ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Term Project

A Finite Volume Nonhydrostatic Atmospheric Model for the Simulation of Mountain Waves

Christoph Winkelmann

July 18, 2003

advised by

Prof. Christoph Schär and Dr. Jürg Schmidli
Institute for Atmosphere and Climate
Swiss Federal Institute of Technology
Zürich

Abstract

Two schemes for the two dimensional nonhydrostatic adiabatic Euler equations are formulated: a combined finite volume/finite difference (FV/FD) split explicit scheme and a pure finite volume (FV) scheme. The equations are solved in a rectangular domain with linear boundary condition for the topography and with periodic boundary conditions horizontally. In the FV/FD scheme, the small timestep is discretized with a vertically implicit forward backward finite difference scheme, while for the large timestep, finite volumes are used for spatial and a leap-frog scheme for temporal discretization. This discretization is also used for the FV scheme. The schemes are implemented in Fortran90 using object oriented design principles. Numerical experiments are performed to verify the schemes.

Contents

1 Introduction

With the increase of available computing power, operational weather forecast models can be run with resolutions that allow direct simulation of convection. This requires nonhydrostatic models that really simulate convection directly. It is however not clear, which set of equations fulfills the new requirements best. In this study we use the full, unapproximated Euler equations for a dry adiabatic flow. This set of equations includes the relevant physical phenomena for the mesoscale, but also the irrelevant one of sound waves. To overcome the restriction of the sound mode on the timestep, we use a split explicit scheme in combination with a finite volume spatial discretization. The latter has the advantage of allowing for a variety of grids like terrain-following grids or the shaved cell approach.

2 Formulation

2.1 Continuous Equations

The twodimensional prognostic (nonhydrostatic) adiabatic Euler equations in conservative form are:

$$\partial_t(\rho u) + \partial_x(\rho u^2 + p) + \partial_z(\rho uw) = 0 \tag{2.1}$$
$$\partial_t(\rho w) + \partial_x(\rho uw) + \partial_z(\rho w^2 + p) = -\rho g \tag{2.2}$$
$$\partial_t(\rho\theta) + \partial_x(\rho u\theta) + \partial_z(\rho w\theta) = 0 \tag{2.3}$$
$$\partial_t\rho + \partial_x(\rho u) + \partial_z(\rho w) = 0 \tag{2.4}$$

where ρ is the mass density, u is the horizontal wind speed, w is the vertical wind speed, p is the pressure, g is the gravity acceleration and θ is the potential temperature. Pressure is obtained from the diagnostic equation of state:

$$p = p_0 \left(\frac{R\rho\theta}{p_0}\right)^{\gamma} \tag{2.5}$$

where p_0 is the bottom pressure, R is the gas constant and $\gamma = c_p/c_v$ is the ratio of specific heats. We want to solve these equations numerically in a computational domain

$$\Omega = \{(x,z)|x_{min} < x < x_{max}, h(x) < z < H\} \tag{2.6}$$

subject to the following boundary conditions:

$$\left.\begin{array}{rcl} u(x = x_{min}) & = & u(x = x_{max}) \\ w(x = x_{min}) & = & w(x = x_{max}) \\ \theta(x = x_{min}) & = & \theta(x = x_{max}) \\ \rho(x = x_{min}) & = & \rho(x = x_{max}) \end{array}\right\} \text{ periodic lateral boundary conditions} \tag{2.7}$$
$$w(z = h) = u\partial_x h \tag{2.8}$$
$$w(z = H) = 0 \tag{2.9}$$

In the beginning, we will solve the problem with the so called linear lower boundary condition: $\Omega = (x_{min}, x_{max}) \times (0, H)$, $w(z = 0) = u\partial_x h$. In addition, we use a diffusive sponge or, alternatively, a Rayleigh damping, as wave absorber in the upper part of the domain.

The initial conditions for θ and ρ are taken from the hydrostatic reference state introduced below. u is initially set to a constant value u_0 throughout the whole domain and w is set to zero:

$$(u, w, \theta, \rho)|_{t=0} = (u_0, 0, \overline{\theta}, \overline{\rho}) \tag{2.10}$$

In our further derivation of the equations and the numerical scheme, we follow [5] closely. The main differences are:

- twodimensional problem $\Rightarrow$ no coriolis terms
- interest in mountain waves $\Rightarrow$ treating buoyancy in large timestep
- finite volume z-grid $\Rightarrow$ no metric terms

We introduce a hydrostatic reference state with constant Brunt-Väisälä frequency N and deviations from this reference state for ρ, θ and p:

$$\rho = \overline{\rho}(z) + \rho' \tag{2.11}$$
$$\theta = \overline{\theta}(z) + \theta' \tag{2.12}$$
$$p = \overline{p}(z) + p' \tag{2.13}$$

The reference state is defined by

$$N^2 = \frac{g}{\overline{\theta}} \partial_z \overline{\theta} \quad \Rightarrow \overline{\theta}(z) = \theta_0 \; e^{z N^2/g} \tag{2.14}$$
$$\overline{p} = p_0 \left(\frac{R \overline{\rho} \overline{\theta}}{p_0} \right)^{\gamma} \tag{2.15}$$
$$\partial_z \overline{p} = -g \overline{\rho} \tag{2.16}$$

where θ_0 is the surface temperature.

Inserting this first perturbation (selectively) into the prognostic equations yields:

$$\partial_t(\rho u) + \partial_x(\rho u^2 + p') + \partial_z(\rho u w) = 0 \tag{2.17}$$
$$\partial_t(\rho w) + \partial_x(\rho u w) + \partial_z(\rho w^2 + p') + g\rho' = 0 \tag{2.18}$$
$$\partial_t(\rho \theta) + \partial_x(\rho u \theta) + \partial_z(\rho w \theta) = 0 \tag{2.19}$$
$$\partial_t \rho' + \partial_x(\rho u) + \partial_z(\rho w) = 0 \tag{2.20}$$

We define the flux quantities

$$U = \rho u \tag{2.21}$$
$$W = \rho w \tag{2.22}$$
$$\Theta = \rho \theta \tag{2.23}$$

and write the prognostic equations in the form:

$$\partial_t U + \partial_x(uU) + \partial_x p' + \partial_z(wU) = 0 \tag{2.24}$$
$$\partial_t W + \partial_x(uW) + \partial_z(wW) + \partial_z p' + g\rho' = 0 \tag{2.25}$$
$$\partial_t \Theta + \partial_x(u\Theta) + \partial_z(w\Theta) = 0 \tag{2.26}$$
$$\partial_t \rho' + \partial_x U + \partial_z W = 0 \tag{2.27}$$

2.2 Pure Finite Volume Fully Explicit Scheme

We introduce the following vector notation for the state $\boldsymbol{r}$:

$$\boldsymbol{r} = \begin{pmatrix} U \\ W \\ \Theta' \\ \rho' \end{pmatrix} \tag{2.28}$$

where $\Theta' = \Theta - \overline{\Theta}$ and $\overline{\Theta} = \overline{\rho}\overline{\theta}$.

Equations (2.24) – (2.27) can then be written as:

$$\partial_t \boldsymbol{r} = -\partial_x \boldsymbol{f}_x(\boldsymbol{r}, z) - \partial_z \boldsymbol{f}_z(\boldsymbol{r}, z) + \boldsymbol{g}(\boldsymbol{r}) =: \boldsymbol{R}(\boldsymbol{r}) \tag{2.29}$$

with

$$\boldsymbol{f}_x(\boldsymbol{r}, z) = \begin{pmatrix} U^2/\rho + p'(\Theta', z) \\ UW/\rho \\ U\Theta/\rho \\ U \end{pmatrix}, \quad \boldsymbol{f}_z(\boldsymbol{r}, z) = \begin{pmatrix} UW/\rho \\ W^2/\rho + p'(\Theta', z) \\ W\Theta/\rho \\ W \end{pmatrix} \tag{2.30}$$

$$\text{and } \boldsymbol{g}(\boldsymbol{r}) = \begin{pmatrix} 0 \\ -g\rho' \\ 0 \\ 0 \end{pmatrix} \tag{2.31}$$

The pressure deviation p' can be obtained from Θ':

$$p' = p - \overline{p} = p_0 \left(\frac{R\Theta}{p_0}\right)^{\gamma} - p_0 \left(\frac{R\overline{\Theta}}{p_0}\right)^{\gamma} = \underbrace{p_0 \left(\frac{R\overline{\Theta}}{p_0}\right)^{\gamma}}_{\overline{p}} \cdot \left(\left(\frac{\Theta}{\overline{\Theta}}\right)^{\gamma} - 1\right)$$

$$p'(\Theta', z) = \overline{p}(z) \left(\left(1 + \frac{\Theta'}{\overline{\Theta}(z)}\right)^{\gamma} - 1\right) \tag{2.32}$$

For small values of $\Theta'/\overline{\Theta}(z)$, this expression might lead to cancellation errors. This can be circumvented by doing a taylor expansion. In our concrete case we just use double precision arithmetics to reduce cancellation errors.

