Application of Grid Refinement to Improve 2D Atmospheric Boundary Layer Software

student: W.A.H. Slump
First supervisor: dr. ir. R. Luppes
Second assessor: dr. A.E. Sterk

Bachelor project Applied Mathematics

university of groningen

faculty of science and engineering
Rijksuniversiteit Groningen
Netherlands
july, 2021

Abstract

This project concerns a numerical model that describes the atmospherical flows above a terrain with different types of vegetation. This numerical model was written in Fortran by dr. ir. R. Luppes in 1993. The aim of the project was to repair errors in the code with the help of mesh-refinement, which is possible because of the significant increase of memory of computers. This means we can do more computations in a shorter amount of time than back in 1993. The project will first consider the original model and its main problem, which is that the solutions of the model change if the top boundary layer changes in height. Then adjustments will be made to try to solve this problem. The main conclusion that grid refinement does not completely solve the main problem. The initial profiles resulting from different heights of the boundary layer do overlap, but the horizontal profiles still show deviation.

Contents

1 Introduction

Nowadays, meteorology is part of everyday life. People check the weather forecast before they leave their house to check what kind of clothes they need to wear or what kind of transportation device they need to use. Therefore it is important that weather forecast models are accurate. These models use simulations of atmospherical flows to predict the weather. Important for these flows is the surface of the earth, and consequently surface flows are important boundary conditions for the atmospherical flows. Surface flows can be calculated using surface characteristics, available energy and atmospherical properties at the surface. The problem that arises is that the atmosperhical properties will adjust to surface changes. For example, the temperature in a forest differs from the temperature on a grassland.

In 1993 research was conducted at Rijksuniversiteit Groningen to investigate the atmospheric boundary layer and the influence of vegetation on this flow [Luppes, 1993]. This resulted in a two-dimensional (2D) simulation software made in Fortran regarding this atmospherical flow. The present study will investigate and use that research in order to improve the simulation. A problem of the old software is that the results changed if upper boundary conditions were implemented at a different height. This means that e.g. the temperature at a height of 20 meters changed if we started computing the solution with a higher top boundary layer, which would be a mistake since the surface does not change and we still compute the same atmospherical flows. Hence, the aim of this project is to improve the 2D simulation software such that the solution does not change when the upper boundary height is changed.

To further improve the software, mesh refinement will be used. Mesh refinement is a method where a more accurate solution is obtained by adding grid points to the grid. In this way the solution is computed for more grid points and thus gets more accurate. For example, if we take $y = x^2$ it is easy to see that if we put more grid points on the x-axis the graph drawn for the solution will approximate the solution better for all x, as shown in figure 1. This also means that with every refinement more computations are made. Because the memory of computers increased significantly since the simulation software was created, mesh refinement is a good method to improve the software.

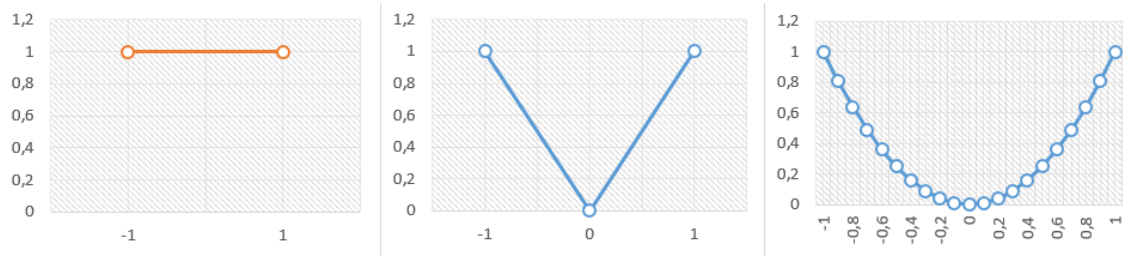


Figure 1: Three graphs of $y = x^2$ with gaining grid points on the x-axis. The left graph uses 2 grid points, the middle graph has 3 grid points and the right graph has 21 grid points.

