tilde{\varphi} = -\frac{1}{\rho_0}\nabla \cdot \rho_0 \tilde{\varphi}\tilde{\boldsymbol{u}} + \frac{1}{\rho_0}\nabla \cdot \rho_0 K_h \nabla\tilde{\varphi} + S_\varphi. \tag{2.12}$$

Here the filtered variables are denoted by $\tilde{}$ and K_h is the eddy diffusivity coefficient. Hence a diffusion term is added as a result of the filtering. However, the diffusion term is ignored in this thesis, since the sub-grid diffusion is solved in a separate routine.

Prognostic variables
The three mandatory prognostic variables of DALES are the velocity $\boldsymbol{u}$, the liquid water potential temperature θ_l and the sub-filter scale turbulence kinetic energy e, which is used in the parametrization of the sub-filter scale dynamics. On top of that, the total water specific humidity q_t, the rain water specific humidity q_r, the rain droplet number concentration N_r, and up to 100 passive and reactive scalars can be included. However, the additional prognostic variables do not have to be calculated unless these are used. Figure 2.2 shows the processes of all variables and how the variables are affected by them.

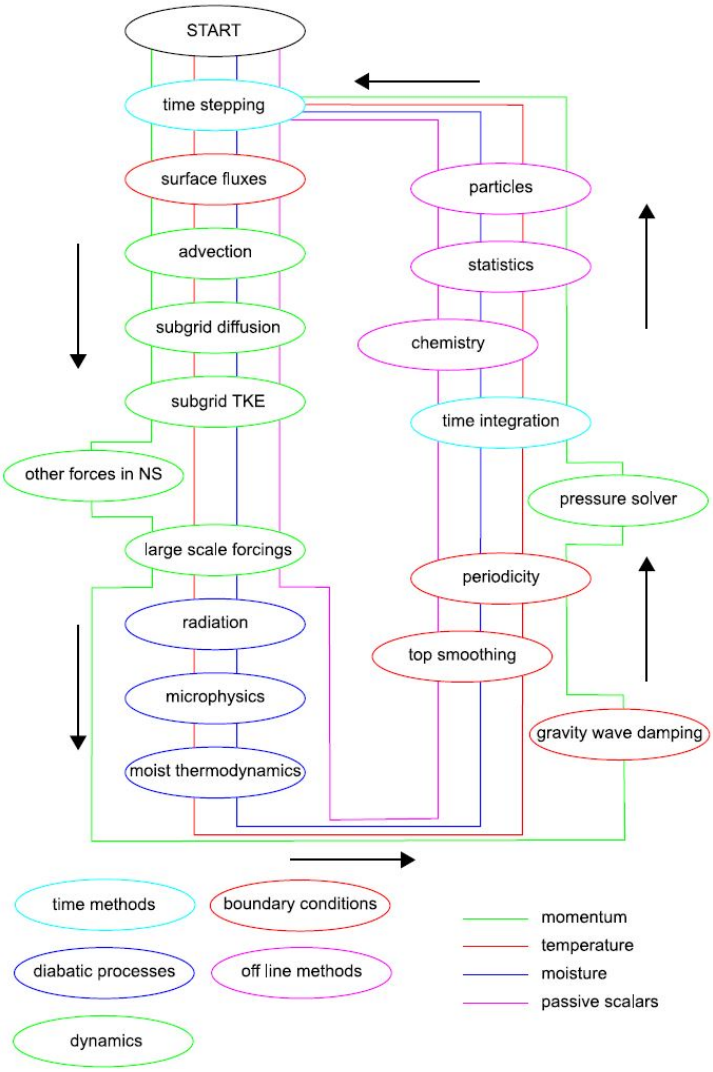


Figure 2.2: Flowchart of DALES. Image taken from [10].

More information on the other processes of DALES than the advection process can be found in [10].

3

Numerical methods of DALES for the advection equation

The present chapter describes the numerical methods that are important for the advection equation in DALES. First, the spacial discretisation of the variables are specified and thereafter, the time integration method. Last but not least, the finite difference methods are described, among others their numerical solutions and their shortcomings.

3.1. Grid spacing

In DALES, a uniform Cartesian grid is used in the horizontal directions with optional stretching in the z-direction. Normally the horizontal grid sizes Δx and Δy are $100\,\mathrm{m}$ and the vertical grid size dz is around $50\,\mathrm{m}$. On top of that, a staggered grid in space is used as an Arakawa C-grid which can be seen in Figure 3.1. An Arakawa C-grid defines the pressure p, the SFS-TKE e and the scalars φ in the centre of the cell, while the velocity components, u, v, and w, are defined at the faces of the cell, respectively. As w is given at a different height than the other variables, this level is called the half level, denoted by z_h and the other variables are at the full level z_f. This is summarized in Table 3.1.

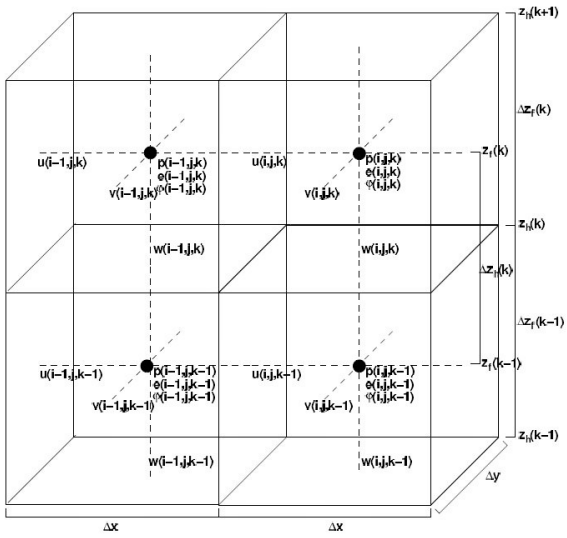


Figure 3.1: Arakawa C-grid. Image taken from [10].

Variable	Position	z level
p, e, φ	$\boldsymbol{x} + \frac{1}{2}(\Delta x, \Delta y, \Delta z)$	z_h
u	$\boldsymbol{x} + \frac{1}{2}(0, \Delta y, \Delta z)$	z_h
v	$\boldsymbol{x} + \frac{1}{2}(\Delta x, 0, \Delta z)$	z_h
w	$\boldsymbol{x} + \frac{1}{2}(\Delta x, \Delta y, 0)$	z_f

Table 3.1: Position of the variables in the Arakawa C-grid.

3.2. Time integration method

A time integration method solves the following equation:

$$\frac{\partial}{\partial t}\phi = g(\phi). \tag{3.1}$$

Thus, the differential equations are rewritten so that it has the same form as in (3.1).

DALES uses Runge Kutta 3 as the time integration method. This method calculates the next time step ϕ^{n+1} as follows:

$$\phi^{*} = \phi^{n} + \frac{\Delta t}{3} g(\phi^{n}), \tag{3.2a}$$

$$\phi^{**} = \phi^{n} + \frac{\Delta t}{2} g(\phi^{*}), \tag{3.2b}$$

$$\phi^{n+1} = \phi^{n} + \Delta t\, g(\phi^{**}), \tag{3.2c}$$

where the asterisks denote the intermediate time steps.

The size of the time step Δt is determined adaptively to keep the numerical solution stable. The two criteria that limit the time steps are the Courant-Friedrichs-Lewy (CFL) condition:

$$\text{CFL} = \max\left(\left|\frac{u\Delta t}{\Delta x}\right| + \left|\frac{v\Delta t}{\Delta y}\right| + \left|\frac{w\Delta t}{\Delta z}\right|\right).$$

and the diffusion number d [26], which is needed for the diffusion terms that arose from the LES-filtering:

$$d = \max\left(\sum_{i=1}^{3} \frac{K_m \Delta t}{\Delta x_i^2}\right),$$

where K_m is the eddy viscosity coefficient.

3.3. Advection schemes

In this section the several advection schemes of DALES are explained with their results, advantages and disadvantages.

In DALES multiple advection schemes can be chosen, for example:

- First order upwind,
- Second order central,
- Fifth order upwind,
- Weighted essentially non-oscillatory (WENO) method.

Each advection scheme has their own favourable properties such as high accuracy or little computation time, however they also have their own cons. These pros and cons are given and shown by solving a test case.

As test case, the following boundary value problem is used:

$$\begin{cases} \frac{\partial \varphi}{\partial t} + \frac{\partial f(\varphi)}{\partial x} = 0 & x \in [a,b], t > 0, \\ \varphi(x,0) = \varphi_0(x) & x \in [a,b], \\ \varphi(a,t) = \varphi(b,t) & t > 0, \end{cases} \tag{3.3}$$

where $\varphi(x,t)$ is the quantity of interest and $f(\varphi) = u\varphi$ the given flux function. If the speed u is constant, the exact solution of this problem is: $\varphi(x,t) = \varphi_0(x - ut)$.

Many properties of an advection scheme can be seen by solving a test case with an initial condition that contains a smooth and a discontinuous part. For example, for domain $\Omega = [-5, 5]$ and initial condition (see Figure 3.2):

$$\varphi_0(x) = \begin{cases} \frac{1-\cos(\pi(x-1))}{2}, & x \in [-3, -1], \\ 1, & x \in [1, 3], \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

When u is set to $1\,\mathrm{m\,s^{-1}}$, then the solution $\varphi(x, t)$ is at the same place after every $10\,\mathrm{s}$ (one period), in mathematical terms

$$\varphi(x, 10c) = \varphi_0(x) \text{ for all } c \in \mathbb{N} \tag{3.5}$$

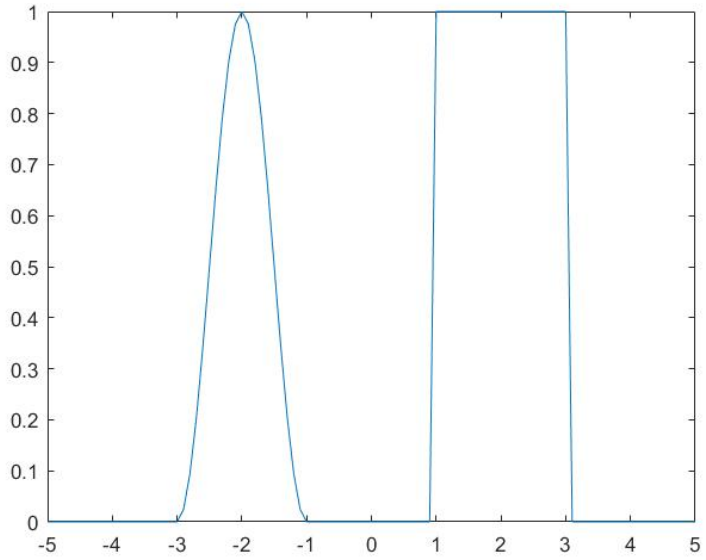


Figure 3.2: Initial condition φ_0.

In general, the advection schemes are defined such that the derivative of the flux is approximated:

$$\frac{\partial f(\varphi_i)}{\partial x} = \frac{F_{i+1/2} - F_{i-1/2}}{\Delta x}, \tag{3.6}$$

where $F_{i+1/2}$ is the convective flux of variable φ through face $i + 1/2$ of the grid box. How $F_{i+1/2}$ is defined, depends on the choice of the advection scheme.

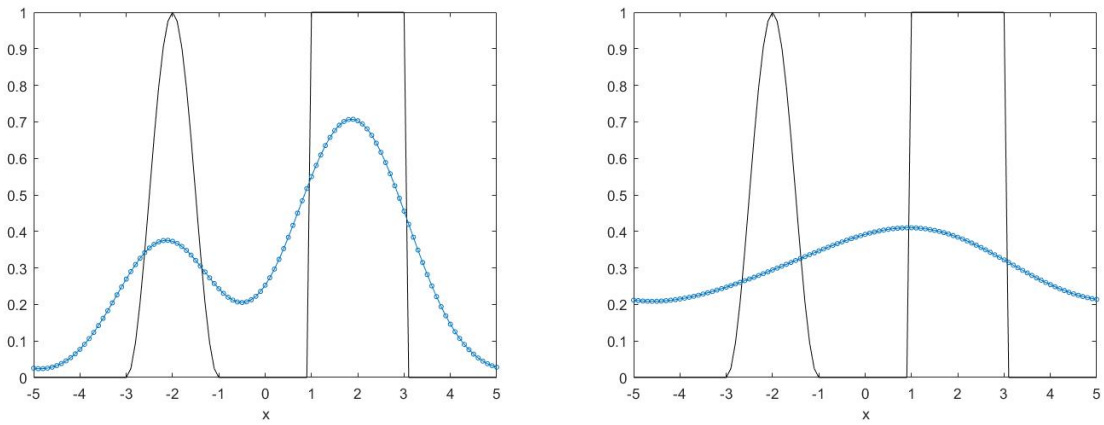
First order upwind

The first order upwind is a simple first order method which defines the flux as follows:

$$\hat{F}^{1\text{st}}_{i+1/2} = \begin{cases} u_{i+1/2}\varphi_i & \text{if } u_{i+1/2} > 0, \\ u_{i+1/2}\varphi_{i+1} & \text{if } u_{i+1/2} < 0. \end{cases}$$

The truncation error of this method is $\mathcal{O}(\Delta x)$ when function $f(\varphi) \in C^2[-5, 5]$ [24].

In Figure 3.3a and Figure 3.3b the numerical solutions are plotted after 1 and 5 periods. Clearly, one can see that the first order upwind is overly diffuse and after only 5 periods the mass is almost completely smoothened out over the domain.



(a) First order upwind at $t = 10$. (b) First order upwind at $t = 50$.

Figure 3.3: First order upwind with $\Delta x = 0.1$ and $\Delta t = 0.3\frac{u}{\Delta x}$.