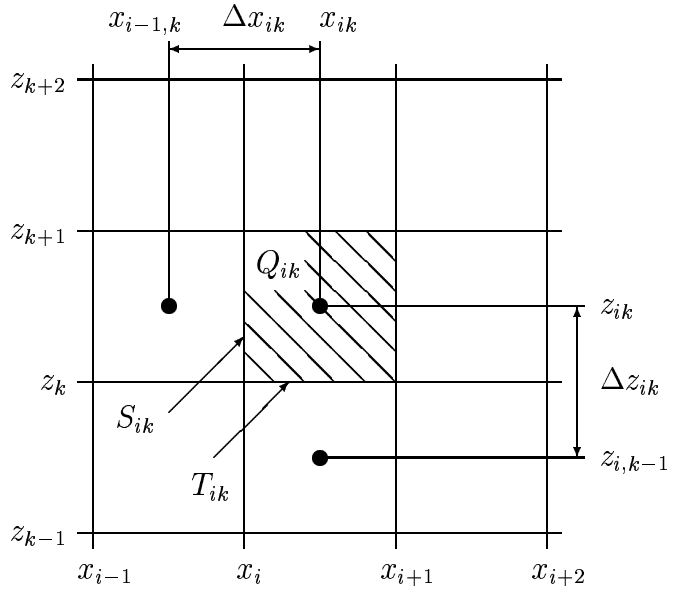

Figure 2.1: Grid with grid coordinates, gridcells, gridcell center coordinates, gridcell center distances and interfaces.

For the discretization, we introduce a grid (see figure 2.1), i. e. a partition of the computational domain $\Omega = (x_{min}, x_{max}) \times (z_{min}, z_{max})$ into $I \times K$ rectangular finite volumes $Q_{ik} = (x_i, x_{i+1}) \times (z_k, z_{k+1})$, setting $x_{min} = x_1 < x_2 < ... < x_I < x_{I+1} = x_{max}$ and $z_{min} = z_1 < z_2 < ... < z_K < z_{K+1} = z_{max}$. For purpose of generalization, we use the notations x_{ik} and z_{ik} for the horizontal and the vertical coordinate, respectively, of the center of cell Q_{ik}. We further denote the average of any variable ϕ over cell Q_{ik} by $\phi_{ik} = \frac{1}{|Q_{ik}|} \int_{Q_{ik}} \phi \mathrm{d}a$. However, we will reinterpret the ϕ_{ik} in the finite difference context later as the value in the center of cell Q_{ik} i. e. $\phi_{ik} = \phi(x_{ik}, z_{ik})$. Finally, we define the interfaces $S_{ik} = \overline{Q_{i-1,k}} \cap \overline{Q_{ik}}$ and $T_{ik} = \overline{Q_{i,k-1}} \cap \overline{Q_{ik}}$.

The finite volume discretization is now achieved by requiring equation (2.29) to hold on average over each cell:

$$\partial_t \boldsymbol{r}_{ik} = \boldsymbol{R}_{ik}(\boldsymbol{r}) \tag{2.33}$$

We can now carry out integration, using Gauss' theorem:

$$\boldsymbol{R}_{ik}(\boldsymbol{r}) = \frac{1}{|Q_{ik}|} \int_{\partial Q_{ik}} (-\boldsymbol{f}_x(\boldsymbol{r}, z) n_x - \boldsymbol{f}_z(\boldsymbol{r}, z) n_z) \mathrm{d}s + \frac{1}{|Q_{ik}|} \int_{Q_{ik}} \boldsymbol{g}(\boldsymbol{r}, z) \mathrm{d}a \tag{2.34}$$

∂Q_{ik} denotes the boundary of Q_{ik}, and $\boldsymbol{n} = (n_x, n_z)^T$ is the outer unit normal on this boundary. We now approximate the flux vectors $\boldsymbol{f}_x$ and $\boldsymbol{f}_z$ by numerical fluxes $\boldsymbol{F}_x$ and $\boldsymbol{F}_z$ and split up the boundary integral as $\partial Q_{ik} = S_{ik} \cup T_{ik} \cup S_{i+1,k} \cup T_{i,k+1}$. We assume here

that all S_{ik} are vertical, i. e. $\boldsymbol{n} = \pm(1,0)^T$ and all T_{ik} are horizontal, i. e. $\boldsymbol{n} = \pm(0,1)$. We then get:

$$\begin{aligned}\boldsymbol{R}_{ik} &= \frac{|S_{ik}|}{|Q_{ik}|}\boldsymbol{F}_x(\boldsymbol{r}_{i-1,k},\boldsymbol{r}_{ik},z_{ik}) + \frac{|T_{ik}|}{|Q_{ik}|}\boldsymbol{F}_z(\boldsymbol{r}_{i,k-1},\boldsymbol{r}_{ik},z_k) \\ &\quad -\frac{|S_{i+1,k}|}{|Q_{ik}|}\boldsymbol{F}_x(\boldsymbol{r}_{ik},\boldsymbol{r}_{i+1,k},z_{ik}) - \frac{|T_{i,k+1}|}{|Q_{ik}|}\boldsymbol{F}_z(\boldsymbol{r}_{ik},\boldsymbol{r}_{i,k+1},z_{k+1}) + \boldsymbol{g}(\boldsymbol{r}_{ik},z_{ik})\end{aligned} \tag{2.35}$$

The spatial discretization is then fully defined by choosing the numerical fluxes. We choose an averaging flux which yields a central difference scheme (CDS):

$$\boldsymbol{F}_x(\boldsymbol{r}_{i-1,k},\boldsymbol{r}_{ik},z) = \boldsymbol{f}_x\left(\frac{\boldsymbol{r}_{i-1,k}+\boldsymbol{r}_{ik}}{2},z\right) \tag{2.36}$$

$$\boldsymbol{F}_z(\boldsymbol{r}_{i,k-1},\boldsymbol{r}_{ik},z) = \boldsymbol{f}_z\left(\frac{\boldsymbol{r}_{i,k-1}+\boldsymbol{r}_{ik}}{2},z\right) \tag{2.37}$$

The boundary conditions here can be chosen in terms of fluxes over the boundary. For the horizontal flux at the lateral boundary, we have periodic boundary conditions:

$$\boldsymbol{F}_x(\boldsymbol{r}_{0,k},\boldsymbol{r}_{1,k}) = \boldsymbol{F}_x(\boldsymbol{r}_{I,k},\boldsymbol{r}_{I+1,k}) := \boldsymbol{F}_x(\boldsymbol{r}_{I,k},\boldsymbol{r}_{1,k}) \tag{2.38}$$

For the vertical flux at the top and the bottom, we put:

$$\boldsymbol{F}_z(\boldsymbol{r}_{i,0},\boldsymbol{r}_{i,1}) := \boldsymbol{f}_z(\boldsymbol{r}_i^{bottom}) \tag{2.39}$$

$$\boldsymbol{F}_z(\boldsymbol{r}_{i,K},\boldsymbol{r}_{i,K+1}) := \boldsymbol{f}_z(\boldsymbol{r}_i^{top}) \tag{2.40}$$

where $\boldsymbol{r}_i^{bottom}$ and $\boldsymbol{r}_i^{top}$ are obtained by a suitable extrapolation of U, Θ' and ρ' and setting

$$W_i^{bottom} = U_i^{bottom}\cdot\partial_x h(x)|_{x=x_{i,1}} \tag{2.41}$$

$$W_i^{top} = 0 \tag{2.42}$$

Notice that at the bottom, we do not have $\boldsymbol{n} = (0,-1)^T$ but $\boldsymbol{n} = (\partial_x h,-1)^T/\sqrt{1+(\partial_x h)^2}$. The true flux over the boundary then is

$$n_x\boldsymbol{f}_x(\boldsymbol{r}^{bottom}) + n_z\boldsymbol{f}_z(\boldsymbol{r}^{bottom}) = \frac{1}{\sqrt{1+(\partial_x h)^2}}\begin{pmatrix}(\partial_x h)p'\\ -p'\\ 0\\ 0\end{pmatrix} \tag{2.43}$$

The resulting flux over the boundary does not vanish entirely. Only the convective fluxes vanish, while the vertical pressure gradient has to balance the gravitation in the source term $\boldsymbol{g}$ and the horizontal pressure gradient represents the mountain drag.

