CEE263B Final Project Report

A basic hydrostatic atmospheric numerical model

Yun Zhang

SUID: 05729191

CEE263B Final Project Report:

A basic hydrostatic atmospheric numerical model

Yun Zhang SUID: 05729191

1. Introduction

As an important tool to simulate and understand various atmospheric phenomena from microscale to global scale, an atmospheric numerical model resolves atmospheric motions, moist processes, heat exchange and other related phenomena. Basically, atmospheric numerical models have four types which are thermotropic, barotropic, hydrostatic, and nonhydrostatic. A hydrostatic atmospheric numerical model assumes vertical acceleration is negligible compared to vertical pressure gradient, and replaces the vertical momentum equation with the hydrostatic approximation which filters out vertically acoustic waves and releases the strict time step limit caused by the sound travel speed. The hydrostatic models have been successfully applied to simulate and predict global and regional weather variations especially with the horizontal resolution as around 10km.

In this final project, a hydrostatic model in spherical-sigma-pressure coordinate with complete Fortran90-based program packages has been proposed and tested by several applications including the simulations of atmospheric motions and passive gas migration under different scenarios. This report will discuss the process to develop a basic hydrostatic atmospheric numerical model from the perspective of equations of atmospheric motions and discretization scheme used to solve equations numerically (Chapter 2 & 3), and further provide a brief introduction to the coding packages of this final project (Chapter 4). The model applications (Chapter 5) validates the ability of this numerical model and indicates its limitations for the modeling of atmospheric motions. The conclusion and discussion about the future applications and improvements of the proposed model will be discussed in Chapter 6.

2. Equations of motion

The proposed hydrostatic atmospheric model solves the set of primitive equations

which include the continuity equation, the thermodynamic equation and the equation of momentum conservation. The flux forms of the three-dimensional primitive equations with the hydrostatic approximation in spherical-sigma-pressure coordinate are given:

The continuity equation:

$$R_e^2 cos\varphi \frac{\partial \pi_a}{\partial t} + \frac{\partial}{\partial \lambda_e}(u\pi_a R_e) + \frac{\partial}{\partial \varphi}(v\pi_a R_e \cos\varphi) + \pi_a R_e^2 cos\varphi \frac{\partial \dot{\sigma}}{\partial \sigma} = 0 \quad (2.1)$$

The thermodynamic equation:

$$R_e^2 cos\varphi \frac{\partial(\pi_a \theta_v)}{\partial t} + \frac{\partial}{\partial \lambda_e}(u\pi_a \theta_v R_e) + \frac{\partial}{\partial \varphi}(v\pi_a \theta_v R_e \cos\varphi) + \pi_a R_e^2 cos\varphi \frac{\partial(\dot{\sigma}\theta_v)}{\partial \sigma} = \pi_a R_e^2 cos\varphi \left[\frac{(\nabla \cdot \rho_a K \nabla)\theta_v}{\rho_a} + \frac{\theta_v}{c_{p,d} T_v} \sum_{n=1}^{N_{e,t}} R_n\right] \quad (2.2)$$

The momentum conservation equations:

$$R_e^2 cos\varphi \frac{\partial(\pi_a u)}{\partial t} + \frac{\partial}{\partial \lambda_e}(u^2 \pi_a R_e) + \frac{\partial}{\partial \varphi}(uv\pi_a R_e \cos\varphi) + \pi_a R_e^2 cos\varphi \frac{\partial(\dot{\sigma}u)}{\partial \sigma} = \pi_a uv R_e sin\varphi + \pi_a f v R_e^2 cos\varphi - R_e\left(\pi_a \frac{\partial \Phi}{\partial \lambda_e} + \sigma c_{p,d}\theta_v \frac{\partial P}{\partial \sigma}\frac{\partial \pi_a}{\partial \lambda_e}\right) + \frac{\pi_a}{\rho_a} R_e^2 cos\varphi (\nabla \cdot \rho_a K_m \nabla) u \quad (2.3)$$

$$R_e^2 cos\varphi \frac{\partial(\pi_a v)}{\partial t} + \frac{\partial}{\partial \lambda_e}(uv\pi_a R_e) + \frac{\partial}{\partial \varphi}(v^2 \pi_a R_e \cos\varphi) + \pi_a R_e^2 cos\varphi \frac{\partial(\dot{\sigma}v)}{\partial \sigma} = -\pi_a u^2 R_e sin\varphi - \pi_a f u R_e^2 cos\varphi - R_e cos\varphi\left(\pi_a \frac{\partial \Phi}{\partial \varphi} + \sigma c_{p,d}\theta_v \frac{\partial P}{\partial \sigma}\frac{\partial \pi_a}{\partial \varphi}\right) + \frac{\pi_a}{\rho_a} R_e^2 cos\varphi (\nabla \cdot \rho_a K_m \nabla) v \quad (2.4)$$