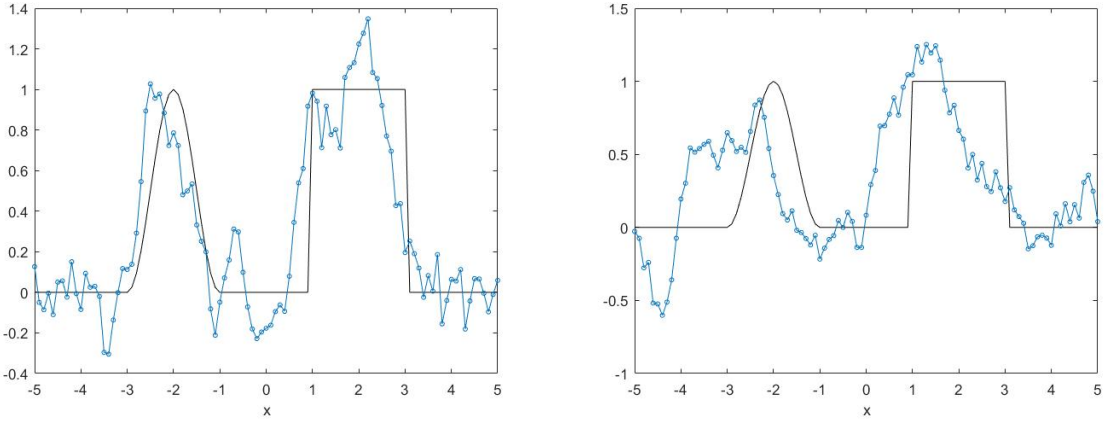
Second order central

The second order central differencing method approximates $F_{i+1/2}$ with:

$$\hat{F}^{2\text{nd}}_{i+1/2} = \frac{u_{i+1/2}}{2}(\varphi_{i+1} + \varphi_i),$$

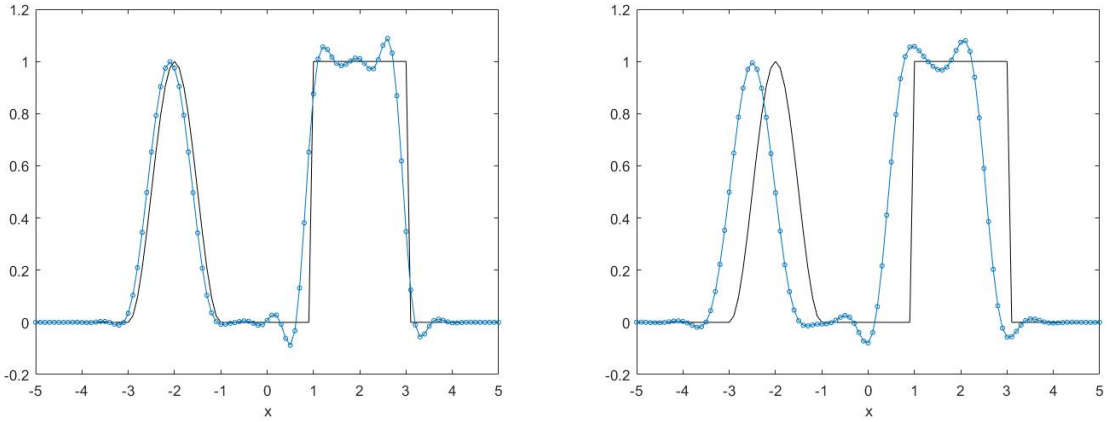
and has a truncation error of $\mathcal{O}(\Delta x^2)$ if $f(\varphi)$ is sufficiently smooth. An other interesting fact is that the first order upwind is actually second order central with extra artificial diffusion. This means that the second order central should theoretically be less diffusive than the first order upwind.

The numerical results of second order central differencing after 1 and 5 periods can be found in Figure 3.4a and in Figure 3.4b respectively. Indeed the second order central is less diffusive than the first order upwind. However, dispersive errors are present when using second order central differencing, leading to solutions that are not recognizable as the initial condition.



(a) Second order central at $t = 10$. (b) Second order central at $t = 50$.

Figure 3.4: Second order central with $\Delta x = 0.1$ and $\Delta t = 0.3\frac{u}{\Delta x}$.



(a) Fifth order upwind at $t = 10$.

(b) Fifth order upwind at $t = 50$.

Figure 3.5: Fifth order upwind method with $\Delta x = 0.1$ and $\Delta t = 0.3\frac{u}{\Delta x}$.

Fifth order upwind

For a high accuracy method with a truncation error of $\mathcal{O}(\Delta x^5)$ for sufficiently smooth functions, the fifth order upwind can be used, which approximates $F_{i+1/2}$ by:

$$\hat{F}^{5\text{th}}_{i+1/2} = \begin{cases} \frac{u_{i+1/2}}{60}\left(-3\varphi_{i+2} + 27\varphi_{i+1} + 47\varphi_i - 13\varphi_{i-1} + 2\varphi_{i-2}\right) & \text{if } u_{i+1/2} > 0, \\ \frac{u_{i+1/2}}{60}\left(-3\varphi_{i-2} + 27\varphi_{i-1} + 47\varphi_i - 13\varphi_{i+1} + 2\varphi_{i+2}\right) & \text{if } u_{i+1/2} < 0. \end{cases}$$

The numerical solutions (see in Figure 3.5a and Figure 3.5b) are less diffusive than the first order upwind results. However, there are non-physical oscillations which are bad for the model; especially the negative values are physically impossible. Moreover, every period the solution is shifted to the left, also known as a time lag.

WENO method

The WENO method is based on an essentially non-oscillatory (ENO) method. The basic concept behind the ENO method is to choose between different stencils, upwind, central or downwind such that the $\varphi_{i+1/2}$ is approximated with the stencil that does not contain a discontinuity [9][21][22].

The three different stencils are given by:

$$\begin{aligned} \hat{\varphi}^0_{i+1/2} &= a^0_0\varphi_{i-2} + a^0_1\varphi_{i-1} + a^0_2\varphi_i, \\ \hat{\varphi}^1_{i+1/2} &= a^1_0\varphi_{i-1} + a^1_1\varphi_i + a^1_2\varphi_{i+1}, \\ \hat{\varphi}^2_{i+1/2} &= a^2_0\varphi_i + a^2_1\varphi_{i+1} + a^2_2\varphi_{i+2}, \end{aligned} \tag{3.7}$$

where the coefficients a^l_k are given in Table 3.2.

a^l_k	$k=0$	$k=1$	$k=2$
$l=0$	2/6	-7/6	11/6
$l=1$	-1/6	5/6	2/6
$l=2$	2/6	5/6	-1/6

Table 3.2: Coefficients a^l_k of the different stencils.

The WENO method is a weighted ENO method that was introduced by Liu et al. [17]. Liu noted that all stencils should contribute to the value of $\varphi_{i+1/2}$ by interpolating:

$$\hat{\varphi}_{i+1/2} = \sum_{l=0}^{2} \omega_l \hat{\varphi}^l_{i+1/2}, \tag{3.8}$$

where $\hat{\varphi}^{l}_{i+1/2}$ are the three different stencils, given in (3.7), the ω_l are normalized weights that depend on a smoothness criterion β_l:

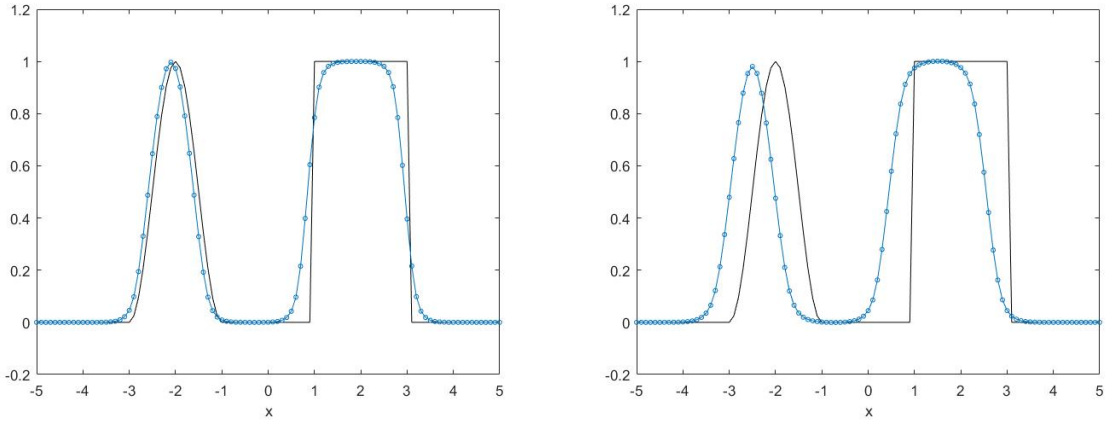
$$\omega_l = \frac{\alpha_l}{\sum_l \alpha_l}, \quad \alpha_l = \frac{C_l}{(\epsilon + \beta_l)^2}, \quad [C_0, C_1, C_2] = \frac{1}{10}[1, 6, 3].$$

The ϵ is a parameter that is very small (10^{-12}) to make sure that the denominator is never zero and the smoothness metric β_l is calculated as:

$$\begin{aligned}
\beta_0 &= \tfrac{13}{12}(\varphi_{i-2} - 2\varphi_{i-1} + \varphi_i)^2 + \tfrac{1}{4}(\varphi_{i-2} - 4\varphi_{i-1} + 3\varphi_i)^2, \\
\beta_1 &= \tfrac{13}{12}(\varphi_{i-1} - 2\varphi_i + \varphi_{i+1})^2 + \tfrac{1}{4}(\varphi_{i-1} - \varphi_{i+1})^2, \\
\beta_2 &= \tfrac{13}{12}(\varphi_i - 2\varphi_{i+1} + \varphi_{i+2})^2 + \tfrac{1}{4}(3\varphi_i - 4\varphi_{i+1} + \varphi_{i+2})^2.
\end{aligned} \tag{3.9}$$

When a completely smooth field is advected, the fifth order WENO scheme is equivalent to the fifth order upwind scheme. Therefore, the truncation error is $\mathcal{O}(\Delta x^5)$ for sufficiently smooth functions.

In Figure 3.6, the numerical results are given for $t = 10$ and $t = 50$. As with the fifth order upwind (See 3.5, a time lag is present. However, there are no oscillations. On top of that WENO is diffuse, which can be seen in the discontinuous part of the solutions, but is significantly less diffusive compared to the overly diffusive first order upwind. All in all, the WENO method is the most accurate advection scheme that can be used in DALES.



(a) WENO method at $t = 10$.

(b) WENO method at $t = 50$.

Figure 3.6: WENO method with $\Delta x = 0.1$ and $\Delta t = 0.3\frac{u}{\Delta x}$.

Computational time

The computational time of an advection scheme is, besides the accuracy of the scheme, important to the KNMI, therefore, the computational time for the various advection schemes of DALES is plotted against the grid size Δx (see Figure 3.7). For this test, a similar implementation is used in MATLAB to remove as many biases as possible.

Since WENO changes the stencils every time step and adapt them to location x, it can be expected and seen in Figure 3.7, the computational time of WENO is the largest of the four different advection schemes. The other three finite difference methods, first order upwind, second order central and fifth order upwind, do not change the stencil which is also very compact, especially for one-dimensional problems. Therefore, the computational time is less and for small grid sizes, the computational time is about the same for all three of them.

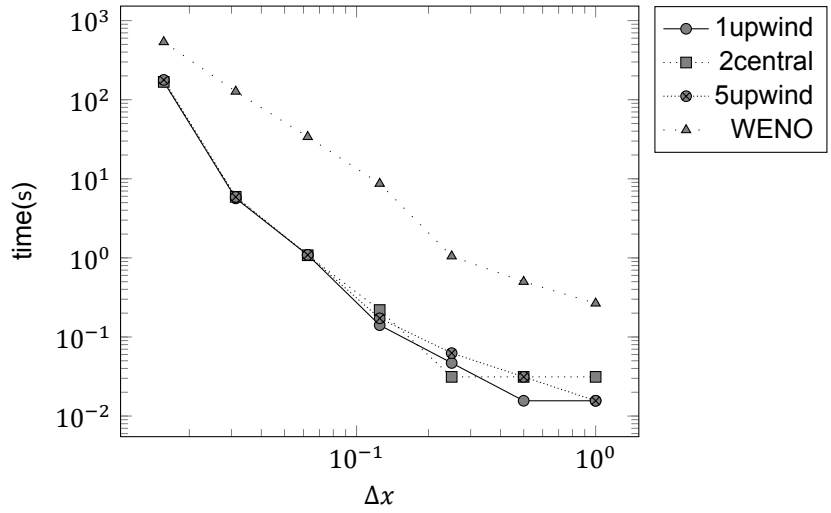


Figure 3.7: Computational time of the various advection schemes of DALES to simulate till $t = 50$ s.

3.3.1. Shortcomings of the advection schemes of DALES

The four discussed advection schemes give different results and every scheme has its own advantages and disadvantages, but none of them are as flawless as it needs to be.

The low order difference methods may have a low computational time, also when implemented for higher dimensional problems. Nonetheless, the first order upwind is overly diffusive and the second order central differencing has too many dispersive errors; the initial condition is not recognizable after a few time steps.

Also, the high accuracy methods, the fifth order upwind and WENO method, are still diffusive, though they are not overly diffusive as the first order upwind is. Moreover, the fifth order upwind is dispersive. This problem is absent when WENO is used, but instead its computational time becomes much larger.

On top of that, the WENO and fifth order upwind method have a time lag making the long simulations inaccurate. This time lag also appears in the solutions of the first order upwind and second order central. It seems that the speed of the numerical solution is lower than the actual speed u, leading to a time lag that increase in time. More research is needed to find the specific reason of the time lags. In this specific case, where the speed u is constant and the time lag is after every period of the same size, the time lag for WENO and the fifth order upwind could be resolved by shifting the solutions horizontally to remove the time lag.

As the advection scheme is important for the DALES model, an other advection scheme should be implemented such that either the accuracy is better than the implemented advection schemes of DALES or the computational time is less. The best possible outcome of this thesis project is to have an advection scheme that is better in both accuracy and computational time.

4

Discontinuous Galerkin for One-dimensional Advection Equation

In this chapter the discontinuous Galerkin (DG) method is explained by solving a one-dimensional advection equation. First, the differences are shown between DG and the standard methods: finite difference method (FDM), finite volume methods (FVM) and finite element methods (FEM). Then the basics of DG will be explained. Thereafter, the two different methods of DG are explained separately: modal discontinuous Galerkin and nodal discontinuous Galerkin in Section 4.3 and Section 4.4 respectively.

4.1. Relations and differences between FDM, FVM, FEM and DG

There are several numerical methods that can be used to solve partial differential equations (PDEs), such as:

- Finite difference method (FDM),
- Finite volume method (FVM),
- Finite element method (FEM).

The most popular method is the finite difference method which is also used in DALES.

FDM solves PDEs directly by approximating the derivatives using local Taylor expansion, while FVM solves the differential equations after integration over a control volume. For FEM, the domain is divided in a finite number of non-overlapping elements so that the numerical solution is reconstructed on every element by giving weights to specified basis functions. The weights are found by solving the equations that are obtained by using the weak form of the PDEs.

Both FEM and FDM find the nodal values while FVM give the cell average values. A disadvantage of FDM is the difficulty that comes in to play when having non-equidistant grids, while this is not a problem for FVM and FEM. Only the FVM has an advantage of guaranteeing mass conservation and allowing discontinuities.

DG is a combination of FEM and FVM. FVM is actually a DG method with constant basis functions and DG is a special case of FEM where discontinuities are allowed (see Figure 4.1). Therefore, DG has good qualities from both methods:

- discontinuities are allowed
- non-equidistant grids can be used,
- conservation of mass,
- dynamic h-p refinements (where h is the grid size and p the polynomial order of the basis function),
- high scalability - only communication between elements which share faces.