In time, we use a leapfrog discretization:

$$\boldsymbol{r}_{ik}^{n+1} = \boldsymbol{r}_{ik}^{n-1} + 2\ \Delta t\ \boldsymbol{R}_{ik}(\boldsymbol{r}^n) \tag{2.44}$$

We use a simple forward Euler time step to start the integration:

$$\boldsymbol{r}_{ik}^{n+1} = \boldsymbol{r}_{ik}^{n} +\ \Delta t\ \boldsymbol{R}_{ik}(\boldsymbol{r}^n) \quad \text{for } n=0 \tag{2.45}$$

2.3 Finite Volume/Finite Difference Split Explicit Scheme

We discretize equations (2.24) – (2.27) in time using the split explicit scheme introduced in [6]. To this end, we express the pressure gradient terms in terms of Θ using the relation (see [5], p. 2):

$$\boldsymbol{\nabla} p \;=\; \gamma R \pi \boldsymbol{\nabla} \Theta \tag{2.46}$$

where $\pi = (p/p_0)^\kappa$ and $\kappa = R/c_p$. The resulting unapproximated prognostic equations are:

$$\partial_t U + \partial_x(uU) + \partial_z(wU) + \gamma R \pi \partial_x \Theta' = 0 \tag{2.47}$$

$$\partial_t W + \partial_x(uW) + \partial_z(wW) + \gamma R \pi \partial_z \Theta' - g\overline{\rho}\frac{\pi'}{\overline{\pi}} + g\rho' = 0 \tag{2.48}$$

$$\partial_t \Theta' + \partial_x(u\Theta) + \partial_z(w\Theta) = 0 \tag{2.49}$$

$$\partial_t \rho' + \partial_x U + \partial_z W = 0 \tag{2.50}$$

Now we group the terms to be integrated in the small timestep on the left, while terms on the right are evaluated at the latest time level n corresponding to the large timestep and held constant during the small timestep. We use this state at time level n for a second perturbation:

$$U'' = \quad U - U^n \tag{2.51}$$

$$W'' = \quad W - W^n \tag{2.52}$$

$$\Theta'' = \quad \Theta - \Theta^n \quad = \Theta' - \Theta'^n \tag{2.53}$$

$$\rho'' = \quad \rho - \rho^n \quad = \rho' - \rho'^n \tag{2.54}$$

$$\boldsymbol{r}'' = \begin{pmatrix} U'' \\ W'' \\ \Theta'' \\ \rho'' \end{pmatrix} = \boldsymbol{r} - \boldsymbol{r}^n \tag{2.55}$$

Using these relations we obtain:

$$\partial_t U'' + \gamma R \pi^n \partial_x \Theta'' \;=\; -\partial_x p'^n - \partial_x(u^n U^n) - \partial_z(w^n U^n) \tag{2.56}$$

$$\partial_t W'' + \gamma R \pi^n \partial_z \Theta'' - g\overline{\rho}\frac{R}{c_v}\frac{\pi^n}{\overline{\pi}}\frac{\Theta''}{\overline{\Theta}} \;=\; -\partial_z p'^n - \partial_x(u^n W^n) - \partial_z(w^n W^n) - g\rho'^n \tag{2.57}$$

$$\partial_t \Theta'' + \partial_x(U''\theta^n) + \partial_z(W''\theta^n) \;=\; -\partial_x(u^n\Theta^n) - \partial_z(w^n\Theta^n) \tag{2.58}$$

$$\partial_t \rho'' + \partial_x U'' + \partial_z W'' \;=\; -\partial_x U^n - \partial_z W^n \tag{2.59}$$

We now discretize in time using partially split operators as described in [4], p. 365. Let $\Delta\tau$ denote the small timestep and $\Delta t = M\Delta\tau$ the large timestep. We use the index

m for time layers associated with the small timestep and the index n for time layers associated with the large timestep. Thus, a timelayer n is equivalent to a time layer m if $m = nM$. As the right hand side is held constant during the small timesteps, we denote it by R_U^n, R_W^n, R_Θ^n and R_ρ^n, respectively:

$$\begin{aligned} R_U^n &= -\partial_x p'^n - \partial_x(u^n U^n) - \partial_z(w^n U^n) && (2.60)\\ R_W^n &= -\partial_z p'^n - \partial_x(u^n W^n) - \partial_z(w^n W^n) - g\rho'^n && (2.61)\\ R_\Theta^n &= -\partial_x(u^n \Theta^n) - \partial_z(w^n \Theta^n) && (2.62)\\ R_\rho^n &= -\partial_x U^n - \partial_z W^n && (2.63) \end{aligned}$$

Note that comparing to (2.29), we have

$$\begin{pmatrix} R_U^n \\ R_W^n \\ R_\Theta^n \\ R_\rho^n \end{pmatrix} = \boldsymbol{R}(\boldsymbol{r}^n) \tag{2.64}$$

2.3.1 Small Timestep

For the small timestep $\Delta\tau$, we use the forward-backward integration scheme with implicit coupling in the vertical, as in [5] and [6]:

$$\begin{aligned} \frac{U''^{m+1} - U''^m}{\Delta\tau} + \underbrace{a^n \partial_x \Theta''^m}_{\partial_x p''} &= R_U^n && (2.65)\\ \frac{W''^{m+1} - W''^m}{\Delta\tau} + \underbrace{a^n \partial_z \Theta''^{m+\beta} - b^n \Theta''^{m+\beta}}_{\partial_z p'' + g\rho''} &= R_W^n && (2.66)\\ \frac{\Theta''^{m+1} - \Theta''^m}{\Delta\tau} + \partial_x(U''^{m+1}\theta^n) + \partial_z(W''^{m+\beta}\theta^n) &= R_\Theta^n && (2.67)\\ \frac{\rho''^{m+1} - \rho''^m}{\Delta\tau} + \partial_x U''^{m+1} + \partial_z W''^{m+\beta} &= R_\rho^n && (2.68) \end{aligned}$$

with

$$a^n = \gamma R \pi^n, \qquad b^n = \frac{g\overline{\rho} R \pi^n}{c_v \overline{\pi}\,\overline{\Theta}}$$

We have used the average in vertical terms:

$$\phi^{m+\beta} = \beta\phi^{m+1} + (1-\beta)\phi^m, \quad 0 < \beta < 1 \tag{2.69}$$

It is usual to forward bias these terms, i. e. to choose $\beta > 0.5$, e. g. $\beta = 0.6$.

We now discretize the small timestep terms in space using centered differences. In order to simplify notation as well as implementation, we introduce the centered finite difference operators:

$$\delta_{2x}(\phi)_{ik} = \frac{\phi_{i+1,k} - \phi_{i-1,k}}{\Delta x_{ik} + \Delta x_{i,k+1}} \tag{2.70}$$

$$\delta_{2z}(\phi)_{ik} = \frac{\phi_{i,k+1} - \phi_{i,k-1}}{\Delta z_{ik} + \Delta z_{i,k+1}} \tag{2.71}$$

$$\delta_z^2(\phi)_{ik} = \frac{2}{\Delta z_{ik} + \Delta z_{i,k+1}} \left(\frac{\phi_{i,k+1} - \phi_{ik}}{\Delta z_{i,k+1}} - \frac{\phi_{ik} - \phi_{i,k-1}}{\Delta z_{ik}} \right) \tag{2.72}$$

with

$$\begin{aligned} \Delta x_{ik} &= x_{ik} - x_{i,k-1} \\ \Delta z_{ik} &= z_{ik} - z_{i,k-1} \end{aligned}$$

First, the horizontal momentum perturbation U'' is updated:

$$U''^{m+1} = U''^{m} + \Delta\tau \left(R_U^n - a^n \delta_{2x}(\Theta''^{m}) \right) \tag{2.73}$$

Then, the vertically coupled implicit equations (2.66) and (2.67) are solved. The system can be written as:

$$\Theta''^{m+1} = \Theta^* - \Delta\tau\beta\partial_z(W''^{m+1}\theta^n) \tag{2.74}$$

$$W''^{m+1} = W^* + \Delta\tau\beta \left(-a^n \partial_z \Theta''^{m+1} + b^n \Theta''^{m+1} \right) \tag{2.75}$$

where

$$\begin{aligned} \Theta^* &= \Theta'^{m} + \Delta\tau \left(R_\Theta^n - \partial_x(U''^{m+1}\theta^n) - (1-\beta)\partial_z(W''^{m}\theta^n) \right) \\ W^* &= W''^{m} + \Delta\tau \left(R_W^n - (1-\beta)a^n \partial_z \Theta''^{m} + (1-\beta)b^n \Theta''^{m} \right) \end{aligned}$$

Inserting (2.74) into (2.75) yields

$$W''^{m+1} = W^{**} + (\Delta\tau\beta)^2 a^n \partial_z^2(W''^{m+1}\theta^n) - (\Delta\tau\beta)^2 b^n \partial_z(W''^{m+1}\theta^n) \tag{2.76}$$

with

$$W^{**} = W^* + \Delta\tau\beta \left(-a^n \partial_z \Theta^* + b^n \Theta^* \right)$$