The explanation of all the symbols are provided in Appendix A. This set of equations above provides the main structure of this hydrostatic model and resolves the velocity, pressure and temperature field of atmospheric motions. By assuming the top and bottom vertical velocity as zero, integrating the continuity equation (2.1) yields the vertically averaged continuity equation to calculate air column pressure directly from horizontal velocity field:

The vertically averaged continuity equation:

$$R_e^2 cos\varphi \frac{\partial \pi_a}{\partial t} + \int_0^1 \left[\frac{\partial}{\partial \lambda_e}(u\pi_a R_e) + \frac{\partial}{\partial \varphi}(v\pi_a R_e \cos\varphi)\right] d\sigma = 0 \quad (2.5)$$

In addition, in order to simulate the processes of the transport and migration of aerosol particles, specific humidity (water vapor particle) and other passive gas

concentration are calculated by the flux form of the species continuity equation which is given by:

The species continuity equation:

$$R_e^2 cos\varphi \frac{\partial(\pi_a q)}{\partial t} + \frac{\partial}{\partial \lambda_e}(u\pi_a q R_e) + \frac{\partial}{\partial \varphi}(v\pi_a q R_e \cos\varphi) + \pi_a R_e^2 cos\varphi \frac{\partial(\dot{\sigma} q)}{\partial \sigma} = \pi_a R_e^2 cos\varphi \left[\frac{(\nabla \cdot \rho_a K \nabla) q}{\rho_a} + \sum_{n=1}^{N_{e,t}} R_n\right] \quad (2.6)$$

in which q can represent the concentration of any gas or aerosol particle. The process to discretize and numerically solve these equations by the Fortran-based program packages will be discussed in details in the following chapters.

3. Discretization scheme and boundary conditions

3.1 Time-stepping scheme

The proposed hydrostatic atmospheric model applies two time-stepping schemes, including Backward Euler scheme (default) and Matsuno scheme, to solve the set of primitive equations. Although these two schemes are both 1st order accurate and conditionally stable, Matsuno scheme tends to provide relatively more stable results under the same time step size according to the model tests (Chapter 5). The extra stability of Matsuno scheme may be explained by a simple model. Assume the equation we want to solve is given:

$$\frac{du}{dt} = \lambda u \quad (3.1)$$

Applying Backward Euler and Matsuno scheme to discretize this equation yields:

Backward Euler Scheme:

$$u^{n+1} = (1 + \lambda h)u^n \quad (3.2)$$

Matsuno Scheme:

$$u^{n+1} = (1 + \lambda h + \lambda^2 h^2)u^n \quad (3.3)$$

Taylor Expansion:

$$u^{n+1} = \left(1 + \lambda h + \frac{\lambda^2 h^2}{2}\right)u^n + o(h^3) \quad (3.4)$$

Meanwhile, based on the Taylor Expansion for u^{n+1}, the leading term of numerical error for each scheme is given:

Backward Euler Scheme:

$$Error = -\frac{\lambda^2 h^2}{2} u^n + o(h^3) \tag{3.5}$$

Matsuno Scheme:

$$Error = \frac{\lambda^2 h^2}{2} u^n + o(h^3) \tag{3.6}$$

Although both Backward Euler and Matsuno Scheme have a 2nd-order leading error term which validates the 1st-order accuracy in time, Matsuno Scheme introduces positive numerical diffusion which may stabilize simulation results with the same time step size.

Another widely-used time-stepping scheme is the leapfrog scheme which is not applied in the proposed model. The leapfrog scheme provides 2nd-order accuracy and prevents numerical diffusion and amplitude loss. However, because this scheme induces numerical dispersion which may ruin numerical stability, the leapfrog scheme needs to be applied together with Matsuno Scheme which reduces the time accuracy into 1st-order accuracy. This combining scheme may be developed into the program packages in future developments

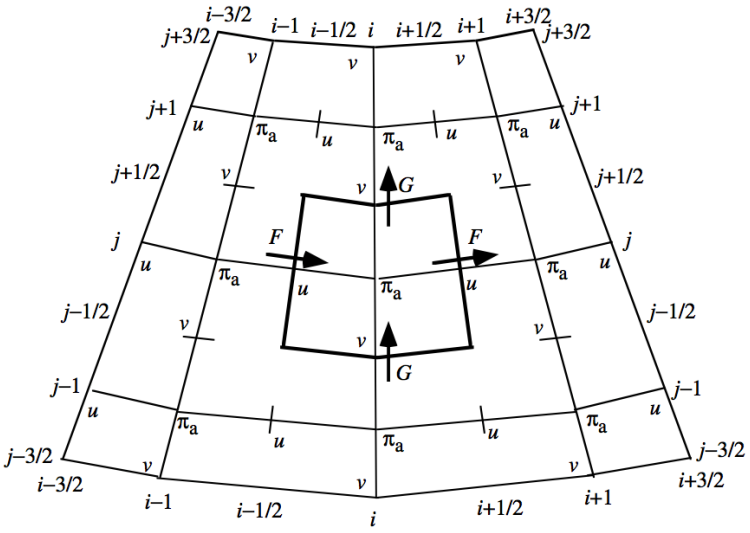

Figure 3.1 The Arakawa C-grid. This figure shows the relative locations of column pressure (π_a) and horizontal velocity (u, v). The bold black box represents a regular computational cell in this grid. (Same picture as on pp209 in the text book)