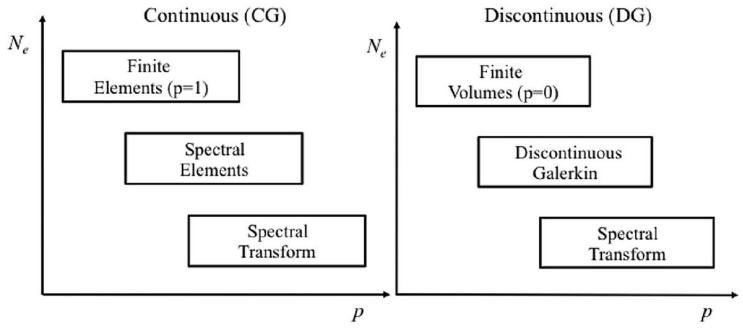


Figure 4.1: Relation between FEM, FVM and DG where N_e is a finite number of non-overlapping elements and p the order of basis functions. (Spectral elements is a finite element method with high order basis functions and spectral transform is also finite element method with only one element and high order basis functions.) Image taken from [18].

4.2. Basics of Discontinuous Galerkin

The problem that is solved in this chapter is the same simple one-dimensional advection equation (3.3) as in Section 3.3:

$$\begin{cases} \frac{\partial \varphi}{\partial t} + \frac{\partial f(\varphi)}{\partial x} = 0 & x \in [a,b], t > 0, \\ \varphi(x,0) = \varphi_0(x) & x \in [a,b], \\ \varphi(a,t) = \varphi(b,t) & t > 0, \end{cases} \tag{3.3 revisited}$$

where $\varphi(x,t)$ is the quantity of interest and $f(\varphi) = u\varphi$ the given flux function.

First, the domain $\Omega = [a,b]$ is partitioned into K non-overlapping elements $\Omega = \cup_{k=1}^{K} I_k$ with $I_k = [x_{k-1/2}, x_{k+1/2}]$, $k = 1, \dots, K$. Just as in the finite element method, the function $\varphi(x,t)$ is approximated locally on every element I_k:

$$\varphi(x,t) \cong \varphi_h(x,t) = \bigoplus_{k=1}^{K} \varphi_h^k(x,t), \text{ where}$$
$$\varphi_h^k(x,t) \in V_h^N = \{v \in L^2(a,b) : v|_{I_k} \in \mathbb{P}^N(I_k), k = 1, \dots, K\}.$$

Here $\mathbb{P}^N$ is the space of polynomials of degree N.

DG is resolved around the weak formulation of the equation which is obtained by multiplying the differential equation and initial condition with an arbitrary piecewise continuous function η and integrating over element I_k:

$$\begin{cases} \int_{I_k} \left[\frac{\partial \varphi}{\partial t} + \frac{\partial}{\partial x} f(\varphi) \right] \eta \, dx = 0, & \text{(4.1a)} \\ \int_{I_k} \varphi(x,0)\eta \, dx = \int_{I_k} \varphi_0(x)\eta \, dx. & \text{(4.1b)} \end{cases}$$

Using integration by parts, equation (4.1a) can be rewritten:

$$\int_{I_k} \frac{\partial \varphi}{\partial t}\eta - f(\varphi)\frac{\partial \eta}{\partial x} \, dx + \left[f(\varphi_h)\eta \right]_{x_{k-1/2}}^{x_{k+1/2}} = 0, \tag{4.2}$$

Hereafter, η is chosen to be a test function from V_h and φ is approximated by assigning weights to specified basis functions:

$$\varphi_h^k(x,t) = \sum_{j=0}^{N} \hat{a}_j^k(t)\psi_j(x) = \sum_{j=0}^{N} a_h^k(x_j^k,t)\ell_j^k(x), \ \forall x \in I_k. \tag{4.3}$$

The first expression $\varphi_h^k(x,t)$ is known as the *modal form* and the second the *nodal form*. Thus $\hat{a}_j^k(t)$ are the modal coefficients and $a_h^k(x_j^k,t)$ the nodal coefficients. Basis functions $\psi_j(x)$ are the functions

belonging to the modal form and $\ell_j^k(x)$ to the nodal form. With the use of the *Vandermonde matrix* which is defined as a $(k+1)\times(k+1)$ matrix V by $V_{ij} = \psi_j(x_i)$, one can switch between the two forms:

$$V\hat{\boldsymbol{a}}^k = \boldsymbol{a}_h^k. \tag{4.4}$$

In Section 4.3 the modal form of DG is used to solve problem (3.3) and in Section 4.4, the nodal form is utilized.

4.3. Modal discontinuous Galerkin Method

First, the basis functions $\psi_j(x)$ have to be chosen. The use of Legendre polynomials is a very popular choice. These polynomials are L^2-orthogonal which causes the mass matrix to become a diagonal matrix.

The Legendre polynomials are defined as follows:

$$P_n(x) = \frac{1}{2^n}\sum_{k=0}^{n}\binom{n}{k}^2 (x-1)^{n-k}(x+1)^k,$$

and its orthogonality relation is given by:

$$\int_{-1}^{1} P_n(x)P_m(x)\,dx = \frac{2}{2m+1}\delta_{nm},$$

where δ_{nm} is the Kronecker delta function.

In order to take full advantage of the Legendre polynomials, the coordinate transformation $\xi(x) = \frac{2(x-x_k)}{\Delta x_k}$ is applied such that the Legendre polynomials can be used on the reference element $[-1,1]$.

In short, the basis function η_h and the approximation φ_h are given by:

$$x \in I_k : \begin{cases} \eta_h(x) = P_i(\xi(x)), \quad i \in \{0,\dots,N\} \\ \varphi_h^k = \sum_{i=0}^{N} \hat{a}_i^k(t)P_i(\xi(x)) \end{cases}. \tag{4.5}$$

Using the above and filling this in the weak formulation (4.2), we get the following equations:

$$\forall i \in \{0,\dots,N\}: \begin{cases} \int_{I_k}\sum_{j=0}^{N}\frac{\partial}{\partial t}\hat{a}_j^k(t)P_j(\xi(x))P_i(\xi(x)) - f(\varphi_h)\frac{\partial P_i(\xi(x))}{\partial x}\,dx + \left[f(\varphi_h)P_i(\xi(x))\right]_{x_{k-1/2}}^{x_{k+1/2}} = 0, \\ \int_{I_k}\sum_{j=0}^{N}\hat{a}_j^k(t)P_j(\xi(x))P_i(\xi(x))\,dx = \int_{I_k}\varphi_0(x)P_i(\xi(x))\,dx. \end{cases}$$

As a result of the coordinate transformation, integrals and derivatives can be rewritten using $dx = \frac{\Delta x_k}{2}\,d\xi$ and $\frac{d}{d\xi}\frac{d\xi}{dx} = \frac{2}{\Delta x_k}\frac{d}{d\xi}$. Furthermore, the orthogonality of the basis functions can be used to simplify the equations:

$$\forall i \in \{0,\dots,N\}: \begin{cases} \frac{\Delta x_k}{2i+1}\frac{\partial}{\partial t}\hat{a}_i^k(t) - \int_{-1}^{1} f(\varphi_h)\frac{d}{d\xi}P_i(\xi)\,d\xi + \left[f(\varphi_h)P_i(\xi(x))\right]_{x_{k-1/2}}^{x_{k+1/2}} = 0, & \text{(4.6a)} \\ \frac{\Delta x_k}{2i+1}\hat{a}_i^k(0) = \int_{I_k}\varphi_0(x)P_i(\xi(x))\,dx. & \text{(4.6b)} \end{cases}$$

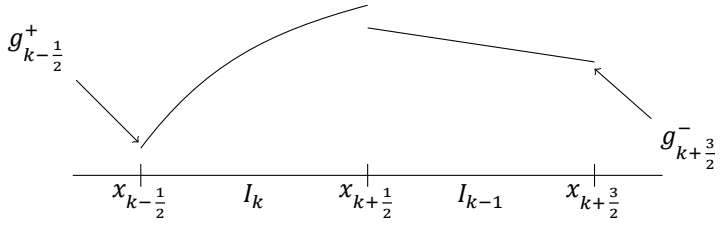


Figure 4.2: Example of a function g which is discontinuous at the element boundaries.

As the approximated function is allowed to have discontinuities, there are some ambiguities around the boundaries of the elements as can be seen in Figure 4.2. Therefore, the following notation is used to indicate which value is used:

$$\eta^+_{h,k-1/2} = \lim_{x \downarrow x_{k-1/2}} \eta_h(\xi(x)),$$
$$\eta^-_{h,k+1/2} = \lim_{x \uparrow x_{k+1/2}} \eta_h(\xi(x)).$$

With the given notation, the third term of (4.6a) can be written as:

$$\left[f(\varphi_h)\eta_h(\xi(x))\right]_{x_{k-1/2}}^{x_{k+1/2}} = \hat{F}_{k+1/2}\eta^-_{k+1/2} - \hat{F}_{k-1/2}\eta^+_{h,k-1/2},$$

where $\hat{F}_{k\pm1/2}$ denotes the numerical flux value on the boundary $x_{k\pm1/2}$ which can depend on both $x^-_{k\pm1/2}$ and $x^+_{k\pm1/2}$.

There are many choices for the numerical fluxes[16], such as:

1. Upwind,
2. Godunov,
3. Lax-Friedrich.

The DG solution is not very sensitive to the choice of numerical fluxes. This means that a very simple numerical flux suffices [18].

4.3.1. Example

For illustration, a constant speed u and a simple numerical flux, the first order upwind method, are chosen:

$$f(\varphi) = u\varphi \text{ where } u \in \mathbb{R}_{>0}, \tag{4.7a}$$
$$\hat{F}_{k\pm1/2} = f(\varphi^-_{h,k\pm1/2}). \tag{4.7b}$$

Equations (4.6) becomes for all $i \in \{0, \dots, N\}$:

$$\begin{cases} \dfrac{\Delta x_k}{2i+1}\dfrac{\partial}{\partial t}\hat{a}^k_i(t) - \displaystyle\int_{-1}^{1} u\left(\sum_{j=0}^{N}\hat{a}^k_j(t)P_j(\xi)\right)\frac{d}{d\xi}P_i(\xi)\,d\xi + \hat{F}_{k+1/2}\eta^-_{h,k+1/2} - \hat{F}_{k-1/2}\eta^+_{h,k-1/2} = 0, & \text{(4.8a)} \\ \dfrac{\Delta x_k}{2i+1}\hat{a}^k_i(0) = \displaystyle\int_{I_k}\varphi_0(x)P_i(\xi(x))\,dx. & \text{(4.8b)} \end{cases}$$

Then with the use of the definition of the chosen numerical fluxes and the basis function, it is known that $\eta^-_{h,k+1/2} = P_i(1) = 1$ and $\eta^+_{h,k-1/2} = P_i(-1) = (-1)^i$ and (4.8a) can be rewritten:

$$\frac{\Delta x_k}{2i+1}\frac{\partial}{\partial t}\hat{a}^k_i(t) - u\sum_{j=0}^{N}\hat{a}^k_j(t)\int_{-1}^{1}P_j(\xi)\frac{d}{d\xi}P_i(\xi)\,d\xi + f(\varphi^-_{h,k+1/2}) - f(\varphi^-_{h,k-1/2})(-1)^i = 0. \tag{4.9}$$

Using the flux function and the definition of φ_h, the fluxes can be rewritten:

$$f(\varphi^-_{h,k+1/2}) - f(\varphi^-_{h,k-1/2})(-1)^j = u\left(\sum_{j=0}^{N} \hat{a}_n^k(t)P_n(1)\right) - u\left(\sum_{j=0}^{N} \hat{a}_n^{k-1}(t)P_n(1)\right)(-1)^j = 0, \quad (4.10a)$$

$$= u\left(\sum_{j=0}^{N} \hat{a}_n^k(t)\right) - (-1)^j u\left(\sum_{j=0}^{N} \hat{a}_n^{k-1}(t)\right). \quad (4.10b)$$

Hence, element I_{k-1}, which is the left neighbour of element I_k, is needed.

Altogether, the following equations have to be solved for $k \in \{1, \dots, K\}$:

$$\begin{cases} \dfrac{\Delta x_k}{2i+1}\dfrac{\partial}{\partial t}\hat{a}_i^k(t) - u\displaystyle\sum_{j=0}^{N} \hat{a}_j^k(t)\int_{-1}^{1} P_j(\xi)\frac{d}{d\xi}P_i(\xi)\,d\xi + \left(\sum_{j=0}^{N} \hat{a}_j^k(t)\right) - (-1)^i\left(\sum_{j=0}^{N} \hat{a}_j^{k-1}(t)\right) = 0, & (4.11a) \\ \dfrac{\Delta x_k}{2i+1}\hat{a}_i^k(0) = \displaystyle\int_{I_k} \varphi_0(x)P_i(\xi(x))\,dx, & (4.11b) \end{cases}$$

for all $i \in \{0, \dots, N\}$, which is in matrix form:

$$\begin{cases} M^k\dfrac{\partial}{\partial t}\hat{\boldsymbol{a}}^k - S^k\hat{\boldsymbol{a}}^k + F_1^k\hat{\boldsymbol{a}}^k - F_2^k\hat{\boldsymbol{a}}^{k-1} = 0, & (4.12a) \\ M^k\hat{\boldsymbol{a}}^k(0) = \left(\displaystyle\int_{I_k} \varphi_0 P_i(\xi(x))\,dx\right)_i = \tilde{\boldsymbol{\varphi}}_h^k. & (4.12b) \end{cases}$$

Suppose we take linear basis functions ($N = 1$), then the mass matrix of element k is:

$$M^k = \begin{pmatrix} \Delta x_k & 0 \\ 0 & \frac{\Delta x_k}{3} \end{pmatrix},$$

the stiffness matrix:

$$S_{jn}^k = u\int_{-1}^{1} P_n(\xi)\frac{d}{d\xi}P_j(\xi)\,d\xi \iff S^k = u\begin{pmatrix} 0 & 0 \\ 2 & 0 \end{pmatrix},$$

and the numerical fluxes for element k, which also depend on element $k-1$:

$$F_1^k = u\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix},\ F_2^k = u\begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}.$$

The same domain $\Omega = [-5, 5]$ and initial condition as in Section 3.3 is taken:

$$\varphi_0(x) = \begin{cases} \frac{1-\cos(\pi(x-1))}{2}, & x \in [-3, -1] \\ 1, & x \in [1, 3] \\ 0, & \text{otherwise} \end{cases} \quad (4.13)$$

The initial vector $\hat{\boldsymbol{a}}^k(0)$ is found by solving $M^k\hat{\boldsymbol{a}}^k(0) = \tilde{\boldsymbol{\varphi}}_h^k$, giving the initial condition projected to the finite element space. The L_2-projection of the initial condition, $\varphi_h(x, 0)$, is the φ which minimizes $\|\varphi_0(x) - \varphi_h(x, 0)\|_2$. For this the initial condition φ_0 is multiplied with the test function and integrated

exactly:

$$\tilde{\boldsymbol{\varphi}}_h^k = \begin{pmatrix} \int_{I_k} \varphi_0 P_0(\xi(x))\, dx \\ \int_{I_k} \varphi_0 P_1(\xi(x))\, dx \end{pmatrix}, \text{ with} \tag{4.14a}$$

$$\int_{I_k} \varphi_0 P_0(\xi(x))\, dx = \begin{cases} \frac{\Delta x_k}{4}\left[\xi - \frac{1}{a}\sin(a\xi + b)\right]_{-1}^{1}, & I_k \subseteq [-3,-1], \\ \Delta x_k, & I_k \subseteq [1,3], \\ 0, & \text{otherwise}, \end{cases} \text{, and} \tag{4.14b}$$

$$\int_{I_k} \varphi_0 P_1(\xi(x))\, dx = \begin{cases} \frac{\Delta x_k}{4}\left[\frac{1}{2}\xi^2 - \frac{1}{a^2}\left(a\xi\sin(a\xi + b) + \cos(a\xi + b)\right)\right]_{-1}^{1}, & I_k \subseteq [-3,-1], \\ 0, & I_k \subseteq [1,3], \\ 0, & \text{otherwise}, \end{cases} \tag{4.14c}$$

where $a = \frac{\pi \Delta x_k}{2}$ and $b = \pi(x_k - 1)$.