Equation (2.76) can now be discretized:

$$W_{ik}''^{m+1} = W_{ik}^{**} + (\Delta\tau\beta)^2 a_{ik}^n \delta_z^2(W''^{m+1}\theta^n)_{ik} - (\Delta\tau\beta)^2 b_{ik}^n \delta_{2z}(W''^{m+1}\theta^n)_{ik} \tag{2.77}$$

$$\begin{aligned} &= W_{ik}^{**} + (C_{ik}^n + B_{ik}^n)\theta_{i,k-1}^n W_{i,k-1}''^{m+1} \\ &\quad - D_{ik}^n \theta_{ik}^n W_{ik}''^{m+1} + (E_{ik}^n - B_{ik}^n)\theta_{i,k+1}^n W_{i,k+1}''^{m+1} \quad \text{for } k = 1 \ldots K \end{aligned} \tag{2.78}$$

$$
\begin{aligned}
W^{**}_{ik} &= (-B^n_{ik} - C^n_{ik})\theta^n_{i,k-1} W''^{m+1}_{i,k-1} + \\
&\quad (1 + D^n_{ik}\theta^n_{ik}) W''^{m+1}_{ik} + \\
&\quad +(B^n_{ik} - E^n_{ik})\theta^n_{i,k+1} W''^{m+1}_{i,k+1} \quad \text{for } k = 1 \ldots K
\end{aligned} \tag{2.79}
$$

where

$$
\begin{aligned}
B^n_{ik} &= \frac{(\Delta\tau\beta)^2 b^n_{ik}}{\Delta z_{ik} + \Delta z_{i,k+1}} \\
C^n_{ik} &= \frac{2(\Delta\tau\beta)^2 a^n_{ik}}{\Delta z_{ik}(\Delta z_{ik} + \Delta z_{i,k+1})} \\
D^n_{ik} &= \frac{2(\Delta\tau\beta)^2 a^n_{ik}}{\Delta z_{ik}\Delta z_{i,k+1}} \\
E^n_{ik} &= \frac{2(\Delta\tau\beta)^2 a^n_{ik}}{\Delta z_{i,k+1}(\Delta z_{ik} + \Delta z_{i,k+1})}
\end{aligned}
$$

and the discretization of W^{**}, W^*, Θ^*, a^n and b^n is straightforward. The values for $W''^{m+1}_{i,0}$ and $W''^{m+1}_{i,K+1}$ needed in (2.79) can be obtained from the boundary conditions (2.8) and (2.9). At the bottom, i. e. for $k = 1$, we require the discrete linearized version of (2.8):

$$
{}^1\!/_2(W''^{m+1}_{i,k-1} + W''^{m+1}_{i,k}) = \partial_x h|_{x=x_{ik}} U''^{m+1}_{ik}
$$

Thus the value of $W''^{m+1}_{i,k-1}$ is

$$
W''^{m+1}_{i,k-1} := 2\partial_x h(x)|_{x=x_{ik}} U''^{m+1}_{ik} - W''^{m+1}_{ik} \quad \text{for } k = 1 \tag{2.80}
$$

For $\theta^n_{i,k-1}$, we just do constant extrapolation:

$$
\theta^n_{i,k-1} := \theta^n_{ik} \tag{2.81}
$$

For $k = K$, i. e. at the top, we use the discrete version of (2.9):

$$
W''^{m+1}_{i,k+1} := 0 \quad \text{for } k = K \tag{2.82}
$$

Equation (2.79) with (2.80) - (2.82) forms a nonsymmetric tridiagonal system which can be solved using a suitable algorithm. The Thomas algorithm, which is widely used for tridiagonal systems, requires diagonal dominance of the matrix. With some simplifications, one can show that the criterion for diagonal dominance is:

$$
\Delta\tau\beta < \Delta z \sqrt{\frac{2}{\gamma RT}}
$$

As $\sqrt{\gamma RT}$ is the speed of sound, this means that again the sound mode dictates the timestep, although this is what we wanted to avoid by treating the vertical terms implicitly. So we use a gauss algorithm with relative column maximum pivoting strategy for tridiagonal systems, from [7], page 50.

As W''^{m+1} is now known, the update of Θ'' can be done using the discrete form of equation (2.74):

$$\Theta''^{m+1} = \Theta^* - \Delta\tau\beta\delta_{2z}(W''^{m+1}\theta^n) \tag{2.83}$$

Finally, the density perturbation ρ'' is updated:

$$\rho''^{m+1} = \rho''^{m} + \Delta\tau\left(R_\rho^n - \delta_{2x}(U''^{m+1}) - \delta_{2z}(W''^{m+\beta})\right) \tag{2.84}$$

If we consider the above finite difference discretization to be the definition of the nonlinear propagation operator $\mathcal{F}$, we can write a small timestep as:

$$\boldsymbol{r}''^{m+1} = \mathcal{F}(\boldsymbol{r}^n)\boldsymbol{r}''^{m} \tag{2.85}$$

2.3.2 Large Timestep

For the large timestep Δt, we use a leapfrog scheme:

$$\boldsymbol{r}^{n+1} - \boldsymbol{r}^n = [\mathcal{F}(\boldsymbol{r}^n)]^{2M}(\boldsymbol{r}^{n-1} - \boldsymbol{r}^n) \tag{2.86}$$

That means that $\boldsymbol{r}''$ is integrated in time using small timesteps of $\mathcal{F}(\boldsymbol{r}^n)$ where $\boldsymbol{r}^n$ is from an intermediate time layer. Thus, the state $\boldsymbol{r}^n$ used in this step is obtained from the previous leapfrog step from $\boldsymbol{r}^{n-2}$ to $\boldsymbol{r}^n$. We use a simple forward Euler time step to start the integration:

$$\boldsymbol{r}^{n+1} - \boldsymbol{r}^n = [\mathcal{F}(\boldsymbol{r}^n)]^{M}(\boldsymbol{r}^n - \boldsymbol{r}^n) \quad \text{for } n = 0 \tag{2.87}$$

The large timestep terms condensed in $\boldsymbol{R}$ are discretized in space with finite volumes. As $\boldsymbol{R}$ has the same form as in the pure finite volume scheme introduced in section 2.2, we use exactly the same discretization.

2.4 Numerical Reference State

The reference state is defined by equations (2.14) – (2.16). In order to obtain a balanced scheme, the hydrostatic terms have to balance numerically. That means that the discretized form of the hydrostatic balance equation (2.16) has to hold exactly when using the same discretization as for the rest of the scheme. See [1] for more details about balancing.

First, we can set the discrete reference potential temperature analytically, as the discrete hydrostatic balance can be fulfilled for any reference potential temperature profile. Inserting discrete values for z into (2.14) yields immediately:

$$\overline{\theta}_{ik} = \theta_0 e^{z_{ik}N^2/g} \tag{2.88}$$

As we will use the equation of state (2.15) to diagnose $\overline{p}_{ik}$ from $\overline{\theta}_{ik}$ and $\overline{\rho}_{ik}$, we have to plug (2.15) into (2.16) and discretize with finite volumes:

$$\begin{aligned}
0 &= -\partial_z \left(p_0 \left(\frac{R\overline{\theta}\overline{\rho}}{p_0} \right)^\gamma \right) - g\overline{\rho} \\
&= -p_0 \left(\frac{R}{p_0} \right)^\gamma \partial_z \left((\overline{\theta}\overline{\rho})^\gamma \right) - g\overline{\rho} \\
0 &= -\frac{1}{|Q_{ik}|} p_0 \left(\frac{R}{p_0} \right)^\gamma \int_{\partial Q_{ik}} (\overline{\rho}\overline{\theta})^\gamma n_z \mathrm{d}s - g\overline{\rho}_{ik} \\
0 &= -\frac{1}{|Q_{ik}|} p_0 \left(\frac{R}{p_0} \right)^\gamma \frac{(\overline{\rho}_{i,k+1}\overline{\theta}_{i,k+1})^\gamma + (\overline{\rho}_{ik}\overline{\theta}_{ik})^\gamma}{2} |T_{i,k+1}| \\
&\quad + \frac{1}{|Q_{ik}|} p_0 \left(\frac{R}{p_0} \right)^\gamma \frac{(\overline{\rho}_{ik}\overline{\theta}_{ik})^\gamma + (\overline{\rho}_{i,k-1}\overline{\theta}_{i,k-1})^\gamma}{2} |T_{ik}| - g\overline{\rho}_{ik}
\end{aligned} \tag{2.89}$$

For $k = 1$, the values of $\overline{\rho}_{ik}$ and $\overline{\rho}_{i,k-1}$ can be derived from analytical expressions, as the discrete balance (2.89) can still be achieved. Then, successive evaluation of (2.89) for $k = 1, 2, 3, \ldots, K-1$ yields all other values for $\overline{\rho}_{i,k+1}$. A reformulation of (2.89) yields the following explicit expression:

$$\overline{\rho}_{i,k+1} = \frac{1}{\overline{\theta}_{i,k+1}} \left(\frac{|T_{ik}|}{|T_{i,k+1}|} [(\overline{\rho}_{ik}\overline{\theta}_{ik})^\gamma + (\overline{\rho}_{i,k-1}\overline{\theta}_{i,k-1})^\gamma] - 2\frac{|Q_{ik}|}{|T_{i,k+1}|} \left(\frac{p_0}{R} \right)^\gamma \frac{g}{p_0} \overline{\rho}_{ik} - (\overline{\rho}_{ik}\overline{\theta}_{ik})^\gamma \right)^{1/\gamma}$$

Having computed discrete values for $\overline{\theta}_{ik}$ and $\overline{\rho}_{ik}$, we can diagnose $\overline{\Theta}_{ik}$ and $\overline{p}_{ik}$:

$$\overline{\Theta}_{ik} = \overline{\rho}_{ik}\overline{\theta}_{ik} \tag{2.90}$$

$$\overline{p}_{ik} = p_0 \left(\frac{R\overline{\Theta}_{ik}}{p_0} \right)^\gamma \tag{2.91}$$

2.5 Damping Terms

In order to damp acoustic waves and wave reflection at the upper boundary of the domain, three well known methods are implemented. All of them modify equation (2.29) by adding a damping term $\boldsymbol{D}$ on the right:

$$\partial_t \boldsymbol{r} = \boldsymbol{R}(\boldsymbol{r}) + \boldsymbol{D}(\boldsymbol{r}) \tag{2.92}$$

All damping terms are introduced into the previously described schemes simply by a performing a suitable discretization of $\boldsymbol{D}$ into $\boldsymbol{D}_{ik}$ and substituting $\boldsymbol{R}_{ik}$ by $\boldsymbol{R}_{ik} + \boldsymbol{D}_{ik}$ in the scheme.