3.2 Grid and spatial discretization scheme

As shown by Fig. 3.1, the structured Arakawa C-grid (Arakawa and Lamb, 1977) is applied into the proposed model for spatially discretizing the set of primitive equations. In this grid, column pressure is located at the center of each cell, while horizontal velocity and vertical velocity are defined at the center of each boundary and the center of bottom and top faces of each cell. Applying a C-grid ensures a central differencing scheme for spatial discretization which provides 2nd-order accuracy in space.

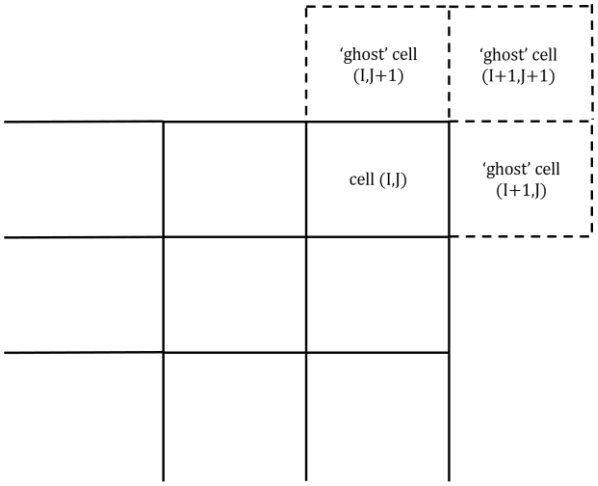

Figure 3.2 Basic setup for 'ghost-point' boundary condition. The figure shows the 'ghost' cells for the boundary cell located at the northeast corner of simulation domain. The velocity, temperature, pressure and other variables of 'ghost' cells will be decided by cell (I, J). I and J represent the number of cells along east-west and north-south directions

3.3 Boundary condition

The proposed model applies three types of boundary conditions including specified boundary condition, 'ghost-point' boundary condition (default) and periodic boundary condition. The specified boundary condition specifies velocity, temperature, pressure and other variables at the domain boundaries by the known values or functions. On the other hand, the 'ghost-point' boundary condition (Figure 3.2) supposes there is a 'ghost' cell outside each boundary and assigns values of all variables in the 'ghost'

cell as the nearest inner cell within simulation domain. Periodic boundary condition can be treated as a special ‘ghost-point’ boundary condition which assumes the values of all variables at ‘ghost’ cell are determined by the boundary cell in the opposite boundary of simulation domain. For example, shown by Figure 3.2, variables at ‘ghost’ cell (I,J+1) will be determined by cell (I,J) for ‘ghost-point’ boundary condition and by cell (I,1) for periodic boundary condition. More tests on periodic boundary condition are shown in Chapter 5.

4. Introduction to Fortran-based program packages

4.1 Program components

The Fortran90-based program of this final project follows the objective oriented programming approach and completely organized. It consists of 11 components including one main function file, nine individual functional modules and a MATLAB-based library for the visualization of all results. Different modules have several different subroutines to fulfill functionality. Separate descriptions and comments on the usage of each subroutine has been noted to increase program readability. The functionality for different components are given below:

(1) **regionalmodel.f90**: The main function for this proposed model which calls different subroutines from other functional modules and includes the main loop of time stepping process to calculate the set of primitive equations.

(2) **constant_parameter.f90**: This module stores all the constant parameters for both numerical simulation setup and physical constants.

(3) **allocate_variable.f90**: This module allocates spaces for all the global variables which stores pressure, velocity, temperature and other variables

(4) **initialization.f90**: This module specify the initial condition for all variables

(5) **boundary.f90**: This module specify the boundary condition if specified boundary condition is applied in the simulation

(6) **basic_state.f90**: This modules interpolates the values of output from input variable based on the table for the variation of gravitational acceleration, air pressure, air temperature, and air density with altitude in a standard atmosphere

(same as Table B.1 on pp714 in the text book)

(7) **source.f90**: This modules calculates the source term for the transport of heat, specific humidity and other aerosol particles.

(8) **turbulence.f90**: This module simulates turbulence process and updates eddy viscosity and eddy diffusivity every time step (not finished).