4.4. Nodal discontinuous Galerkin Method

For the basis function $\ell_j^k(x)$, we take the Lagrangian polynomials, which are based on the Legendre-Gauss-Lobatto (LGL) points; those are $N+1$ nodes in the interval $[-1,1]$ that satisfy:

$$(1 - x^2)P_N'(x) = 0. \tag{4.15}$$

One can also take $N+1$ equidistant nodes, however, the extrema and minima of the Lagrangian polynomials can get out of control when N becomes greater. In [18] the problems of using equidistant nodes are explained more thoroughly. For this reason we chose to use LGL nodes.

After we have chosen the basis functions, we can fill in

$$\forall x \in I_k : \begin{cases} \eta_h(x) = \ell_i(\xi(x)),\ i \in \{0, \dots, N\}, & \text{(4.16a)} \\ \varphi_h^k = \sum_{j=0}^{N} a_h^k(x_j^k, t)\ell_j(\xi(x)), & \text{(4.16b)} \end{cases}$$

into the weak formulation (4.1), $\forall i \in \{0, \dots, N\}$:

$$\begin{cases} \int_{I_k} \sum_{j=0}^{N} \frac{\partial}{\partial t} a^k(x_j^k, t)\ell_j(\xi(x))\ell_i(\xi(x)) - f(\varphi_h)\frac{\partial \ell_i(\xi(x))}{\partial x}\, dx + \left[f(\varphi_h)\ell_i(\xi(x))\right]_{x_{k-1/2}}^{x_{k+1/2}} = 0, & \text{(4.17a)} \\ \int_{I_k} \sum_{j=0}^{N} \hat{a}^k(x_j^k, t)\ell_j(\xi(x))\ell_i(\xi(x))\, dx = \int_{I_k} \varphi_0(x)\ell_i(\xi(x))\, dx. & \text{(4.17b)} \end{cases}$$

4.4.1. Example

Just as in the example of the modal form (4.3.1), we take a simple flux and numerical flux as given in (4.7), domain $\Omega = [-5,5]$ and the same initial condition given in (4.13).

For $N = 1$, the coordinate transformation $\xi(x) = \frac{x - x_{k-1/2}}{\Delta x}$ is used to take advantage of a linear coordinate transformation for the integration. Hence the element matrices of the matrix-vector form of the

problem, see (4.12a), are defined as:

$$M_{ij}^k = \Delta x \int_0^1 \ell_i(\xi)\ell_j(\xi)\, d\xi \Rightarrow M = \Delta x \begin{pmatrix} \frac{2}{3} & \frac{2}{6} \\ \frac{2}{6} & \frac{2}{3} \end{pmatrix}, \tag{4.18a}$$

$$S_{ij}^k = u \int_0^1 \ell_j(\xi)\frac{d}{d\xi}\ell_i(\xi)\, d\xi \Rightarrow S = u \begin{pmatrix} -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \tag{4.18b}$$

$$F_1^k = u \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}, \tag{4.18c}$$

$$F_2^k = u \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}. \tag{4.18d}$$

To find the L_2-projection of the initial condition, the initial condition multiplied with the basis function is calculated exactly:

$$\tilde{\varphi}_h^k = \begin{pmatrix} \int_{I_k} \varphi_0 \ell_0(\xi(x))\, dx \\ \int_{I_k} \varphi_0 \ell_1(\xi(x))\, dx \end{pmatrix}, \tag{4.19a}$$

$$\int_{I_k} \varphi_0 \ell_0(\xi(x))\, dx = \begin{cases} \frac{\Delta x}{2}\left[\xi - \frac{1}{2}\xi^2 + \frac{1}{a^2}\cos(a\xi + b) + (\xi - 1)\frac{1}{a}\sin(a\xi + b)\right]_0^1, & I_k \subseteq [-3, -1], \\ \frac{\Delta x}{2}, & I_k \subseteq [1, 3], \\ 0, & \text{otherwise}, \end{cases} \tag{4.19b}$$

$$\int_{I_k} \varphi_0 \ell_1(\xi(x))\, dx = \begin{cases} \frac{\Delta x}{2}\left[\frac{1}{2}\xi^2 - \frac{1}{a^2}\left(a\xi\sin(a\xi + b) + \cos(a\xi + b)\right)\right]_0^1, & I_k \subseteq [-3, -1], \\ \frac{\Delta x}{2}, & I_k \subseteq [1, 3], \\ 0, & \text{otherwise}, \end{cases} \tag{4.19c}$$

where $a = \pi\Delta x$ and $b = \pi(x_{k-1/2} - 1)$.

For an $N > 1$ and particularly if the initial condition is a vector of values instead a function, then integrating exactly cannot be done. This problem can be solved by using quadrature rules. The quadrature rule that works well with the LGL nodes x_i is the Lobatto-Gauss-Legendre quadrature:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{N} \omega_i f(x_i) \quad \text{with } \omega_i = \frac{2}{N(N+1)(P_N(x_i))^2}. \tag{4.20}$$

This rule is exact for all polynomials of order $2N - 1$ or less [12].

The linear coordinate transformation $\xi(x) = \frac{2(x - x_k)}{\Delta x}$ is used in order to have reference element $[-1, 1]$

where also the LGL nodes lay. The following element matrices are obtained for (4.12a):

$$M_{ij}^k = \int_{I_k} \ell_i(\xi(x))\ell_j(\xi(x))\,dx = \frac{\Delta x}{2}\int_{-1}^{1} \ell_i(\xi)\ell_j(\xi)\,d\xi \approx \frac{\Delta x}{2}\sum_{p=0}^{N} \omega_p \ell_i(\xi_p)\ell_j(\xi_p) = \frac{\Delta x}{2}\omega_i\delta_{ij}, \tag{4.21a}$$

$$S_{ij}^k = u\int_{I_k} \ell_j(\xi(x))\frac{d\ell_i(\xi(x))}{dx}\,dx = u\int_{-1}^{1} \ell_j(\xi)\frac{d\ell_i(\xi)}{d\xi}\,d\xi \approx u\sum_{p=0}^{N} \omega_p \ell_j(\xi_p)\ell_i'(\xi_p) = u\omega_j\ell_i'(\xi_j), \tag{4.21b}$$

$$F_{1,ij} = f(\varphi_{k+1/2}^-)\ell_i^k(x_{k+1/2}^-) \Rightarrow F_1 = u\begin{pmatrix} 0 & \dots & \dots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & -1 \end{pmatrix}, \tag{4.21c}$$

$$F_{2,ij} = f(\varphi_{k-1/2}^-)\ell_i^k(x_{k-1/2}^-) \Rightarrow F_2 = u\begin{pmatrix} 0 & \dots & 0 & 1 \\ \vdots & \ddots & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & 0 \end{pmatrix}. \tag{4.21d}$$

The integration for the L_2-projection of the initial condition can also be calculated using the LGL quadrature:

$$\tilde{\varphi}_i^k = \int_{I_k} \varphi_0\ell_i(\xi(x))\,dx \approx \frac{\Delta x}{2}\sum_{p=0}^{N} \varphi_0(\xi_p)\omega_p\ell_i(\xi_p) = \frac{\Delta x}{2}\varphi_0(\xi_i)\omega_i. \tag{4.22}$$

4.5. Numerical Experiments

In this section, several numerical experiments are done with DG. First of all, the reason for the nodal representation is given. Thereafter, the CFL restrictions for the time integrations and the error calculation are given. Next, the numerical results are shown and discussed.

As we have seen, there are two sorts of DG: nodal and modal, although as given in (4.3), the solutions are equivalent. Certainly, one can see that in Figure 4.3 there is no difference between the modal and nodal solutions. However, each representation has its own favourable properties. The modal form can be handy when going from order $N-1$ to N since only one extra basis function is added. This is not the case for the nodal form. Moreover, most limiters are used on the modal form of the solutions. However, for the nodal representation there is no need to transfer between the spectral and physical space, making it easier to plot and implementing the boundary and initial conditions. Also for the definition of element continuity the nodal form is handier. Therefore, the nodal representation is used in this thesis project.

CFL condition

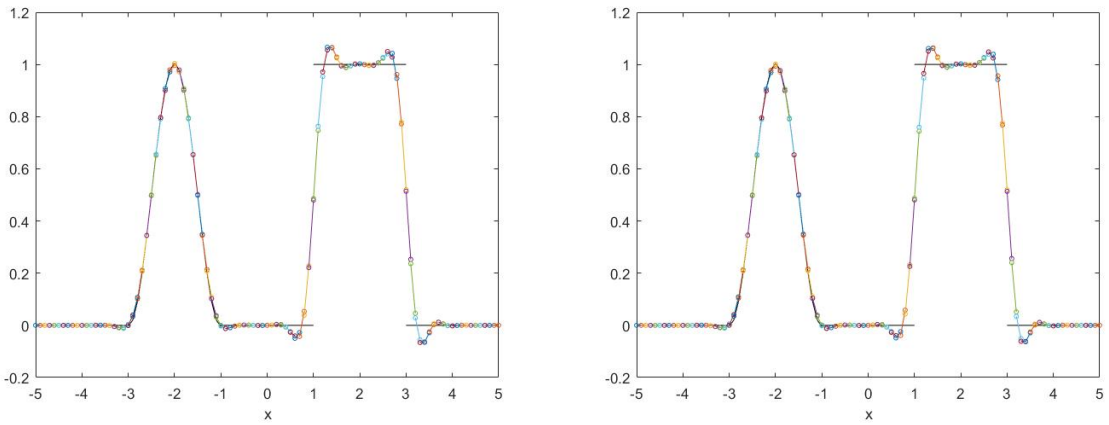
For the Runge-Kutta discontinuous Galerkin (RKDG) the CFL condition is not restricted by the common 1-limit, but the condition has to ensure L^2-stability:

$$|u|\frac{\Delta t}{\Delta x} \leq \text{CFL}_{L^2}.$$

In [5], Cockburn and Shu give the CFL_{L^2} conditions for different order Runge-Kutta integration methods and polynomial orders. These CFL_{L^2} numbers are given in Table 4.1.

Error calculation

To see how fast the method convergences, the errors are shown for different grid sizes. The error is estimated by measuring the difference between the exact solution $\varphi(x,t) = \varphi_0(x-ut)$ and its finite element approximation $\varphi_h(x,t)$. In the error calculation, instead of the exact solution, the L_2-projected initial condition is compared to the numerical solution after exactly one period which should exactly be the initial condition (recall (3.5)). Because the L_2-projection is advected, which is the finite element



(a) Nodal solution (b) Modal solution after transforming to physical space

Figure 4.3: Nodal and Modal solution with $N = 1$, $\Delta x = 0.1$ and $\Delta t = 10^{-4}$ at $t = 10$.

p	0	1	2	3	4	5	6	7	8
$v = 1$	1.000	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$
$v = 2$	1.000	0.333	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$
$v = 3$	1.256	0.409	0.209	0.130	0.089	0.066	0.051	0.040	0.033
$v = 4$	1.392	0.464	0.235	0.145	0.100	0.073	0.056	0.045	0.037
$v = 5$	1.608	0.534	0.271	0.167	0.115	0.085	0.065	0.052	0.042

Table 4.1: CFL_{L^2} for RKDG order v and polynomial order p, where $\star$ is unstable. Table taken from [5].

version of the initial condition and not the initial condition, moreover, comparing the L_2-projection simplifies the error calculation.

A very simple way to obtain the error is by using vector norms. The vector of the L_2-projection is given by $\boldsymbol{a}_h(0)$ and the approximation by $\boldsymbol{a}_h(10c)$ for $c \in \mathbb{N}$. In fact the difference between $\varphi_0(x)$ and $\varphi_h(x, 10c)$ for every grid point x is calculated with different vector norms. The vector norms that are used are the ℓ_1-norm, ℓ_2-norm (Euclidean norm) and the ℓ_∞-norm (infinity norm):

$$\left\|\boldsymbol{a}_h(0) - \boldsymbol{a}_h(10c)\right\|_1 = \sum_i \left|a_h(0)_i - a_h(10c)_i\right|, \tag{4.23a}$$

$$\left\|\boldsymbol{a}_h(0) - \boldsymbol{a}_h(10c)\right\|_2 = \sqrt{\sum_i \left(a_h(0)_i - a_h(10c)_i\right)^2}, \tag{4.23b}$$

$$\left\|\boldsymbol{a}_h(0) - \boldsymbol{a}_h(10c)\right\|_\infty = \max_i \left|a_h(0)_i - a_h(10c)_i\right|. \tag{4.23c}$$

Since two functions are compared, a function norm should, however, be used. For this reason the difference between the two functions is also calculated with the L_2-norm:

$$\|\varphi - \varphi_h\|_2 = \sqrt{\int_\Omega |\varphi - \varphi_h|^2 \, d\Omega}. \tag{4.24}$$

With the basis function representation of the L_2-projected initial condition and the solution exactly c

periods later, the L_2-norm can be written as:

$$\left\|\varphi_h(x,0)-\varphi_h(x,10c)\right\|_2 = \sqrt{\int_\Omega |\varphi_h(x,0)-\varphi_h(x,10c)|^2\, d\Omega} \tag{4.25a}$$

$$= \sqrt{\sum_{k=1}^{K}\int_{I_k}\left(\sum_{j=0}^{N} a_h^k(x_j^k,0)\ell_j^k(x)-\sum_{j=0}^{N} a_h^k(x_j^k,10c)\ell_j^k(x)\right)^2 dx} \tag{4.25b}$$

$$= \sqrt{\sum_{k=1}^{K}\int_{I_k}\left(\sum_{j=0}^{N} [a_h^k(x_j^k,0)-a_h^k(x_j^k,10c)]\ell_j^k(x)\right)^2 dx}. \tag{4.25c}$$

For simplification, a new vector $\boldsymbol{\alpha}$ is defined which has $\alpha_j^k = a_h^k(x_j^k,0)-a_h^k(x_j^k,10c)$ as elements.

$$\|\varphi-\varphi_h\|_2 = \sqrt{\sum_{k=1}^{K}\left(\sum_{i=0}^{N}\sum_{j=0}^{N}\alpha_i^k\alpha_j^k\int_{I_k}\ell_i^k(x)\ell_j^k(x)\,dx\right)} \tag{4.26a}$$

$$= \sqrt{\sum_{k=1}^{K}\sum_{i=0}^{N}\alpha_i^k\left(\int_{I_k}\ell_i^k(x)\ell_0^k(x)\,dx \quad \dots \quad \int_{I_k}\ell_i^k(x)\ell_N^k(x)\,dx\right)\begin{pmatrix}\alpha_0^k\\ \vdots\\ \alpha_N^k\end{pmatrix}} \tag{4.26b}$$

$$= \sqrt{\sum_{k=1}^{K}(\boldsymbol{\alpha}^k)^T M^k \boldsymbol{\alpha}^k} \tag{4.26c}$$

$$= \sqrt{\boldsymbol{\alpha}^T M \boldsymbol{\alpha}}. \tag{4.26d}$$

4.5.1. Numerical Solutions

In this section, three different DG methods are shown by changing the polynomial order of the basis functions. First, only linear basis functions ($N=1$) are used. Thereafter, the results of DG using quadratic basis functions ($N=2$) and basis functions of order $N=4$ are described. These polynomial orders are chosen, because of their theoretical convergence of $N+1$. Since the time integration method has only order 3, using basis functions higher than 2 would theoretically not add any advantages, because the error of the time integration would play a bigger role. However, the WENO method is of order 5, therefore, using basis functions of order 4 would give a fair comparison.