2 Theory

2.1 General description of the model

The goal of the model is to determine the influence of surface heterogeneity on the fluxes of pressure, velocity, temperature and humidity. We define a heterogeneous zone as a region consisting of multiple homogeneous zones with different surface characteristics. Consider for example a region that consists of a forest in the middle of two zones of grassland, where we define the forest and the grassland as separate homogeneous zones. The model is 2D in the x- and z-direction, we assume that the zones are constant in y-direction. For example, we assume that the forest at $y = 1$ has the same characteristics as the forest at $y = \infty$. The 2D model is described in two versions, a single and multi-layer model. The difference between these models is that the multi-layer model is much more detailed. It describes the surface as multiple layers with different characteristics. In figure 2 the bottom layers of the multi-layer model are illustrated. It shows that for every height layer below the canopy (z_0, y_0 to z_{NF}, y_{NF}), new surface variables are defined. The single-layer describes the surface as a single layer with one set of characteristics. Especially in forests, the multi-layer model is much more accurate since a forest does not have the same amount of vegetation at every height.

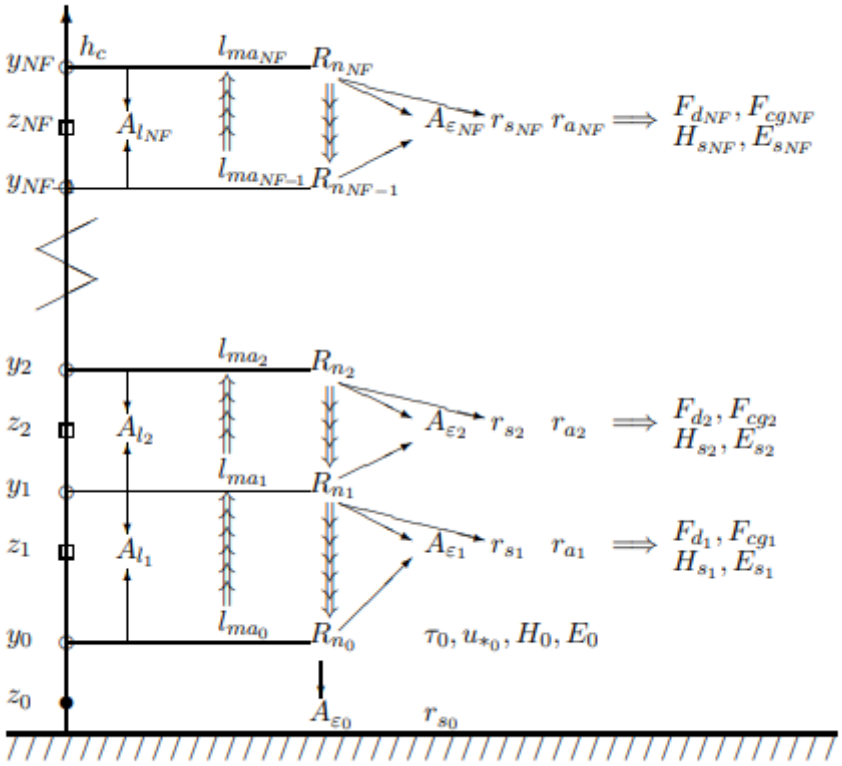


Figure 2: Bottom layers of the multi-layer model. The amount of vegetation is indicated by the leaf-area index (A_l). Other variables are explained in the next sub-sections.