(9) **phys.f90**: This module includes all the subroutines to calculate the set of primitive equations and other physical processes like the calculation of air density.

(10) **output.f90**: This module includes all the subroutines to output all the results. The output format include binary and regular txt files.

(11) **MATLAB-library**: MATLAB functions to read and plot 2D and 3D results based on the format of the output module.

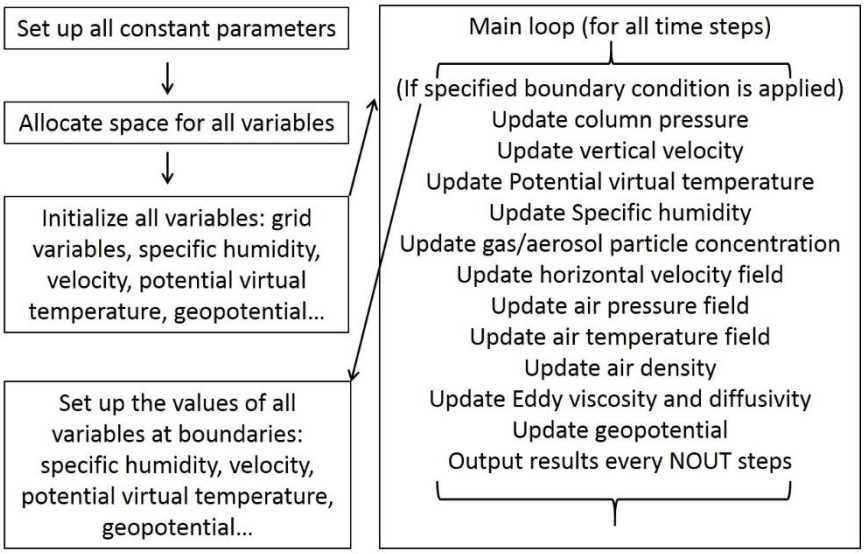

Figure 4.1 The workflow chart of the Fortran90-based program packages for the proposed hydrostatic atmospheric model. This figure shows the basic setup for the main function in regionalmodel.f90. NOUT represents how often to output simulation results.

4.2 How program works

As shown by Fig. 4.2, the workflow of this program basically include five parts which are the setup of constant parameter, the space allocation for all variables, the setup of initial condition, the main loop and the setup of boundary condition. In order to modify

this program to simulate different scenarios, a pre-setup process to only adjust the constant parameters, initial and boundary conditions is required. All the required pre-setup subroutines and parameters are located in three separate files which are constant_parameter.f90, initialization.f90 and boundary.f90. The calculation within the main loop has been optimized by applying vectorized computation to reduce the number of do-loop which may increase computational latency.

4.3 Switches

For the numerical setup, the functions of several switches which will improve the applicability of this program will be given below (modify all switches by editing constant_parameter.f90):

(1) qvmodel: whether to consider the transport of specify humidity (1) or not (0)
(2) PVTmodel: whether to consider the transport of potential virtual temperature (1) or not (0)
(3) gasmodel: whether to consider the transport of passive gas and aerosol particle (1) or not (0)
(4) turbmodel: whether to consider turbulence model (1) or not (0)
(5) outputswitch: whether to output results (1) or not (0)
(6) periodicBC: whether to apply periodic boundary condition (1) or not (0)
(7) matsuno: whether to apply Matsuno time-stepping scheme (1) or not (0)
(8) coriolis: whether to consider coriolis effects (1) or not (0)

4.4 How to run and download the program

To run this program is simple by three basic commands in its Makefile:

(1) make: compile all the .f90 files to a general executive program
(2) make test: run the program
(3) make clean: clean the output results and the executive program

The latest version of this code can be downloaded from github.com by the following command:

git clone https://github.com/zyaj/atomsphere-regional-model.git

The latest version may be released to public after June 8th.

5. Model test and application

The proposed hydrostatic atmospheric model has been tested and applied to simulate atmospheric motions under different scenarios to discuss its accuracy, stability and applicability. All the tests and applications are applied on the same numerical grid with the parameters shown by Table 5.1. This grid has a symmetric square shape with a center at the Equator where Coriolis effects is not significant in momentum balance. The symmetry of this computation mesh also ensures the rationality to apply periodic boundary condition.

Table 5.1 Parameter of computational grid for the model tests and applications

φ_0 (Latitude of the origin at southwest corner)	-1^o	λ_0 (Longitude of origin at southwest corner)	-1^o
Nx (Number of cells in east-west direction)	40	Ny (Number of cell in north-south direction)	40
Nk (Number of vertical layers)	15	$d\varphi$ (grid size)	0.05^o
$d\lambda$ (grid size)	0.05^o	R_e (Earth radius)	6371km

5.1 Time accuracy and stability

In order to discuss the time accuracy of Backward Euler and Matsuno schemes, a simple test case has been run for several times with different time step size under different time-stepping schemes to illustrate how the variation of time step size affect simulation results. The test case applies the 'ghost-point' boundary conditions for all variables and the computation mesh shown by Table 5.1, initializes the column pressure field by a constant east-west pressure gradient as shown by Fig. 5.1 and then simulates the atmospheric velocity field (Fig. 5.2). The results of horizontal velocity along east-west direction at grid center after 2mins real-time simulation are extracted

to calculate the relative error which is defined as below:

$$Error(dt) = \left|u_{dt} - u_{dt/2}\right|$$

in which dt means a specific time step, u_{dt} represents the horizontal velocity along east-west direction.