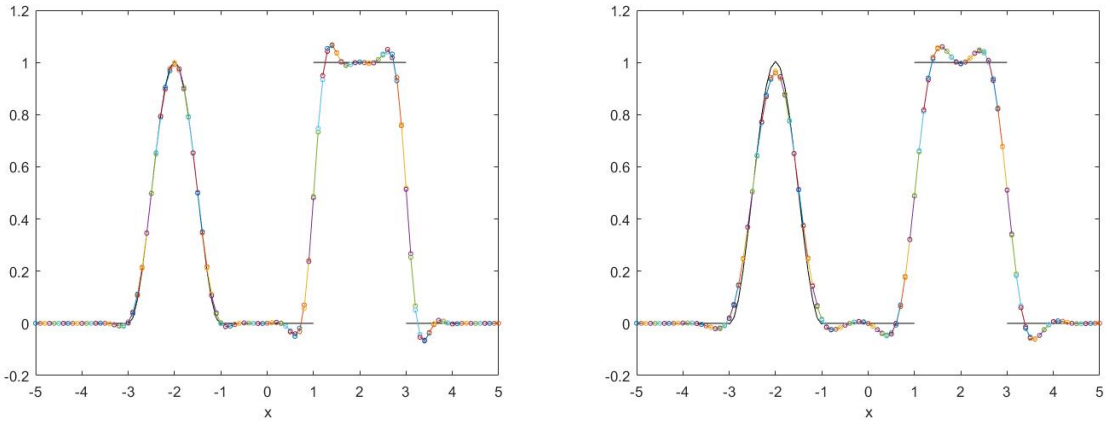
In Figure 4.4a and Figure 4.4b, the numerical solutions at $t=10$ and $t=50$ are given for DG with linear basis functions. It shows that for $N=1$, the results are dispersive and also a bit diffusive, but significantly less than the first order upwind. A more positive feature is the absence of time lag which are present in the finite difference methods.

The numerical solutions of DG with $N=2$ at $t=10$ and $t=50$ are shown in Figure 4.5a and Figure 4.5b respectively. For $N=2$, DG is less dispersive and diffusive as DG with $N=2$. Hence there are significantly more grid points than for $N=1$, since every element has an extra point in the middle of the element. Therefore, Figure 4.6 is given to show the results when all the grid points are the same as for $N=1$. Still for $N=2$, the diffusion is less.

For $N=4$, the solutions are shown in Figure 4.7. First, it must be noted that for $N=4$ only inexact integration with LGL quadrature is used. However, there is no sign of diffusion. On top of that, only a small part of the domain, around the discontinuities, $\varphi(x,10)$ is approximated incorrectly due to the oscillations. An extra result is plotted with the same grid points as the combinations $N=1$, $K=100$ and $N=2$, $K=50$ in Figure 4.8 to compare those three solutions in an other way. For $N=4$, the

diffusion is less than for $N = 1$ and $N = 2$, nevertheless, the dispersion is more. An other interesting note, is the similarity between the fifth order upwind at $t = 10$ and DG with $N = 4$ at $t = 50$, obviously without the time lag.

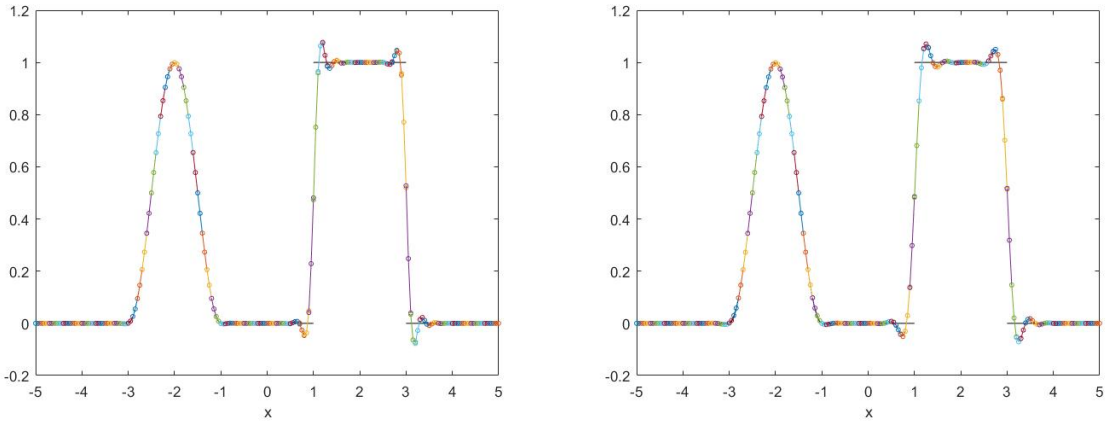
For all three different polynomial orders, DG seems to work well, obviously for higher N the accuracy is higher. Especially DG is very good in smooth regions. It is as diffusive as the fifth order upwind and the WENO method and is also conservative. Moreover, there is no time lag as the finite difference methods that are implemented in DALES. The only disadvantage of DG is the dispersion error around the discontinuity.



(a) DG with $N = 1$ at $t = 10$.

(b) DG with $N = 1$ at $t = 50$.

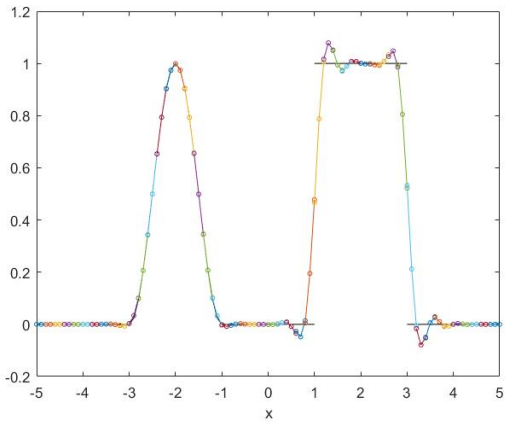
Figure 4.4: DG with $N = 1$, $\Delta x = 0.1 \Leftrightarrow K = 100$ and $\Delta t = 0.95\text{CFL}_{L^2} \frac{\Delta x}{|u|}$.



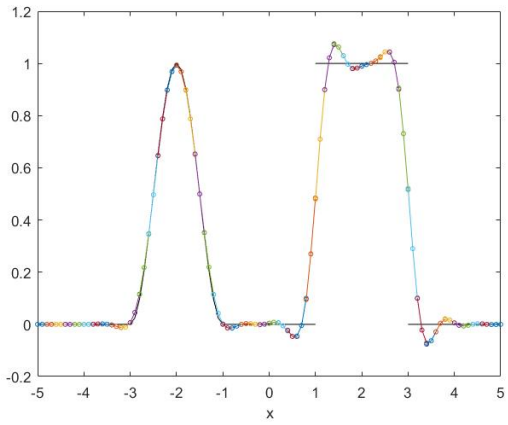
(a) DG with $N = 2$ at $t = 10$.

(b) DG with $N = 2$ at $t = 50$.

Figure 4.5: DG with $N = 2$, $\Delta x = 0.1 \Leftrightarrow K = 100$ and $\Delta t = 0.95\text{CFL}_{L^2} \frac{\Delta x}{|u|}$.

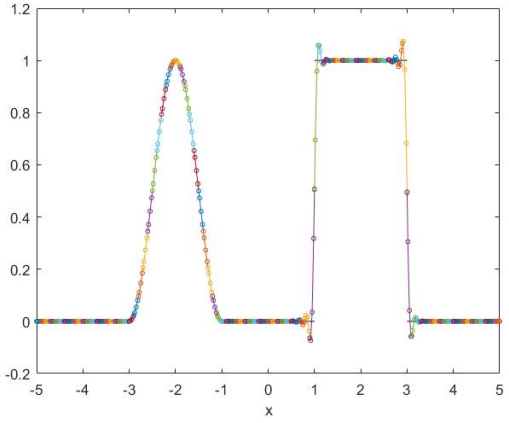


(a) DG with $N = 2$ at $t = 10$.

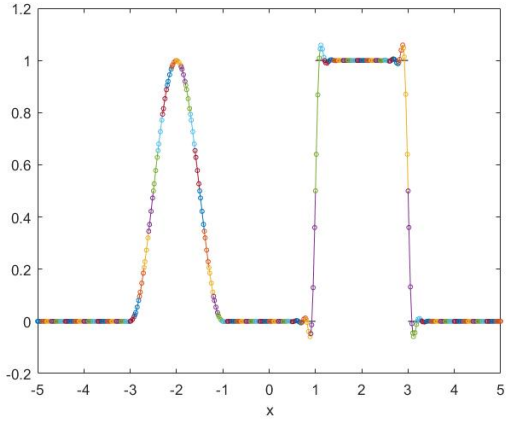


(b) DG with $N = 2$ at $t = 50$.

Figure 4.6: DG with $N = 2$, $\Delta x = 0.2 \Leftrightarrow K = 50$ and $\Delta t = 0.95\text{CFL}_{L^2} \frac{\Delta x}{|u|}$.

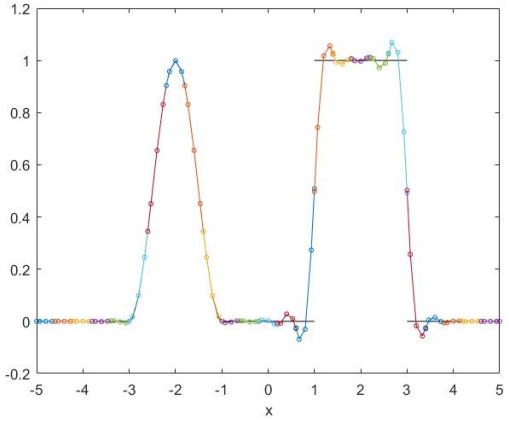


(a) DG with $N = 4$ at $t = 10$.

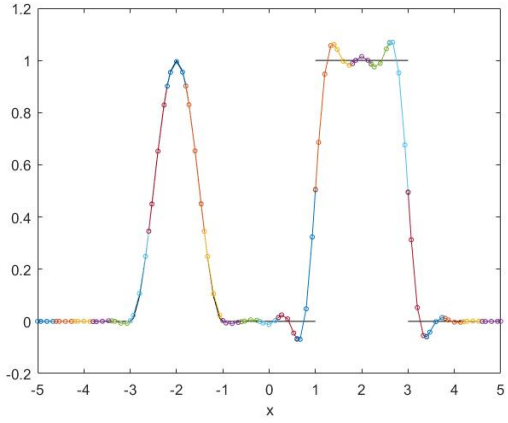


(b) DG with $N = 4$ at $t = 50$.

Figure 4.7: DG with $N = 4$, $\Delta x = 0.1 \Leftrightarrow K = 100$ and $\Delta t = 0.95\text{CFL}_{L^2} \frac{\Delta x}{|u|}$.



(a) DG with $N = 4$ at $t = 10$.



(b) DG with $N = 4$ at $t = 50$.

Figure 4.8: DG with $N = 4$, $\Delta x = 0.4 \Leftrightarrow K = 25$ and $\Delta t = 0.95\text{CFL}_{L^2} \frac{\Delta x}{|u|}$.

4.5.2. Computational time

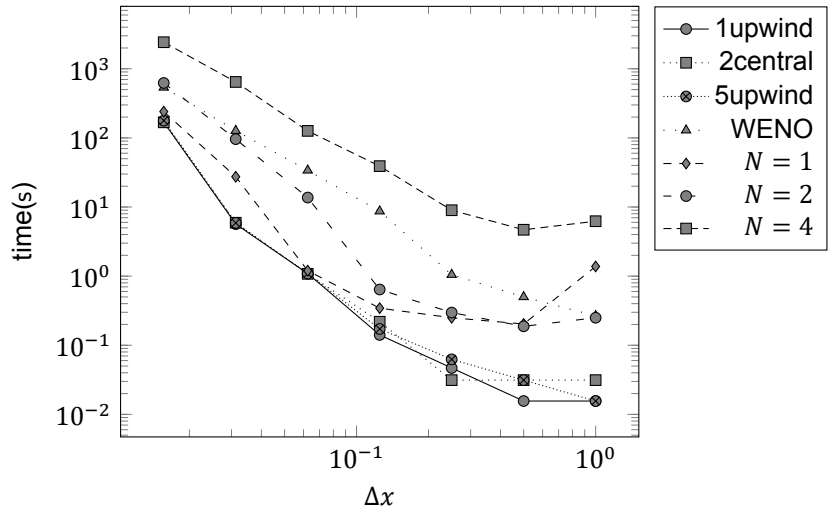


Figure 4.9: Computational time of the advection schemes in DALES and the tested DG to simulate till $t = 50$ s.