2.5.1 Diffusive Sponge

The diffusive sponge damps reflection at the upper boundary by introducing a horizontal diffusion of the prognostic variables in the upper part of the computational domain:

$$\begin{aligned} \boldsymbol{D}_s(\boldsymbol{r}) &= c_s(z)\ \partial_{xx}\boldsymbol{r} \\ c_s(z) &= \gamma_s\ \frac{1}{2}\left(1-\cos\left(\pi\frac{z-z_s}{z_{max}-z_s}\right)\right) \quad \text{if } z > z_s \\ c_s(z) &= 0 \quad \text{if } z < z_s \end{aligned} \tag{2.93}$$

γ_s is the maximal diffusion coefficient, z_s is the height where the diffusion starts.

This damping term is discretized using the finite difference operator δ_x^2, defined the same way as δ_z^2.

2.5.2 Rayleigh Damping

Rayleigh damping does a first order relaxation of the state towards the reference state in the upper part of the computational domain in order to damp wave reflection at the upper boundary:

$$\begin{aligned} \boldsymbol{D}_r(\boldsymbol{r}) &= c_r(z)\ (\overline{\boldsymbol{r}}-\boldsymbol{r}) \\ c_r(z) &= \gamma_r\ \frac{1}{2}\left(1-\cos\left(\pi\frac{z-z_r}{z_{max}-z_r}\right)\right) \quad \text{if } z > z_r \\ c_r(z) &= 0 \quad \text{if } z < z_r \end{aligned} \tag{2.94}$$

γ_r is the maximal relaxation coefficient, z_r is the height where the relaxation starts. $\overline{\boldsymbol{r}}$ denotes the reference state:

$$\overline{\boldsymbol{r}} = \begin{pmatrix} u_0\overline{\rho} \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.95}$$

This damping term is discretized by simply taking the already discretized values of $\boldsymbol{r}$ and $\overline{\boldsymbol{r}}$.

2.5.3 Divergence Damping

Divergence damping mimics the effect of an additional diffusion of the reference density weighted divergence Div*, yielding a global damping of acoustic waves.

$$\mathrm{Div}^*(\boldsymbol{r}) = \partial_x(\overline{\rho}u) + \partial_z(\overline{\rho}w) = \partial_x\left(\frac{\overline{\rho}}{\rho}U\right) + \partial_z\left(\frac{\overline{\rho}}{\rho}W\right) \tag{2.96}$$

$$\boldsymbol{D}_d(\boldsymbol{r}) = \begin{pmatrix} \alpha_x\ \partial_x(\mathrm{Div}^*(\boldsymbol{r})) \\ \alpha_z\ \partial_z(\mathrm{Div}^*(\boldsymbol{r})) \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha_x\ (\partial_{xx}(U\overline{\rho}/\rho) + \partial_{xz}(W\overline{\rho}/\rho)) \\ \alpha_z\ (\partial_{xz}(U\overline{\rho}/\rho) + \partial_{zz}(W\overline{\rho}/\rho)) \\ 0 \\ 0 \end{pmatrix} \tag{2.97}$$

α_x and α_z denote the damping coefficients in x and z direction, respectively.

This damping term is discretized by replacing the derivatives by finite difference operators, namely ∂_{xx} by δ_x^2, ∂_{zz} by δ_z^2 and ∂_{xz} by $\delta_x \delta_z$.

For more details about divergence damping, see [8] and references therein.

2.6 Time Filter

Leapfrog time integration sometimes has a computational mode which has to be damped with a suitable time filter. To this end we implement an Asselin filter:

$$\tilde{\boldsymbol{r}}^n = \boldsymbol{r}^n + c_a(\tilde{\boldsymbol{r}}^{n-1} - 2\boldsymbol{r}^n + \boldsymbol{r}^{n+1}) \tag{2.98}$$

c_a is the filter coefficient. In order for the scheme to stay stable, c_a must not exceed 0.25.

3 Implementation

3.1 Introduction

For reasons of compatibility with other existing codes, the implementation was done in Fortran90. Because the code was essentially built from scratch, it was possible to take advantage of new features of Fortran90. This mainly includes the adoption of some object oriented approaches to software design. The next section gives a short overview of object orientation and its use in this project. See [2] and [3] for more information about object orientation in Fortran90.

3.2 Object Orientation

Object orientation has four main goals:

Encapsulation Keep data and procedures working on this data together

In Fortran90, this is typically done in a module. The module defines a derived type to hold the data, and module procedures (member functions) to modify them. The entire module then represents an object.

Information Hiding Hide implementation details and data structure

This can be achieved in Fortran90 by making variables in the derived type and helper procedures private. Thereby, only module procedures in the same module can modify the derived type, i. e. information hiding enforces encapsulation.

Code Inheritance Let child types inherit the parent's properties

Inheritance is not available in Fortran90, but code inheritance can be replaced without loss by aggregation, i. e. by letting the child type have a variable of the parent type.

Type Inheritance Use descendents of a type wherever the type itself can be used

Type inheritance allows polymorphism. As there is no inheritance available in Fortran90, run time type inheritance can only be faked by ugly constructs destroying the advantages of type inheritance.

Encapsulation, information hiding and code inheritance through aggregation are widely used in the code. Concering type inheritance, a pattern close to it was used to reach at least compile time polymorphism. See section 3.3.1 for more details.

3.3 Building Blocks

3.3.1 Topography

Topography information is needed in several points in the code, namely in the construction of the grid and in the application of boundary conditions in both the finite volume and the finite difference part. For the code to stay consistent, it is favorable to have one single point to specify which topography has to be used. This was realized with the following simple general topography module `topo_m`:

```
1  module topo_m
2    use bell_topo_m
3    implicit none
4    private
5    public init_topo
6    public height_topo
7  end module topo_m
```

Line 2 specifies the special topography to be used. Here, it is a bell shaped topography. Lines 4 to 6 make sure that only the procedures `init_topo` and `height_topo` are made public from the special topography module. In addition, lines 5 and 6 assert that the used special topography module offers the two procedures. Like this, any other topography module can be used provided it has the same interface.

In all other parts of the code, the module `topo_m` is used instead of e. g. `bell_topo_m`. Like this, a change of line 2 in `topo_m` changes the topography in the whole code.

The following topography modules are implemented:

module name	topography function
`no_topo_m`	$h(x) = 0$
`bell_topo_m`	$h(x) = H/(1 + (x/a)^2)$
`gaussian_topo_m`	$h(x) = H \exp(-(x/a)^2)$

3.3.2 Grid

Very much like for the topography, one would like to use different grids the same way. So the same approach has been chosen with the modules `grid_m` and `uniform_grid_m`. The uniform grid is the only grid implemented. Unfortunately, some parts of the code,

namely the finite volume part, rely on strictly horizontal and vertical grid lines. These parts have to be changed in order to allow more general grids like terrain following ones.

The module `uniform_grid_m` represents a grid of $n_x \times n_z$ grid cells which all have the same size $\Delta x \times \Delta z$.

3.3.3 Fields, Fluxes and States

Field The field is the core component of the code. The field module `field_m` defines a field type `field_t`:

```
1  type field_t
2     private
3     real, pointer, dimension(:,:) :: v
4     integer :: status
5  end type field_t
```

So a field is mainly a pointer array holding discrete values of a field ϕ_{ik}. This has the advantage that the array always has the correct size, as allocation is done by the module procedure `new` which gets the size from the grid module. By overloading operators, one can use fields very much like arrays. In addition, we can define for instance finite difference operators on a field, i. e. the Fortran equivalents to the operators δ.

Fluxes In some cases, e. g. for the evaluation of finite volume fluxes, we need fields that are defined not for the grid cells but for the grid cell interfaces. Therefore, there are the analogous types `flux_x_t` and `flux_z_t`.