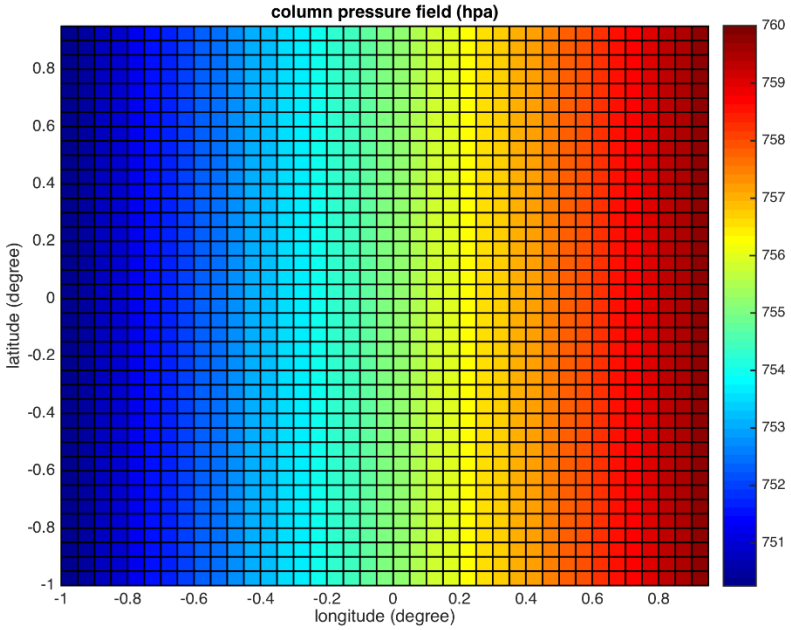

Figure 5.1 The initialized column pressure field (hpa) for the test case to discuss time accuracy and stability

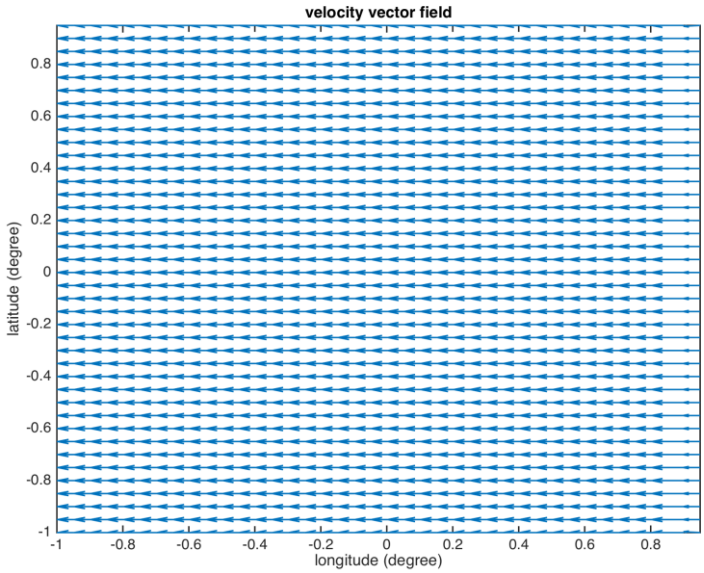

Figure 5.2 The velocity field induced by the column pressure field in Figure 5.1

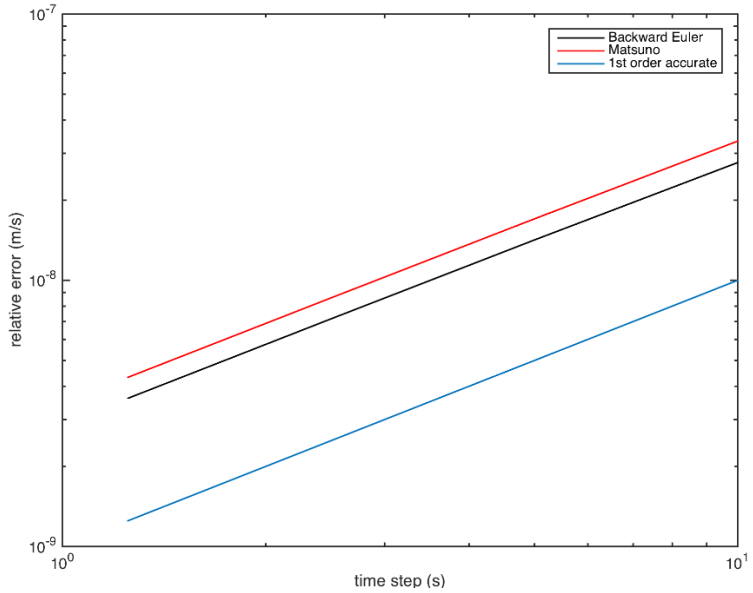

Figure 5.3 Time-accuracy for Backward Euler and Matsuno schemes. This figure indicates 1st order time accuracy for both schemes.