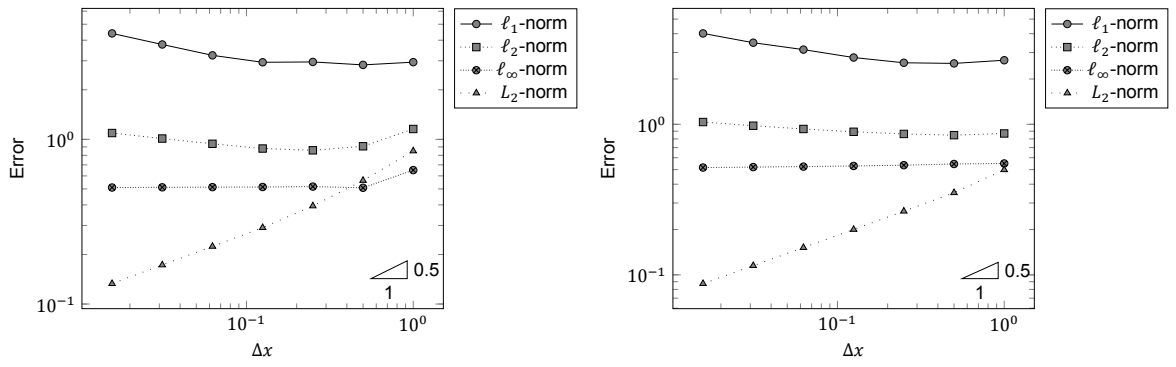
The computation time of DG is 10 times longer than WENO for $N > 2$. This is due to more grid points where the solution is approximated; three grid points per element extra are needed. However, we suspect that DG could be faster when parallelization is used.

4.5.3. Convergence

The convergence of DG for Cartesian grids is $\mathcal{O}(\Delta x^{N+1})$, which has been proven by LeSaint and Raviart [15]. In this subsection, the numerical methods are tested to see whether they converge with order $N+1$. The advection equation depends on time and space, therefore, both time and spatial discretization are tested on convergence at the same time. Since the time integration method is theoretical of order 3, DG with $N = 4$ is tested without the coupling to time.

Coupled tests

Figure 4.10a and Figure 4.10b show that the vector errors do not converge for both $N = 1$ and $N = 2$. The L_2-norm for $N = 1$ and $N = 2$ does converge with $\frac{1}{2}$, while it should be 2 and 3. The reason for this could be the presence of oscillations around the discontinuities. Therefore, the tests are also done without advecting the discontinuous part of φ_0.



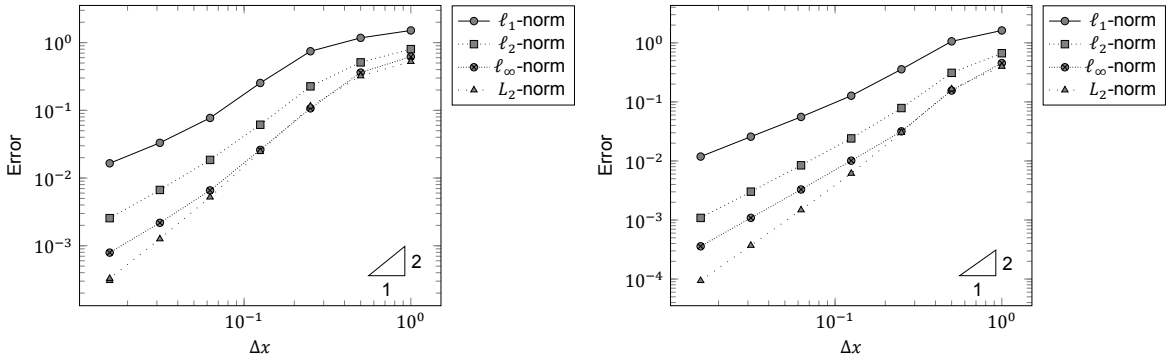
(a) $N = 1$ (b) $N = 2$

Figure 4.10: Numerical error of advecting φ_0 till t =10 s with $\Delta t = 0.95\text{CFL}_{L_2} \frac{u}{\Delta x}$.

The initial condition with only the smooth part of φ_0 is given by:

$$\breve{\varphi}_0(x) = \begin{cases} \frac{1-\cos(\pi(x-1))}{2}, & x \in [-3,-1] \\ 0, & \text{otherwise} \end{cases} \tag{4.27}$$

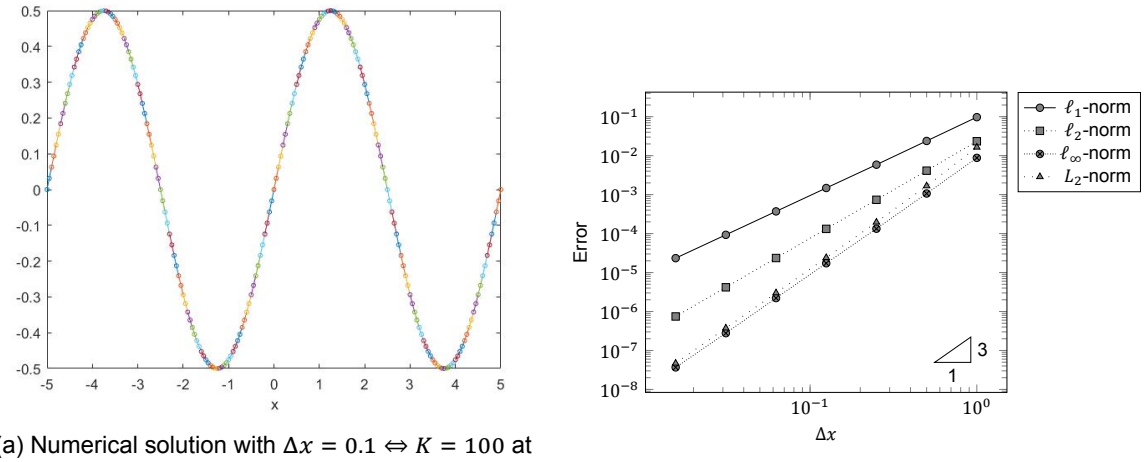
In Figure 4.11b and Figure 4.11a the errors of advecting this function for several grid sizes are given. In Figure 4.11b we see that DG with $N = 1$ can indeed converge with order 2, however with $N = 2$ a convergence of order 3 has not been achieved. This means that this function is not smooth enough which can for example be seen in the dispersion around the smooth part of φ_0 in Figure 4.6. Moreover, the vector norms converge which was not the case when φ_0 was advected.



(a) $N = 1$ (b) $N = 2$

Figure 4.11: Numerical error of advecting the smooth part of φ_0 till t =10 s with $\Delta t = 0.95\text{CFL}_{L_2}\frac{u}{\Delta x}$.

A very smooth and period function is a cosine or sine. Therefore, a sinus function with amplitude 0.5 and a period of $\frac{5}{2\pi}$ is used. In Figure 4.12a a numerical result of DG with $N = 2$ is shown of the sinus function which show no limitations. Moreover, a convergence of order 3 is achieved, which confirms the order of convergence to be $N + 1$ for $N \leq 2$ for very smooth functions.



(a) Numerical solution with $\Delta x = 0.1 \Leftrightarrow K = 100$ at $t = 10$. (b) Numerical error.

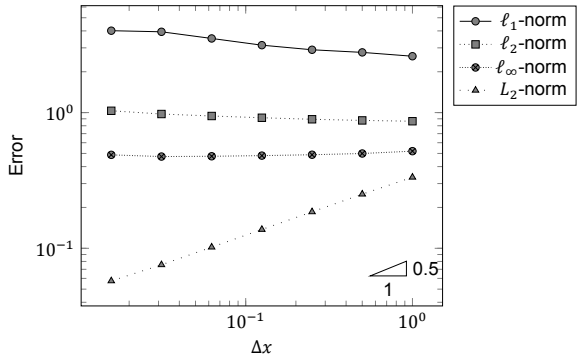
Figure 4.12: Advecting $0.5\sin(\frac{2\pi}{5}x)$ till t =10 s with $\Delta t = 0.95\text{CFL}_{L_2}\frac{u}{\Delta x}$.

Decoupled tests

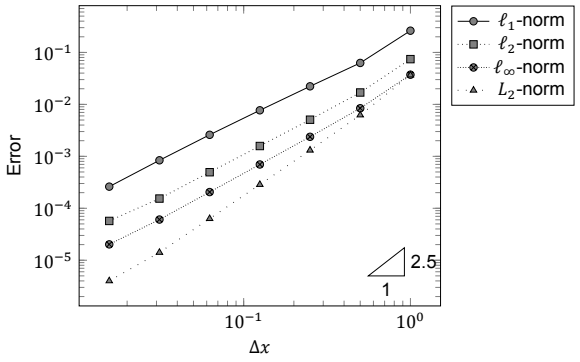
By taking Δt constant and small enough to keep the method stable, only the discretization errors can influence the results. These decoupled tests are done for $N = 4$ with different initial conditions. For

Figure 4.13a, the initial condition φ_0 is advected. The numerical error calculated with the L_2-norm converges with order 0.5, nevertheless, the vector norms do not converge. This was also the case for $N = 1$ and $N = 2$ (see Figure 4.10). The convergence of DG with $N = 4$ achieves order 2.5 when only the smooth part of φ_0 is advected, which can be seen in Figure 4.13b. In Figure 4.13c, the numerical error is given when a smoother function as a sinus function is used. It shows that for $N = 4$ the method converges with order 5 till it does not converge anymore and for the ℓ_1- and ℓ_2- errors even rise. We suspect that the time error starts playing a bigger role for the smaller grid sizes, which can be resolved by using a smaller Δt.

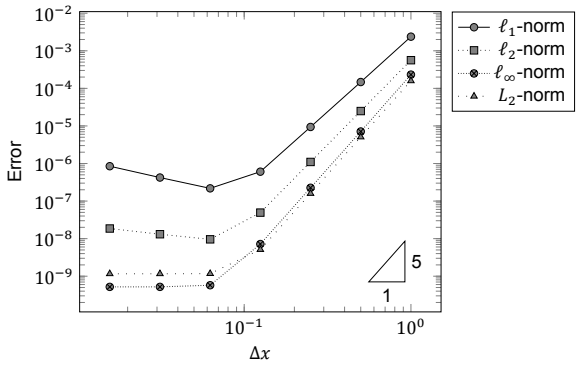
All in all, the DG method with basis functions of polynomial order N can indeed converge with order $N + 1$ for smooth functions.



(a) Advecting φ_0 till t =10 s.



(b) Advecting the smooth part of φ_0 till t =10 s.



(c) Advecting $0.5\sin(\frac{2\pi}{5}x)$ till t =10 s.

Figure 4.13: Numerical errors of DG with $N = 4$ using $\Delta t = 0.001$.

4.6. Moment Limiter

A disadvantage of the DG method is the presence of non-physical oscillations in the results, therefore, a limiter is needed. However, with most limiters the solution is reduced to first-order accuracy and the advantage of high-order methods is lost.

The limiters that are made for DG are defined for the modal form of the DG, thus, the following steps have to be taken:

1. Transform from nodal to modal representation,
2. Apply limiter,
3. Transform back from modal to nodal representation.

These steps can be done on element basis[18], meaning that there is no global assembly operation needed which can save computational time.

In this literature study the moment limiter is used. The choice for the moment limiter is the simple implementation and the underlying idea. The moment limiter was first developed by Biswas et al. and is a generalization to higher order of the second-order minmod limiter of van Leer [3]. Krivodonova generalized the limiter and extended it to two-dimensional problems on tensor-product meshes [13].

The moment limiter gradually reduces the order if it is needed. It limits the derivative of order j in a given cell using the derivatives of order $j-1$ in the neighbouring cells. The limiter starts by limiting, when needed, the highest orders first. Then the limiting process continues until it is not needed to limit any more or until all terms are limited. With this strategy the solution has the highest order accuracy possible when limiting is needed.

The moment limiter uses the minmod function which is defined as:

$$m(a_1, \dots, a_N) = \begin{cases} s \min_{1 \le j \le N} |a_j|, & \text{if } \operatorname{sgn}(a_1) = \dots = \operatorname{sgn}(a_N) = s, \\ 0, & \text{otherwise.} \end{cases} \tag{4.28}$$

The moment limiter also uses the modal form of the DG solution $\varphi_h^k(x,t) = \sum_{j=0}^{N} \hat{a}_j^k(t)\psi_j(x)$. The pseudoalgorithm of the Moment limiter can be found in Algorithm 1.

Algorithm 1 Moment Limiter

for all elements I_k **do**
  Set $j = N$
  while $j = N$ **or** ($\tilde{a}_j^k \neq \hat{a}_j^k$ **and** $j > 1$) **do**
    $\beta_j = \frac{\sqrt{j-1/2}}{\sqrt{j+1/2}}$
    $\tilde{a}_j^k = m\left(\hat{a}_j^k, \beta_j(\hat{a}_{j-1}^{k+1} - \hat{a}_{j-1}^k), \beta_j(\hat{a}_{j-1}^k - \hat{a}_{j-1}^{k-1})\right)$
    $j = j - 1$
  end while
end for

The idea behind this algorithm is that roughly speaking the $\tilde{a}_j^k$ corresponds to the jth derivative of the solution of element k. Thus, this coefficient is compared with the numerical derivative using forward and backward differences.

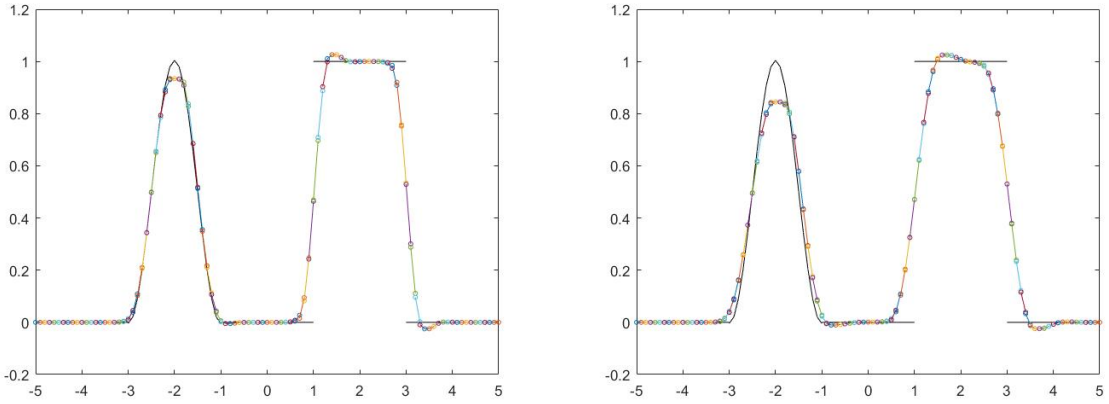
4.6.1. Numerical Results

In this subsection, the numerical results are given for the moment limited DG methods with basis functions of polynomial order $N = 1$, $N = 2$ and $N = 4$ using coupled tests. Moreover, the convergence and computational time will be computed.