State A variable of type `state_t` represents a discrete state $\boldsymbol{r} = (U, W, \Theta', \rho')^T$. It consists very naturally of four fields:

```
1  type state_t
2     type (field_t) :: U, W, Theta, rho
3  end type state_t
```

This structure allows to write big parts of the code in terms of states. The fields are not private, as many other modules need access to the fields.

Statefluxes Statefluxes are related to states as fluxes are to fields.

3.3.4 Reference State

A variable of type `ref_state_t` holds all reference state data used by the schemes:

```
1  type ref_state_t
2     type (field_t) :: Theta, rho, pi
3     type (flux_x_t) :: p_x
4     type (flux_z_t) :: p_z
4     real :: p_0, R, gamma, g
5  end type ref_state_t
```

3.3.5 Solver for the Tridiagonal System

For the solution of the tridiagonal system (2.79), a suitable solver is required. As the system is not necessarily diagonal dominant, the simple thomas algorithm is not suitable. Therefore, we implemented the gauss algorithm with relative column maximum pivoting strategy described in [7], page 50, algorithm (1.122) and (1.123). This algorithm has also linear complexity like the Thomas algorithm, but manages to solve systems which are not diagonal dominant.

The module `gauss_colmax_m` is implemented in terms of fields, such that all the tridiagonal systems are solved at once.

3.3.6 Small Timestep

Having set up the infrastructure, the module for the small timestep can easily be implemented. The module `fd_smallstep_m` encapsulates everything related to the finite difference implementation of the small timestep. The interface has been designed to fit also other possible implementations. The public procedures are namely:

```
1  subroutine new(this, ref_state, beta, timestep)
2  subroutine set_state_n(this, state, state_n, rhs)
3  subroutine do_steps(this, nbr)
4  subroutine get_state(this, state)
```

The subroutine `new` initializes the variable `this`, which is of type `smallstep_t`, with the reference state, the forward biasing coefficient β and the small timestep $\Delta\tau$. The subroutine `set_state_n` updates the operator $\mathcal{F}(\boldsymbol{r}^{n-1})$ to $\mathcal{F}(\boldsymbol{r}^n)$, where `state_n` represents $\boldsymbol{r}^n$, `rhs` represents $\boldsymbol{R}^n$, and tells it to start timestepping with $\boldsymbol{r}'' = \boldsymbol{r} - \boldsymbol{r}^n$, where $\boldsymbol{r}$ is represented by `state`. `do_steps` does `nbr` steps, and `get_state` retrieves the reached state in `state`. Note that all perturbations from $\boldsymbol{r}^n$ are made by the module `fd_smallstep_m` makeing confusions outside impossible.

For the small timestep module, the same approach could be used in order to allow other implementations of the small timestep, e. g. a finite volume type small timestep.

3.3.7 Finite Volume and Damping

The finite volume module has to calculate $\boldsymbol{R}$ from $\boldsymbol{r}$. Both $\boldsymbol{R}$ and $\boldsymbol{r}$ are represented by variables of type `state_t`. The interface of the finite volume module `fv_m` is very simple:

```
1  subroutine new(this, ref_state)
2  subroutine calc_rhs(this, state, rhs)
```

Subroutine `new` initializes the finite volume object `this` with the reference state `ref_state`. Subroutine `calc_rhs` calculates the new right hand side $\boldsymbol{R}$ (`rhs`) from `state` $\boldsymbol{r}$.

The damping terms for the acoustic waves analogously have to calculate $\boldsymbol{D}$ from $\boldsymbol{r}$. They therefore have the same structure.

4 Numerical Results

4.1 Introduction

In order to verify the model, two types of numerical experiments are carried out. In the first type of experiment, we consider a flow past topography and compare it with an analytic solution for the steady state. The analytic solution is obtained using the linear lower boundary condition, the Boussinesq approximation and a Fourier transformation. Note that the analytic solution has neither upper nor lateral boundary conditions which influence the solution.

In the second type of experiment, we consider a fluid at rest over a flat topography, with a buoyant bubble, i. e. a temperature anomaly and compare the resulting fluctuation of the state with the expected oscillation with the Brunt-Väisälä frequency N.

Due to lack of time, the implementation of the split explicit scheme could not be brought into a stable state allowing numerical experiments. All numerical experiments are therefore only carried out with the pure finite volume scheme.

4.2 Flow past Topography

We consider a flow of 10 m/s over a bell shaped hill with $h(x) = H/(1 + (x/a)^2)$. Depending on the size of a, we obtain a hydrostatic or a nonhydrostatic case, yielding two experiments. For both of them, we use Rayleigh relaxation damping, divergence damping and an Asselin time filter.

4.2.1 Hydrostatic Case

The used parameters are listed in table 4.1.

parameter		value	unit
mountain half width	a	10	km
mountain height	H	1	m
horizontal cell size	Δx	2000	m
# of horizontal grid cells	I	100	1
vert. cell size	Δz	300	m
# of vertical grid cells	K	80	1
horizontal domain size		200	km
vertical domain size		24	km
initial Brunt-Väisälä frequency	N	0.01	1/s
bottom potential temperature	θ_0	280	K
initial speed	u_0	10	m/s
timestep	Δt	0.4	s
final simulation time	t_{final}	25000	s
Rayleigh coefficient	γ_r	0.01	$1/\Delta t$
divergence damping coefficient	$\alpha_{x,z}$	1000	m^2/s
Asselin coefficient	c_a	0.05	1

Table 4.1: Parameters for simulation of hydrostatic flow past topography

The used finite volume scheme is fully explicit and does contain acoustic waves. Therefore, the sound speed dictates the timestep. The advective time scale is $T = a/u_0 = 1000s$. We simulate until $t_{final} = 25000s = 25T$. All plots show contours of $u' = u - u_0$ or w. Positive and zero contours are black, negative contours are gray.

Figure 4.1 shows the analytic solution, the numeric solution and the error of u' and w. The numeric solution has the following properties:

- structure similar to analytic solution
- wave amplitude increases with height due to effect of decreasing density
- wavelength very close to analytic solution
- amplitude of waves in u' slightly too small, in w slightly too large
- spurious oscillations close to the bottom

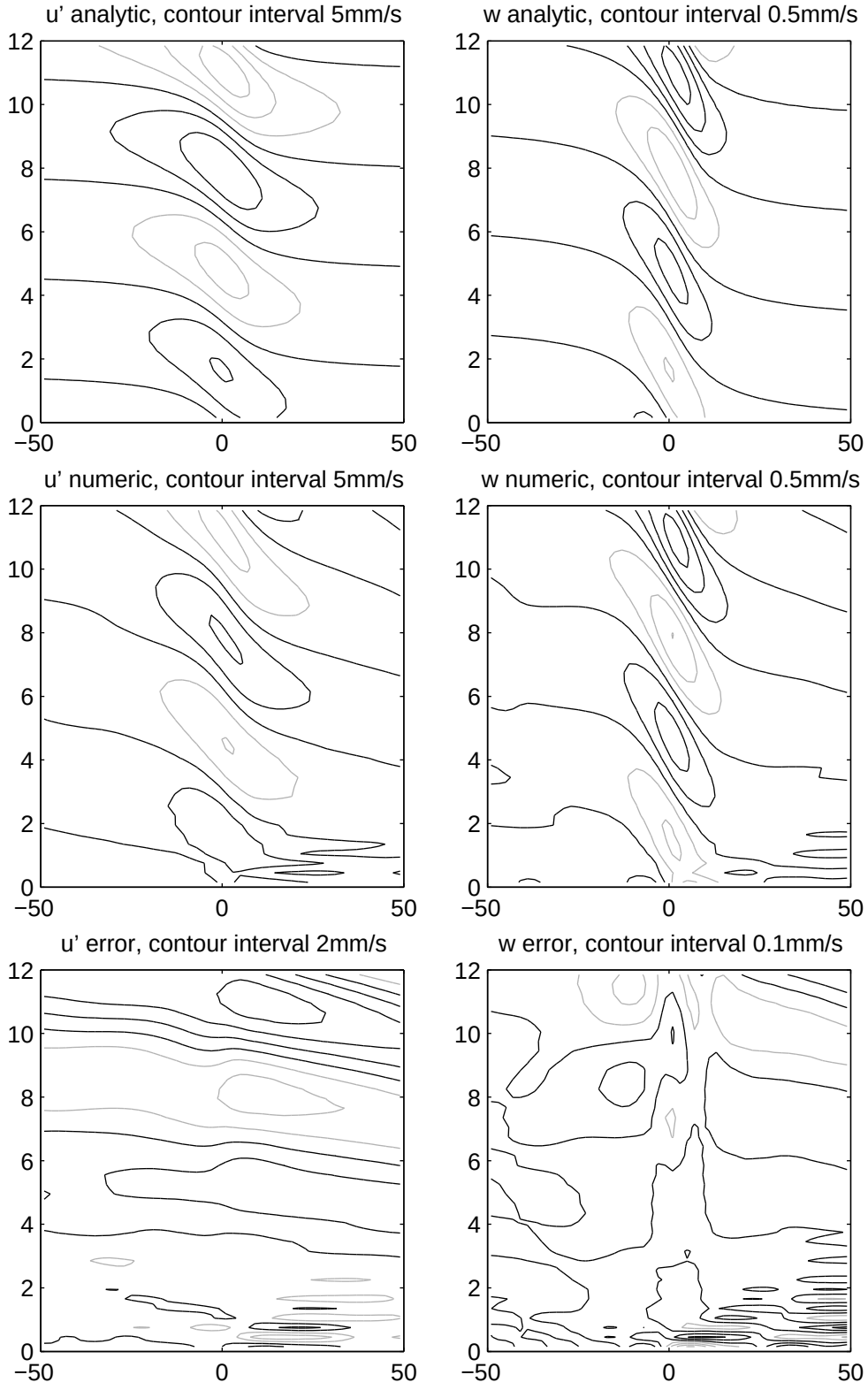

Figure 4.1: Analytic solution (top), numeric solution (center) and error (bottom) of u' (left) and w (right). Axes are x/km and z/km.