The relationship between time step size and relative error has been shown by Fig. 5.4 which further indicates 1st order time accuracy for both schemes and validates the previous discussion in Chapter 3.1. Moreover, the stability of these two schemes has been illustrated by the comparison of the number of steps after which the program blows up (Table 5.2). Although both two schemes have the same order of accuracy in time, Matsuno scheme is more stable compared to Backward Euler scheme under the same time step size due to the additional numerical diffusion to decrease velocity amplitude. For example, when time step size is less than 5 seconds, the program with Matsuno scheme keeps stable for more than 1000 time steps under the test case scenario, which it blows up after 115 time steps if Backward Euler scheme is applied.

Table 5.2 The number of time step after which program blows up for the test case

Time step (s)	5	10	15
Backward Euler scheme	115	49	20
Matsuno scheme	>1000	65	24

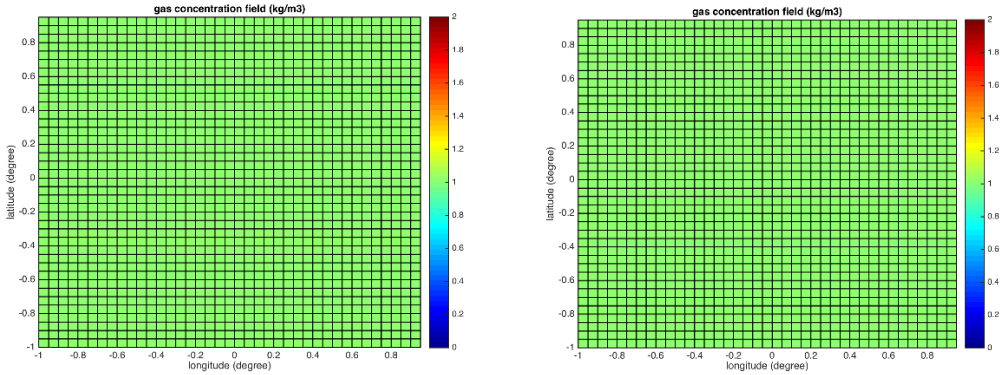

Figure 5.4 Comparison of the gas concentration field (kg/m^3) for initial condition (left) and results after 1000 time steps with 5 seconds time step size (right). This figure indicates gas concentration keeps constant under atmospheric motions (as shown in Figure 5.2) and further validates that the proposed model is mass-conserved.

5.2 Mass conservation check

Beyond numerical accuracy and stability, whether the proposed model is mass-conserved for the continuity requirement and scalar transport is another significant requirement to test its applicability. A test case to validate this requirement for the proposed model has been designed by initializing the temperature field for the whole simulation domain with a constant value to check whether this value keeps unvaried after several time steps. The test case keeps applying the same initial column pressure field (Fig. 5.1) which induces east-west horizontal velocity, while the passive gas concentration field is initialized as 1kg/m^3 everywhere in the domain. The comparison between the gas concentration field for initial condition and results after 1000 time steps with 5 seconds time step size (Figure 5.3) indicates that gas concentration maintains constant as 1 under atmospheric motions, and further validates that the proposed model is mass-conserved and applicable to calculate the migration of potential virtual temperature, specific humidity and other passive aerosol particles.

5.3 Atmospheric motions under pressure gradient

The proposed model has been applied to simulate atmospheric motions under pressure gradient caused by a pressure peak at the center of the whole domain (Fig. 5.5, left).

The computational mesh is described by Table 5.1, while Matsuno time-stepping scheme with a 5-second time step size is chosen for this test case. The comparison between the initial condition and simulation results after 0.7hr (500 time steps) has been shown by Figure 5.5 to 5.7. The results suggest that the column pressure peak at the center dies out after 0.7hr while the remaining temperature gradient dominates the velocity field. This test case successfully validates the capability of the proposed model to simulate how atmospheric motions evolve under pressure gradient.