For $N = 1$, only one coefficient can be limited, meaning that when limiting is needed, the second order derivatives are set to zero. In Figure 4.14, the numerical results are given at $t = 10$ and $t = 50$. The peak is clipped and diffusion is added. Moreover, the extrema of the dispersion is smaller, but not completely removed. In time, we see that the diffusion has a significant impact. On top of that, the smooth part of the results lean to the right, the direction of the speed and towards the discontinuous part. The cause of this could be the dispersion that comes from the discontinuous part of φ_0.

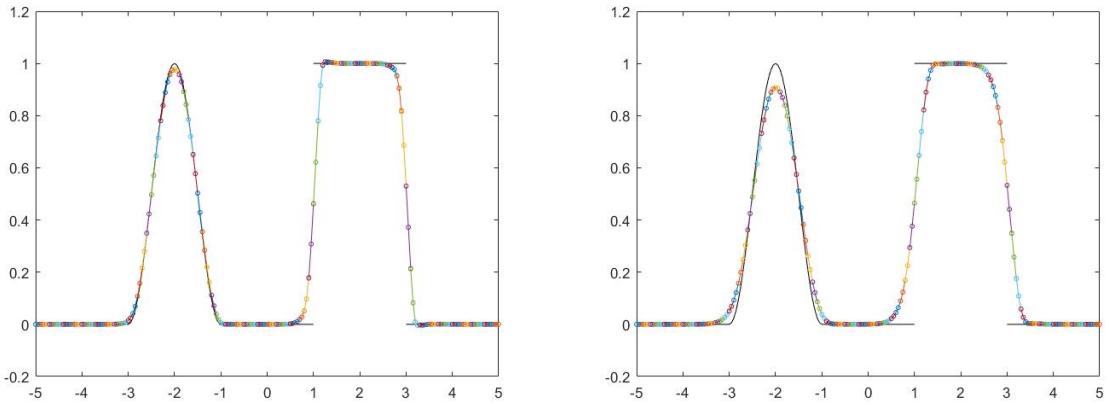
For a one order higher method, DG with $N = 2$, the effect of the moment limiter is significantly better than for $N = 1$ (see 4.15). The remains of the dispersion that were still present for $N = 1$, are almost removed at $t = 10$ and for $t = 50$ completely gone. Moreover, the amount of diffusion is similar as the amount for $N = 1$, only there is no peak clipping.

In Figure 4.16, the numerical results of moment limited DG with $N = 4$ are given. The remains of the dispersive errors which were still present for $N = 1$, $N = 2$ at $t = 10$, are absent for $N = 4$. An other striking result, is the similarity between the numerical solutions at $t = 50$ for both $N = 2$ and $N = 4$. It seems as if the moment limiter considers this function as smooth when $\Delta x = 0.1$. The solution for $t = 10$ is better approximated when $N = 4$ than for $N = 2$.



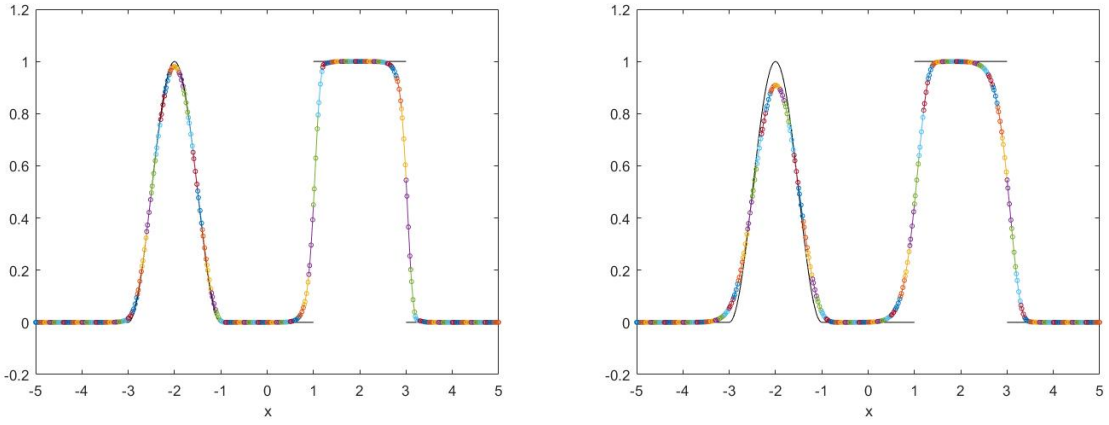
(a) at $t = 10$ (b) at $t = 50$

Figure 4.14: Moment limited DG with $N = 1$, $\Delta x = 0.1$ and $\Delta t = 0.95\text{CFL}_2 \frac{u}{\Delta x}$



(a) at $t = 10$ (b) at $t = 50$

Figure 4.15: Moment limited DG with $N = 2$,$\Delta x = 0.1$ and $\Delta t = 0.95\text{CFL}_2 \frac{u}{\Delta x}$

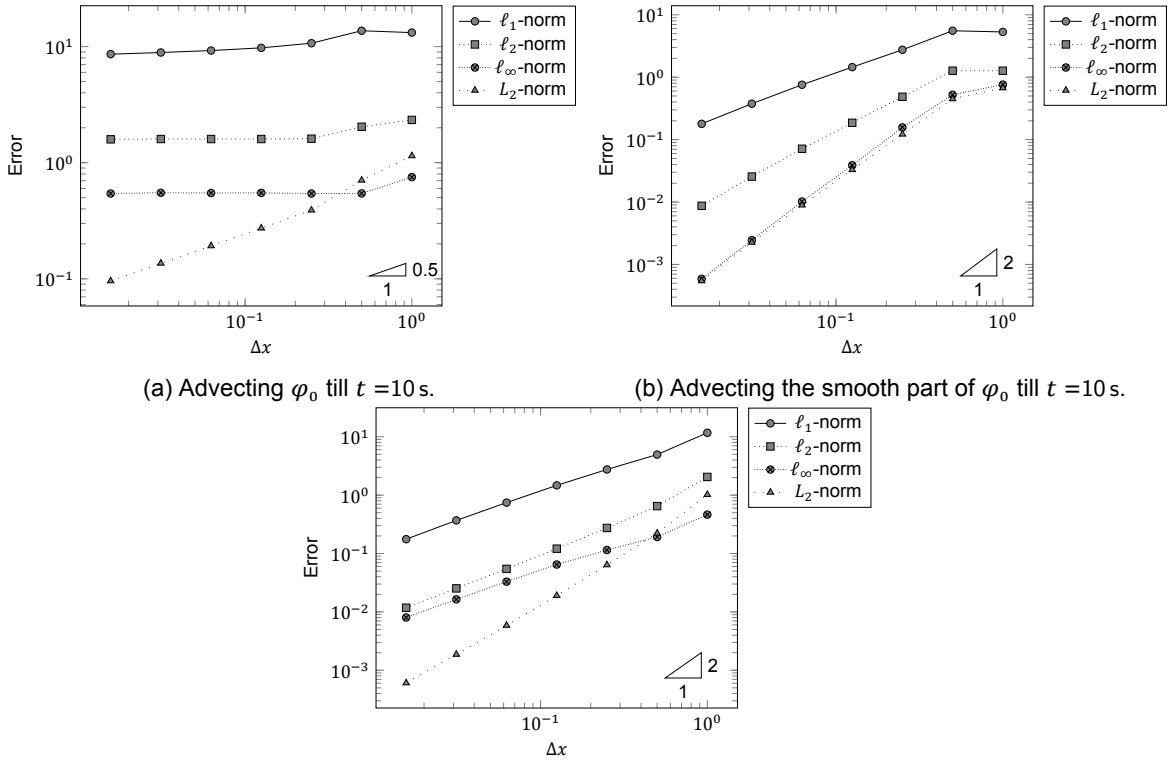


(a) at $t = 10$ (b) at $t = 50$

Figure 4.16: Moment limited DG with $N = 4$, $\Delta x = 0.1$ and $\Delta t = 0.95\text{CFL}_2 \frac{u}{\Delta x}$

Convergence

In this section, the convergence of the moment limited DG is tested with $N = 4$. The convergence of the not limited DG with $N = 4$ is of order 5 for very smooth functions. In general, a limiter reduces the order of the method. This can also be concluded from the convergence plots in Figure 4.17. Even for smooth functions the convergence of order 5 is not obtained. However, if we compare this with the convergence of the WENO method which can be seen in Figure 4.18, it can be seen that the convergence of the methods are the same. Nevertheless, the numerical error of the limited DG with $N = 4$ is less in all cases, for φ_0, the smooth part of φ_0 and the sine. This is due to the time lag the WENO method has. Thus even though the shape is better maintained with WENO, the numerical error of limited DG is less.

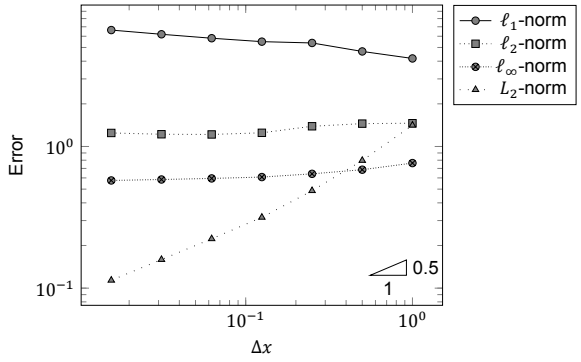


(a) Advecting φ_0 till t =10 s.

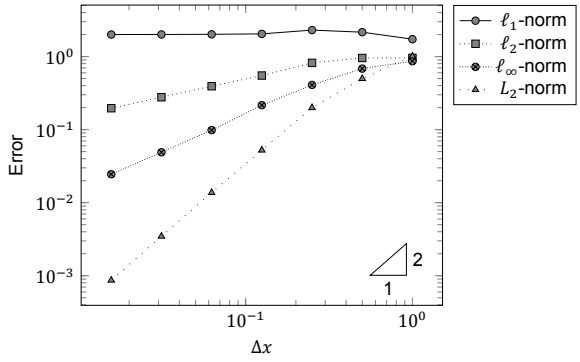
(b) Advecting the smooth part of φ_0 till t =10 s.

(c) Advecting $0.5\sin(\frac{2\pi}{5}x)$ till t =10 s.

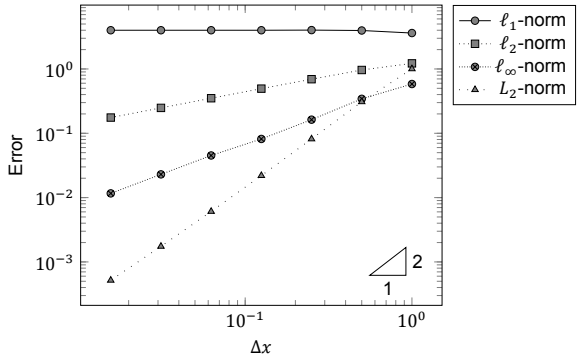
Figure 4.17: Numerical errors of the moment limited DG with $N = 4$ using $\Delta t = 0.001$.



(a) Advecting φ_0 till t =10 s.



(b) Advecting the smooth part of φ_0 till t =10 s.



(c) Advecting $0.5\sin(\frac{2\pi}{5}x)$ till t =10 s.

Figure 4.18: Numerical errors of the WENO method using $\Delta t = 0.001$.

Computation time

In Figure 4.19, the computation time of DG, limited DG and the WENO method is given. For $N = 2$ and $N = 4$ the computational time with the addition of a limiter is twice as long than without limiter. For $N = 1$ the computational time becomes 5 times as long with than without limiter. Compared to WENO, the limited DG with $N = 4$ is twice as slow. However, parallelization is not used for the computation of the method of DG and the limiter and that could speed up the computation.

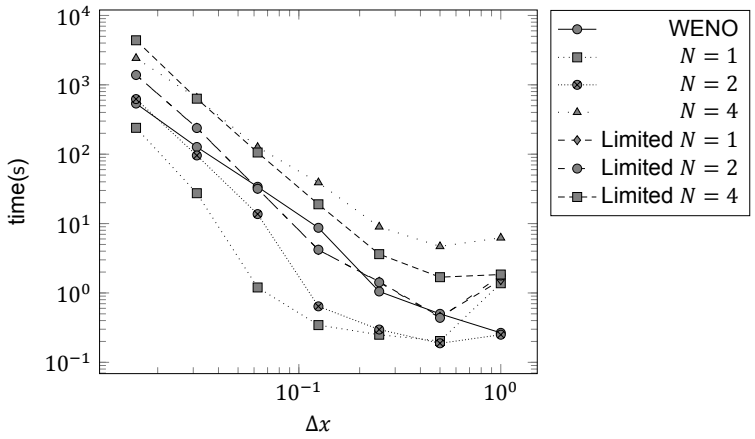


Figure 4.19: Computational time of the advection schemes in DALES and the tested DG to simulate till t = 50 s.

4.7. Concluding Remarks

All in all, the discontinuous Galerkin method works good for smooth functions. Namely, the truncation error of the method is $\mathcal{O}\left(\Delta x^{N+1}\right)$ where N is the polynomial order of the basis functions. Moreover,

there is no time lag present which was present when the WENO method was used. The absence of the time lag is very important to the KNMI, especially for the long-time predictions. However, at discontinuous parts non-physical oscillations take place. Therefore, a limiter is needed.

The limiter that is tested in this chapter is the moment limiter. This limiter removes the oscillations by adding diffusivity. Unfortunately, this makes the method too diffusive and it also reduces the method to a lower order. Nonetheless, the convergence is the same as the WENO method and on top of that, the numerical error is smaller than of the WENO, due to the absence of a time lag.

5

Open problems & Issues

In this chapter the open problems and issues will be examined. In the past chapters, one dimensional advection equations are discussed, nevertheless, DALES is a three dimensional model. This has several consequences for the DG method. Moreover, other options for limiters and ideas for test cases are given.

5.1. Higher Dimensions & Polynomial Basis Functions

5.1.1. Basis Functions

As told before, DALES considers a three dimensional problem. This means that the DG method needs three dimensional basis functions. Since the grid cells in DALES are cuboids, a tensor product of the one-dimensional basis functions is a simple option. In other words, a tensor product of the Lagrangian functions which are based on LGL grid points should be used. When for example a triangulated grid was used, multivariate lagrangian polynomials would be an option [20]. For the modal form, which is needed for the limiter, a tensor product of the scaled Legendre polynomials could be a good choice.

5.1.2. Staggered Grids

As told in Chapter 3, DALES uses an Arakawa C-grid for the grid discretization. This must not be changed when DG is implemented. Therefore, the use of staggered grids with DG needs to be reflected on.