4.2.2 Nonhydrostatic Case

The used parameters are listed in table 4.2.

parameter		value	unit
mountain half width	a	1	km
mountain height	H	1	m
horizontal cell size	Δx	400	m
# of horizontal grid cells	I	100	1
vert. cell size	Δz	300	m
# of vertical grid cells	K	80	1
horizontal domain size		40	km
vertical domain size		24	km
initial Brunt-Väisälä frequency	N	0.01	1/s
bottom potential temperature	θ_0	280	K
initial speed	u_0	10	m/s
timestep	Δt	0.4	s
final simulation time	t_{final}	10000	s
Rayleigh coefficient	γ_r	0.01	$1/\Delta t$
divergence damping coefficient	$\alpha_{x,z}$	1000	m^2/s
Asselin coefficient	c_a	0.05	1

Table 4.2: Parameters for simulation of nonhydrostatic flow past topography

The advective time scale is $T = a/u_0 = 100s$. We simulate until $t_{final} = 10000s = 100T$. All plots show contours of $u' = u - u_0$ or w. Positive and zero contours are black, negative contours are grey.

Figure 4.2 shows the analytic solution, the numeric solution and the error of u' and w. The numeric solution has the following properties:

- structure similar to analytic solution
- wave amplitude increases with height due to effect of decreasing density
- wavelength very close to analytic solution
- amplitude of waves in u' slightly too small, in w slightly too large
- spurious oscillations close to the bottom

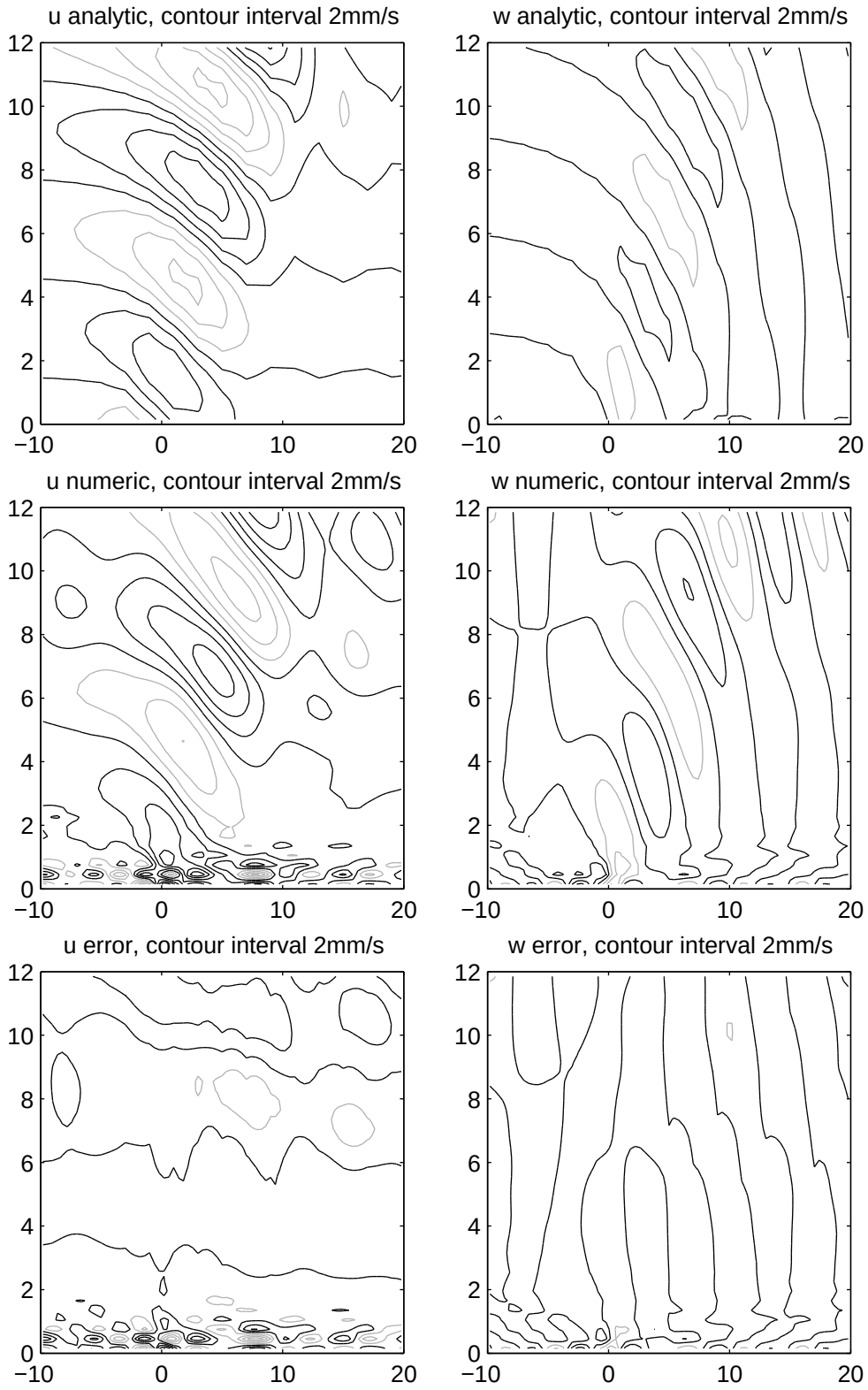

Figure 4.2: Analytic solution (top), numeric solution (center) and error (bottom) of u' (left) and w (right). Axes are x/km and z/km.

4.3 Buoyant Bubble

We consider a warm buoyant bubble in a fluid at rest. The bubble is a gaussian density anomaly of the form:

$$\rho'(x,z) = \overline{\rho}(x,z) \cdot A \exp\left(\frac{x^2}{\sigma_x^2} + \frac{(z-z_c)^2}{\sigma_z^2}\right)$$

with $A = -0.001$, $\sigma_x = \sigma_z = 1km$ and $z_c = 7km$. It may seem strange to reduce just density to get a warm bubble, but in our primary variables, Θ alone determines pressure, and as we don't want to touch pressure, we have to leave $\Theta' = 0$. So reducing the density turns out to be the correct thing to do. From theory, we expect an oscillation with the Brunt-Väisälä frequency $N = 0.01$. We use Rayleigh relaxation damping, divergence damping and an Asselin time filter. All parameters are listed in table 4.3.

parameter		value	unit
mountain half width	a	-	km
mountain height	H	0	m
horizontal cell size	Δx	300	m
# of horizontal grid cells	I	100	1
vert. cell size	Δz	300	m
# of vertical grid cells	K	80	1
horizontal domain size		30	km
vertical domain size		24	km
initial Brunt-Väisälä frequency	N	0.01	1/s
bottom potential temperature	θ_0	280	K
initial speed	u_0	0	m/s
timestep	Δt	0.4	s
final simulation time	t_{final}	60	min
Rayleigh coefficient	γ_r	0.01	$1/\Delta t$
divergence damping coefficient	$\alpha_{x,z}$	1000	m^2/s
Asselin coefficient	c_a	0.05	1
relative anomaly amplitude	A	-0.001	1
anomaly extent	$\sigma_{x,z}$	1	km
initial anomaly height	z_c	7	km

Table 4.3: Parameters for simulation of buoyant bubble

In order to check the right behaviour, we plot $\rho^*(t) = \rho'(x = 0, z = z_c, t)$ against time. Figure 4.3 shows that the resulting fluctuation is actually an osciallation with a frequency very close to N.