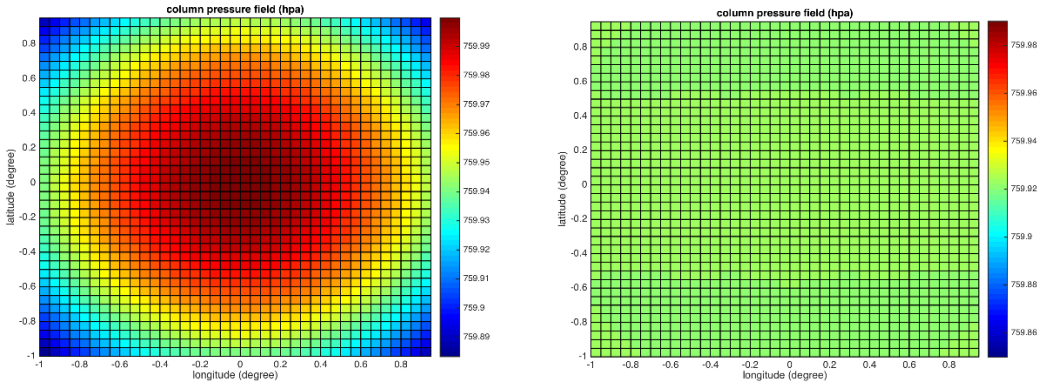

Figure 5.5 Column pressure field (hpa) for initial condition (left) and simulation results after 0.8 hour real time (right). This figure indicates the initial column pressure gradient dies out after 0.7 hour.

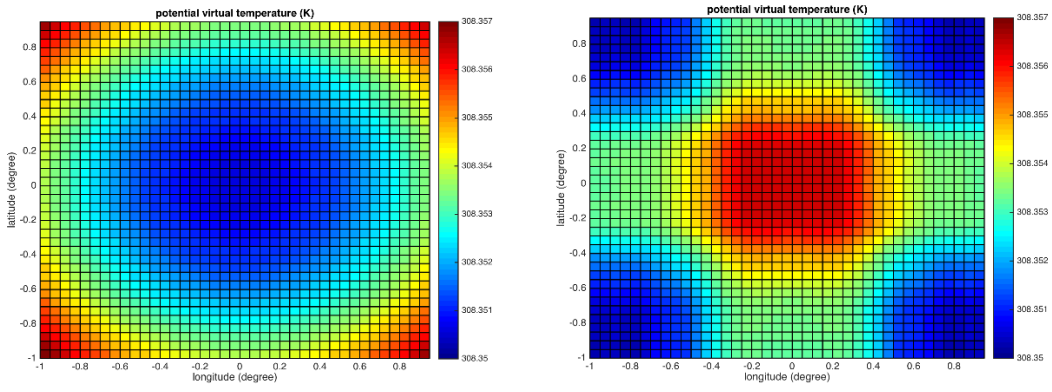

Figure 5.6 Potential virtual temperature field (K) for initial condition (left) and simulation results after 0.7 hour real time (right).

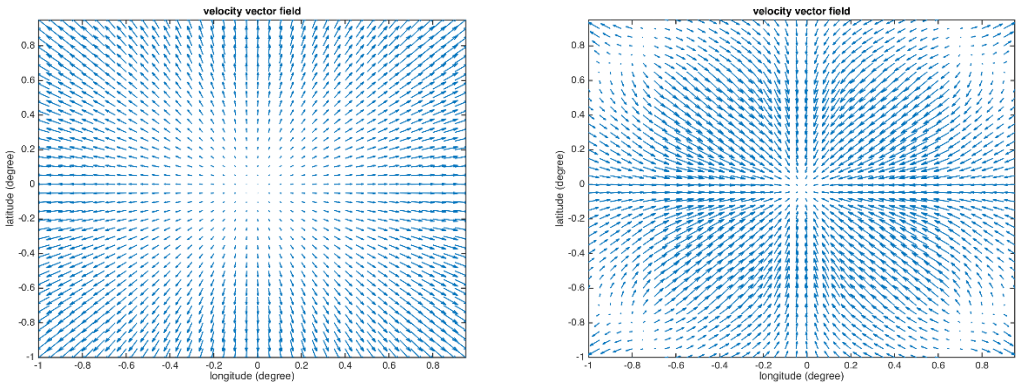

Figure 5.7 Velocity vector field for initial condition (left) and simulation results after 0.7 hour real time (right). This figure shows that velocity field after 0.7 hour is dominated by the gradient of potential virtual temperature.

5.4 Aerosol particle transport under atmosphere motions

The proposed model is capable to simulate aerosol particle transport under atmosphere motions. By initializing the concentration of a passive gas as constant 1 at the center of the simulation domain, the simulated aerosol particle migration under eastward and center-peak pressure gradient (Fig. 5.1 & 5.5 left) has been shown by Figure 5.8 and 5.9. The simulated results clearly indicate that the west wind caused by the eastward pressure gradient induces a downwind transport for the passive gas, while the pressure peak at the center pushes passive gas particle to farther locations.