Recall that the weak formulation is written as:

$$\int_{I_k} \frac{\partial \varphi}{\partial t}\eta - f(\varphi)\frac{\partial \eta}{\partial x}\, dx + \int_{\partial I_k} f(\varphi_h)\eta \cdot \boldsymbol{n}\, d\Gamma = 0, \tag{5.1}$$

where the flux function is given by $f(\varphi) = \boldsymbol{u}\varphi$. In other words, the velocity $\boldsymbol{u}$ is integrated over the grid box and the faces of the grid cell. Especially when $\boldsymbol{u}$ is not constant, the location of the approximated $\boldsymbol{u}$ is very important.

For every N, many options are available. One-dimensional drawings are used to illustrate several options which can be extended to higher dimensions by using a tensor product.

In Figure 5.1, two possibilities for linear basis functions are displayed. The Arakawa C-grid for one-dimension is drawn in black and the DG option in red. The left figure is the most straightforward choice: take the whole grid cell as element. Then φ_k is calculated by interpolating φ_1 and φ_2 or their average. On the right side, the grid cell is split into two elements. This means that no interpolation is needed to acquire φ_k but it can be taken as φ_{12} or φ_{21}. Instead the velocity at x_k must be approximated by interpolating u_k and u_{k+1}.

For $N = 2$, two options are illustrated in Figure 5.2. On the left side, the element is exactly one grid cell. For this method, the velocity has to be approximated at $x = x_k$, but no interpolation is needed

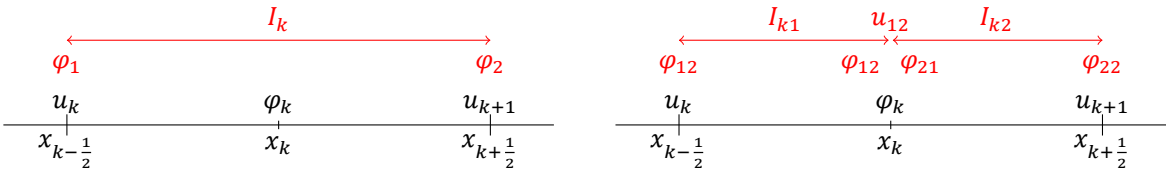


Figure 5.1: Staggered grid options for $N = 1$.

to find φ_k. On the right side, a less straightforward suggestion is given which shows an element that covers two grid cells. The advantage of this is that computational time is shorter, because there are less extra values of φ approximated that are not needed (like φ_1 and φ_2 in the left side of the figure). However, interpolation is needed to get φ_k and φ_{k+1} leading to even more inaccuracy than it has by having bigger elements. For higher polynomial degrees $N > 2$, about the same options can be used. By taking even polynomial degrees φ_k is always calculated and no extra interpolation is needed than when taking uneven polynomial orders.

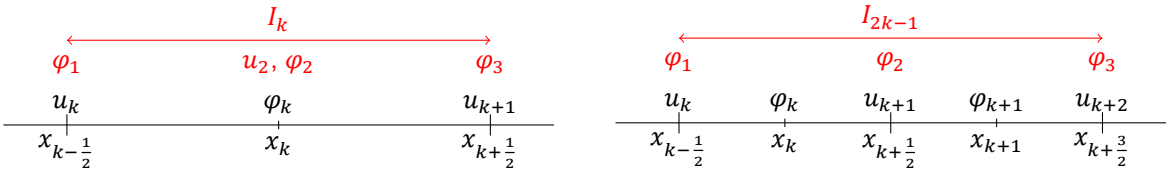


Figure 5.2: Staggered grid options for $N = 2$.

In my thesis project, the whole grid cell will be taken as an element. This gives a higher accuracy than taking more grid cells as one element and a shorter computational time than when a grid cell is split into multiple elements.

5.1.3. Initial and Boundary Conditions

As shown in the left figure of Figure 5.2, the value φ of multiple grid points are calculated, meaning that these has also to be given by the initial condition and also for the boundary condition. When the initial and/or boundary condition is a function, there is no problem. However, the initial and/or boundary condition is a vector of values, so an extra interpolation is needed.

5.1.4. Higher Dimensional Moment Limiter

The moment limiter has to be extended to three dimensions. In [13], the derivations of the limiter for one-dimensional and two-dimensional problems are thoroughly explained, therefore, the extension to three-dimensional problems should be doable.

5.2. Other Limiters

In this section two other limiters than the moment limiter are given that can be used for advection equations.

5.2.1. Algebraic Flux Correction

An algebraic approach to find a high-resolution scheme for scalar conservation laws is algebraic flux correction (AFC) by Kuzmin [14]. AFC is a generalization of the flux-corrected transport (FCT) methodology. AFC creates from a given standard Galerkin discretization, a positivity-preserving implicit Galerkin scheme and removes excessive artificial diffusion in sufficiently smooth areas. This is done by decomposing the antidiffusive part of the discrete operator into numerical fluxes and limit those fluxes in a conservative way.

5.2.2. Shock Detection using Multiwavelets

Another limiter that could be used is a troubled cell indicator by using multiwavelets [25]. Vuik used a multiwavelet formulation to decompose the modal DG approximation into a sum of global averages and finer details on different levels. Moreover, she proved that there is an exact relation between the multiwavelet coefficients of the highest decomposition level and jumps in the DG approximation. With this troubled cell indicator, only troubled cells can be limited instead that also local extrema are limited.

5.3. Test Cases

When DG is implemented in DALES, it must be tested and compared with the other implemented advection equations. This will be done with several test cases. First, a simple advection equation will be tested with periodic boundaries and flux function $f(\varphi) = \boldsymbol{u}\varphi$ where $\boldsymbol{u}$ is constant. The initial condition would be a smooth function like a sinus. Thereafter, a discontinuous initial condition condition will be tested. This could be a cubic that is advected. Another test could be a temperature and/or moisture inversion as in Figure 5.3 is shown. With this test it becomes clear whether there is too much artificial diffusivity is added. If that is the case the eddies that arise, would not be shown in the simulation.

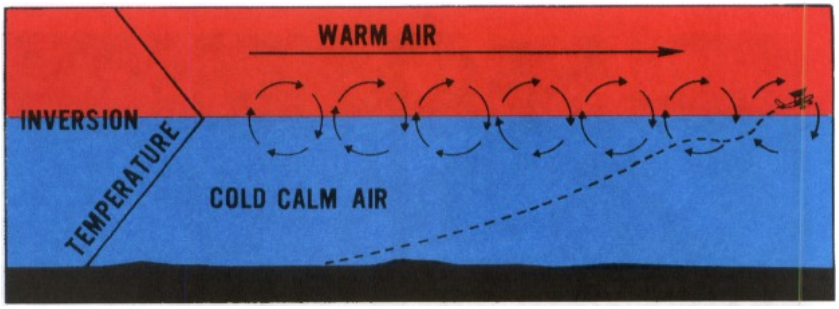


Figure 5.3: Temperature inversion. Image taken from [8].

6

Summary and Conclusion

In this chapter, the summary and conclusion of the literature study are given. At last, the literature study is concluded with the future work of the thesis project.

6.1. Summary

The problems of numerical weather prediction and climate models are mathematically modelling atmospheric processes and to evaluate the models as accurate and efficiently as possible. One of these atmospheric processes that could be solved more accurately and efficiently is the advection. Advection is one of the most important processes that takes place in the atmosphere. The Dutch Atmospheric Large-Eddy Simulation model (DALES) that is used, among other things, for parametrization development, still has problems with its advection schemes. Therefore, the discontinuous Galerkin (DG) method is suggested which is known for its high scalability, geometric flexibility and allowance of discontinuous approximations.

The atmosphere is a layer of air around the Earth that is kept in place by the Earth's gravity. Most weather takes place in the lowest part of the atmosphere, the troposphere. The origin of the processes that cause the different weather types is the movement of air in the atmosphere. For the movement of air and even in absence of air movement, three conservation laws must hold: conservation of motion, mass and (thermodynamic)energy. This is described by three equations: the Navier-Stokes equation, the continuity equation and the advection equation. In DALES, the filtered versions of these equations are used such that the computational cost is reduced by filtering out the smallest scales of the turbulence.

Moreover, various finite difference advection schemes of DALES have been tested for a simple one-dimensional advection problem. The low order advection schemes have a very short computation time, but do not have a high accuracy. The WENO method has a higher accuracy and no dispersion errors, whereas the fifth order upwind has dispersion errors. Not to mention, the finite difference schemes all have a time lag creating a problem for long-time predictions.

The DG method is a combination of the finite element method and the finite volume method. This method solves the weak form of the differential equations instead solving the differential equations like the finite difference method does. The idea of DG works well for smooth applications, DG shows no problems like time lags or diffusion. On top of that, DG superconverges with $N + 1$ where N is the polynomial order of the basis functions. However, at discontinuities there are non-physical oscillations in the approximations. These can be resolved by using a limiter. Like most limiters, the tested moment limiter removes the dispersion errors, but adds too much artificial diffusion which reduces the order of the method. Moreover, the limited DG method does not have a shorter computation time than the WENO method.

As DG will be implemented in DALES, the implementation of DALES should be taken into account.

This means that the one-dimensional DG that was tested in this literature study, has to be extended to the three-dimensional advection problem. The extension has consequences on the basis functions, initial and boundary conditions, and the limiters. Moreover, the Arakawa C-grid of DALES, which is a staggered grid, also has influence on DG. These open problems are reflected on in Chapter 5. Moreover, more limiters are given as other options for the moment limiter. At last, several test cases are shown that can be used to test the higher dimensional advection DG schemes.

6.2. Conclusion

All in all, the DG method shows that it is very promising. The method converges with order $N+1$ where N is the polynomial degree of the basis functions and even though the computation time is not shorter than the other implemented advection schemes of DALES, we have not yet taken the advantage of the high-scalability of DG. The disadvantage of DG is the dispersion errors at the discontinuities. To remove these oscillations, a limiter is needed. Unfortunately, the tested moment limiter reduces the order by adding too much artificial diffusion. This does not take away that the results of DG are much better because of the absence of the time lags.

6.3. Future Work

First, DG will be implemented for a two-dimensional Discontinuous Galerkin method in the horizontal direction x and vertical direction z (height). Thereafter, the three-dimensional DG method will be implemented in such way the Fortran subroutine can be used in the DALES model, but will be tested outside the DALES model. Last but not least, the moment limiter will be implemented and tested. If the project goes smoothly, the shock detection method of Vuik [25] can be tested.

Bibliography

[1] P. Bechtold. Atmospheric thermodynamics, May 2009. URL https://www.ecmwf.int/sites/default/files/elibrary/2015/16954-atmospheric-thermodynamics.pdf.

[2] L.C. Berselli, L.C. Iliescu, and W.J. Layton. *Mathematics of Large Eddy Simulation of Turbulent Flows*. Springer, Berlin, Heidelberg, 2006.

[3] R. Biswas, K. D. Devine, and J. E. Flaherty. Parallel, adaptive finite element methods for conservation laws. *Applied Numerical Mathematics*, 14(1-3):255–283, apr 1994. doi: 10.1016/0168-9274(94)90029-9.

[4] S.J. Böing. *The Interaction Between Deep Convective Clouds and Their Environment*. PhD thesis, TU Delft, 2014.

[5] B. Cockburn and C.-W. Shu. Runge-kutta discontinuous galerkin methods for convection-dominated problems. *Journal of Scientific Computing*, 16(3):173–261, 2001. doi: 10.1023/a:1012873910884.

[6] R. Desai. The cyclone [image]. Educational Blog, May 2011. URL http://drrajivdesaimd.com/2011/05/14/the-cyclone/comment-page-1/.

[7] A. Dosio. *Turbulent Dispersion in the Atmospheric Convective Boundary Layer*. PhD thesis, Wageningen Universiteit, May 2005.

[8] Federal Aviation Administration. *Aviation Weather [Image]*. Aviation Supplies & Academics, Inc., 1975. URL https://www.aviationweather.ws/046_Wind_Shear.php.

[9] A. Harten, B. Engquist, S. Osher, and S.R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, III. *Journal of Computational Physics*, 71(2):231–303, aug 1987. doi: 10.1016/0021-9991(87)90031-3.

[10] T. Heus, C. C. van Heerwaarden, H. J. J. Jonker, A. Pier Siebesma, S. Axelsen, K. van den Dries, O. Geoffroy, A. F. Moene, D. Pino, S. R. de Roode, and J. Vilà-Guerau de Arellano. Formulation of the dutch atmospheric large-eddy simulation (DALES) and overview of its applications. *Geoscientific Model Development*, 3(2):415–444, sep 2010. doi: 10.5194/gmd-3-415-2010.

[11] J. R. Holton and G. J. Hakim. *An Introduction to Dynamic Meteorology*. Academic Press. Elsevier, 2013.

[12] G.T. Huntington, D.A. Benson, and A.V. Rao. A comparison of accuracy and computational efficiency of three pseudospectral methods. *AIAA Paper 2007-6405*, 2007.

[13] L. Krivodonova. Limiters for high-order discontinuous galerkin methods. *Journal of Computational Physics*, 226(1):879–896, sep 2007. doi: 10.1016/j.jcp.2007.05.011.

[14] D. Kuzmin. Algebraic flux correction i. scalar conservation laws.

[15] P. Lesaint and P. A. Raviart. On a finite element method for solving the neutron transport equation. *Publications mathématiques et informatique de Rennes*, (S4):1–40, 1974.

[16] R.J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[17] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, nov 1994. doi: 10.1006/jcph.1994.1187.

[18] S. Marras, J. F. Kelly, M. Moragues, A. Müller, M. A. Kopera, M. Vázquez, F. X. Giraldo, G. Houzeaux, and O. Jorba. A review of element-based galerkin methods for numerical weather prediction: Finite elements, spectral elements, and discontinuous galerkin. *Archives of Computational Methods in Engineering*, 23(4):673–722, 2016. ISSN 1886-1784.

[19] Met Office. Clouds, 2012. URL http://www.metoffice.gov.uk/binaries/content/assets/mohippo/pdf/library/factsheets/12_0674-factsheet-1_clouds.pdf.

[20] K. Saniee. A simple expression for multivariate lagrange interpolation. 2007.

[21] C.-W. Shu and Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, aug 1988. doi: 10.1016/0021-9991(88)90177-5.

[22] C.-W. Shu and Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes, II. *Journal of Computational Physics*, 83(1):32–78, jul 1989. doi: 10.1016/0021-9991(89)90222-2.

[23] J. van der Dussen. *Stratocumulus Transitions in Present-day and Future Climate*. mathesis, TU Delft, June 2015.

[24] C. Vuik, P. van Beek, F. Vermolen, and J. van Kan. *Numerieke Methoden voor Differentiaalvergelijkingen*. VSSD, 2006.

[25] M.J. Vuik. *Multiwavelets and Outlier Detection for Troubled-cell indiation in Discontinuous Galerkin Methods*. PhD thesis, TU Delft, 2017.

[26] P. Wesseling. von neumann stability conditions for the convection-diffusion eqation. *IMA journal of Numerical Analysis*, 16(4):583–598, 1996.