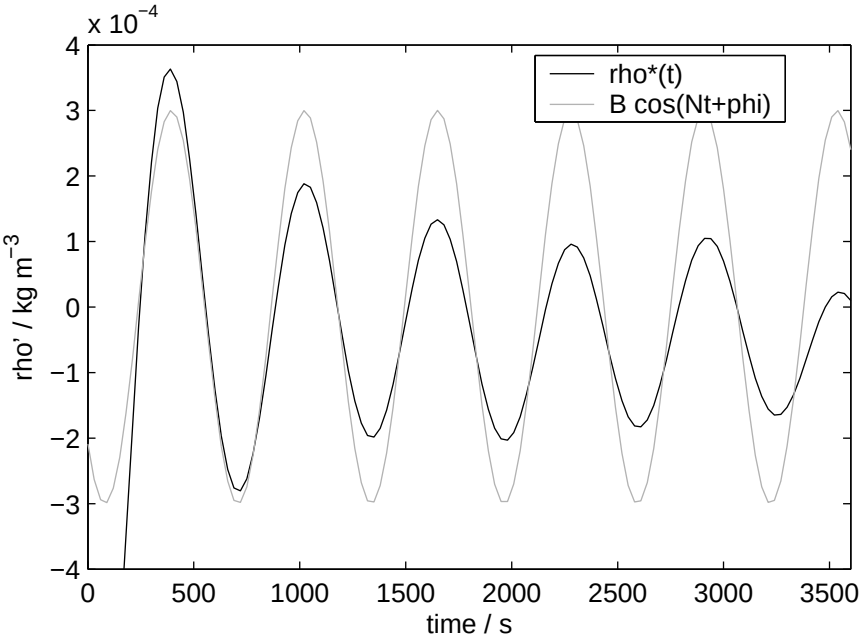

Figure 4.3: Fluctuation of density perturbation in the initial center of the anomaly, compared to sinusoidal oscillation with frequency N

5 Outlook

The project can be extended in several directions. First, tests should be run with initial conditions different to the reference state. Something else to be tested is the type of extrapolation suitable for the lower boundary condition. Constant instead of linear extrapolation might help to reduce the spurious oscillations at the ground.

Then, it would be nice to have a faster version than the presented fully explicit finite volume scheme. Possible candidates are:

- FD/FV Split Explicit Scheme: The presented and implemented split explicit scheme can be debugged and tested.
- FV Implicit Scheme: With the current implementation, it would be easy to try a scheme of the form $\boldsymbol{r}^{n+1} = \boldsymbol{r}^n + \Delta t \ \boldsymbol{R}(\boldsymbol{r}^{n+1})$ or similar. An iterative approach using $\boldsymbol{r}^n$ as first guess might allow for a much larger Δt.

Furthermore, in order to obtain a fully conservative scheme, the following modifications are needed:

- FV Split Explicit Scheme: A conservative formulation of the small timestep has to be found.
- Conservative Damping: The diffusive sponge and divergence damping terms have to be formulated conservatively.

Finally, one would like to simulate topograpies higher than Δz. To this end, the finite volume terms have to be generalized to interfaces having any direction, and a different grid has to be implemented. Interesting candidates are terrain following gridlines or shaved cells. This change would allow to use the true nonlinear lower boundary condition. That could also help to remove the spurious oscillations observed in the numerical experiments.

A Parameter Files

A.1 Hydrostatic Case

```
! parameter file for use with param_m
!
! format specification
! --------------------
!
! lines starting with !, # or / are ignored, as well as blank lines
! valid key-value pairs look like:
!   <key> = <value>
! or:
!   <key> = <value> ! <comment> (/ or # allowed too)
! all values are interpreted as reals
! -> be sure to include a dot even for later integers

! topography
topo_width  = 10000.
topo_height = 1.

! grid
x_min = -100000.
x_max = 100000.
n_x   = 100.
z_min = 0.
z_max = 24000.
n_z   = 80.

! reference state
theta_0 = 280.
p_0     = 100000.
N_0_2   = 0.0001
gamma   = 1.4
```

```
R       = 287.05
g       = 9.81
u_0     = 10.

! sponge diffusion
diff   = 0. ! dimensionless, max is 0.125
z_diff = 14100.

! rayleigh relaxation
relax   = 0.01 ! dimensionless
z_relax = 14100.

! timestepping
t_final  = 25000.
timestep = 0.4

! writing stride
! t_stride must divide t_final
! t_stride must be divisible by timestep
t_stride = 250.

! asselin filter coefficient
filter_coeff = 0.05 ! dimensionless, max is 0.25

! density anomaly
a_anomaly =     0. ! -0.001 ! relative amplitude
sigma_x   = 10000. ! in m
sigma_z   =  1000. ! in m
z_anomaly =  8000. ! in m

! divergence damping
a_x = 1000. ! in m^2 s^-1
a_z = 1000. ! in m^2 s^-1
```

A.2 Nonhydrostatic Case

```
! parameter file for use with param_m
!
! format specification
! --------------------
!
! lines starting with !, # or / are ignored, as well as blank lines
! valid key-value pairs look like:
!   <key> = <value>
! or:
!   <key> = <value> ! <comment> (/ or # allowed too)
! all values are interpreted as reals
! -> be sure to include a dot even for later integers

! topography
topo_width  = 1000.
topo_height = 1.

! grid
x_min = -20000.
x_max = 20000.
n_x   = 100.
z_min = 0.
z_max = 24000.
n_z   = 80.

! reference state
theta_0 = 280.
p_0     = 100000.
N_0_2   = 0.0001
gamma   = 1.4
R       = 287.05
g       = 9.81
u_0     = 10.

! sponge diffusion
diff   = 0. ! dimensionless, max is 0.125
z_diff = 14100.

! rayleigh relaxation
relax   = 0.01 ! dimensionless
```

```
z_relax = 14100.

! timestepping
t_final  = 10000.
timestep = 0.4

! writing stride
! t_stride must divide t_final
! t_stride must be divisible by timestep
t_stride = 100.

! asselin filter coefficient
filter_coeff = 0.05 ! dimensionless, max is 0.25

! density anomaly
a_anomaly =     0. ! -0.001 ! relative amplitude
sigma_x   = 10000. ! in m
sigma_z   =  1000. ! in m
z_anomaly =  8000. ! in m

! divergence damping
a_x = 1000. ! in m^2 s^-1
a_z = 1000. ! in m^2 s^-1
```

A.3 Buoyant Bubble

```
! parameter file for use with param_m
!
! format specification
! --------------------
!
! lines starting with !, # or / are ignored, as well as blank lines
! valid key-value pairs look like:
!   <key> = <value>
! or:
!   <key> = <value> ! <comment> (/ or # allowed too)
! all values are interpreted as reals
! -> be sure to include a dot even for later integers

! topography
topo_width  = 10000.
topo_height = 0.

! grid
x_min = -15000.
x_max = 15000.
n_x   = 100.
z_min = 0.
z_max = 24000.
n_z   = 80.

! reference state
theta_0 = 280.
p_0     = 100000.
N_0_2   = 0.0001
gamma   = 1.4
R       = 287.05
g       = 9.81
u_0     = 0.

! sponge diffusion
diff   = 0. ! dimensionless, max is 0.125
z_diff = 14100.

! rayleigh relaxation
relax   = 0.01 ! dimensionless
```

```
z_relax = 14100.

! timestepping
t_final  = 3600.
timestep = 0.4

! writing stride
! t_stride must divide t_final
! t_stride must be divisible by timestep
t_stride = 30.

! asselin filter coefficient
filter_coeff = 0.05 ! dimensionless, max is 0.25

! density anomaly
a_anomaly = -0.001 ! relative amplitude
sigma_x   =  1000. ! in m
sigma_z   =  1000. ! in m
z_anomaly =  7050. ! in m

! divergence damping
a_x = 1000. ! in m^2 s^-1
a_z = 1000. ! in m^2 s^-1
```

Bibliography

[1] N. Botta, R. Klein, S. Langenberg, S. Lützenkirchen: Well balanced finite volume methods for nearly hydrostatic flows. submitted to J. Comp. Phys. (2002)

[2] V. K. Decyk, C. D. Norton, B. K. Szymanski: How to Express C++ Concepts in Fortran90. `http://exodus.physics.ucla.edu/Fortran95/ExpressC++.pdf`

[3] V. K. Decyk, C. D. Norton, B. K. Szymanski: Introduction to Object-Oriented Concepts using Fortran90. `http://exodus.physics.ucla.edu/Fortran95/F90_Objects.pdf`

[4] D. R. Durran: Numerical Methods for Wave Equations in Geophysical Fluid Dynamics. Springer, New York, 1999.

[5] J. B. Klemp, W. C. Skamarock, J. Dudhia: Conservative Split-Explicit Time Integration Methods for the Compressible Nonhydrostatic Equations. `http://www.mmm.ucar.edu/individual/skamarock/wrf_equations_eulerian.pdf`, 2000.

[6] J. B. Klemp, R. B. Wilhelmson: The Simulation of Three-Dimensional Convective Storm Dynamics. J. Atmos. Sci. 35 (1978), 1070-1096.

[7] H. R. Schwarz: Numerische Mathematik. 4th edition, Teubner, Stuttgart (1997)

[8] M. Xue, K. K. Droegemeier, V. Wong: The Advanced Regional Prediction System (ARPS) – A multi-scale nonhydrostatic atmospheric simulation and prediction model. Part I: Model dynamics and verification. Meteorol. Atmos. Phys. 75 (2000), 161-193.