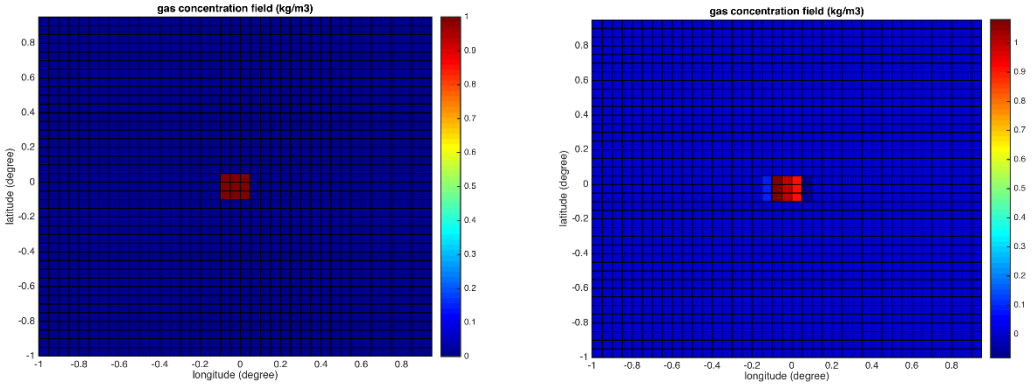

Figure 5.8 Passive gas concentration field (kg/m^3) for the initial condition (left) and simulation results after 0.7 hour real time (right). This figure indicates a downwind transport of passive gas. The initial pressure and velocity field for this test case is shown by Figure 5.1 and 5.2

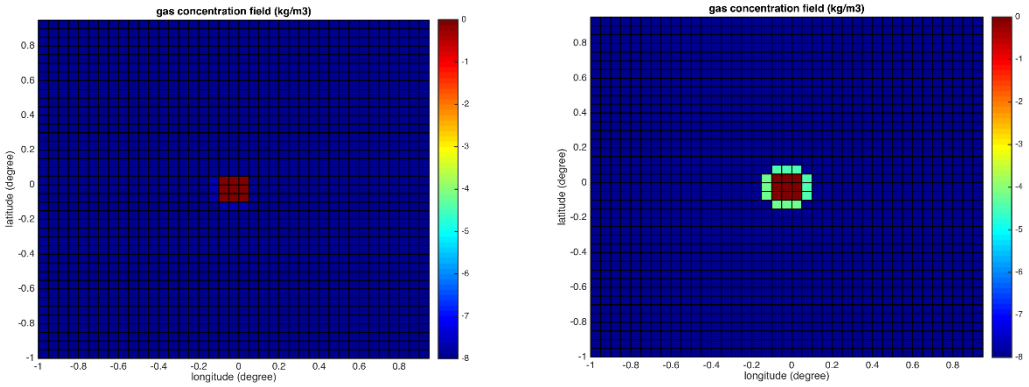

Figure 5.9 Passive gas concentration field in log-scale (log(kg/m^3)) for the initial condition (left) and simulation results after 0.7 hour real time (right). This figure indicates that passive gas migrates to farther locations from the center. The pressure, potential virtual temperature and velocity fields of this test case are shown by Figure 5.5 to 5.7.

6. Conclusion and future development

In this final project, a hydrostatic atmospheric model which solves the set of primitive equations under hydrostatic approximation has been proposed to simulate atmospheric motions under different scenarios. According to the proposed model, a Fortran90-based program has been developed and tested by the different test cases to validate its capability from various aspects including numerical accuracy, numerical stability and model applicability. The discussion on model application further ensures its feasibility on the simulation of atmospheric pressure, wind velocity, air temperature and aerosol particle transport under different dynamic conditions. However, due to the limited time, the proposed model still has some limitations to be optimized in the future. First, since the proposed model ignores boundary-layer and surface processes, it cannot be applied to simulate atmospheric motions when bottom shear become non-negligible. A turbulence model which calculates the correct eddy viscosity and diffusivity is required to ensure both momentum balance and scalar transport. Moreover, the proposed model can only apply structured grid which limits the shape of simulation domain as square and is not suitable for the simulation for an arbitrary domain shape.

Therefore the compatibility of orthogonal unstructured grids may be another aspect to improve the applicability of the proposed model.

Appendix A: symbol explanation

R_e: diameter of the Earth

t: time

φ: latitude

λ_e: longitude

σ: vertical coordinate

P: air pressure

π_a: air column pressure

θ_v: air potential virtual temperature

$c_{p,d}$: specific heat of dry air

T_v: air virtual temperature

u: East/West velocity

v: North/South velocity

$\dot{\sigma}$: vertical velocity in sigma coordinate

ρ_a: air density

K: eddy diffusivity

K_m: eddy viscosity

Φ: geopotential

q: gas/ aerosol particle concentration

R_n: source term for scalar transport