2.2 Summary of equations to solve

For 2-dimensional, stationary incompressible atmospheric boundary layer flow, the coriolis forces can be neglected and incompressibility can be assumed since it is a small scale meteorological model. Because the atmosperical properties do not change in time, a steady state model is used [Luppes, 1993]:

$$u\frac{\partial u}{\partial x} + w\frac{\partial u}{\partial z} + \frac{1}{\varrho}\frac{\partial p}{\partial x} = \frac{1}{\varrho}\frac{\partial \tau}{\partial z} - F_d + F_{cg} \tag{1}$$

$$\frac{\partial p}{\partial z} = 0 \tag{2}$$

$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0 \tag{3}$$

$$u\frac{\partial \theta}{\partial x} + w\frac{\partial \theta}{\partial z} + = \frac{1}{\varrho c_p}(-\frac{\partial H}{\partial z} + H_s) \tag{4}$$

$$u\frac{\partial \theta}{\partial x} + w\frac{\partial \theta}{\partial z} + = \frac{1}{\varrho}(-\frac{\partial E}{\partial z} + E_s) \tag{5}$$

Where u,w are the horizontal and vertical wind velocity, respectively. Furthermore, p is pressure, θ potential temperature, q specific humidity, ϱ air density, c_p specific heat of air at constant pressure, F_d drag force, F_{cg} counter gradient force and H_s, E_s are source terms.
The mathematical descriptions for the fluxes of momentum (τ), sensible heat (H) and latent heat (E) are given by:

$$\tau = \varrho\varepsilon\frac{\partial u}{\partial z} \tag{6}$$

$$H = -\varrho c_p \varepsilon_H \frac{\partial \theta}{\partial z} \tag{7}$$

$$E = -\varrho \varepsilon_H \frac{\partial q}{\partial z} \tag{8}$$

Where ε and ε_H are the eddy viscosities for momentum and heat given by:

$$\varepsilon = (\frac{l}{\Phi_m})^2 |\frac{\partial u}{\partial z}| \tag{9}$$

$$\varepsilon_H = \varepsilon\frac{\Phi_m}{\Phi_h} \tag{10}$$

Where Φ_h and Φ_m are the stability fluxes for heat and momentum and l is the actual mixing length. The arguments of the stability fluxes, the reciprocal of the Monin-Obukhov length in combination with the height ($\frac{z}{L}$) and the Richardson number (Ri), are given by:

$$\frac{z}{L} = \frac{kgz}{\varrho c_p}\frac{\{H(1+0.61q) + 0.61c_p\theta E\}}{\theta_v u_*^3} \tag{11}$$

$$Ri = g(\frac{\partial \theta_v}{\partial z})/(\theta_v\frac{\partial u}{\partial z}) \tag{12}$$

Where k is the Von Karman constant, g the gravity constant and θ_v the virtual potential temperature. Since $\frac{z}{L} \approx Ri$, the choice of which argument is used may depend on the way the model is solved.

To finish of the description of the atmospheric fluxes we have:

$$u_* = \frac{l}{\Phi_m}|\frac{\partial u}{\partial z}| \tag{13}$$

$$|\tau| = \varphi u_*^2 \tag{14}$$

were u_* is the friction velocity.

The mixing length can be calculated from the equation introduced by [Klaassen, 1992]:

$$u\frac{\partial l}{\partial x} + w\frac{\partial l}{\partial z} = uC_l(1 - \frac{l}{l_{ma}}) \tag{15}$$

Where C_l is the rate of adjustment constant and l_{ma} is the adjusted mixing length.

2.3 Surface Fluxes

The equations for the fluxes at the surface can be written as a system of 7 equations with 7 unknowns:

$$u_{*0}|ln(z_1/z_0) + \Psi_m| - |u_1 k| = 0 \tag{16}$$
$$H_0 r_a - \varrho c_p (T_s - T_a) = 0 \tag{17}$$
$$E_0 p(r_a + r_s) - 0.622\varrho(e_s^* - e_a) = 0 \tag{18}$$
$$R_{n_0}(1 - groco) - (\lambda E + H) = 0 \tag{19}$$
$$r_a k u_{*0} - (ln(z_1/z_{0h}) + \Psi_h) = 0 \tag{20}$$
$$\varrho c_p \theta_1 u_{*0}^3 \frac{1}{L_0} + kg(H_0 + 0.61 c_p \theta_1 E_0) = 0 \tag{21}$$
$$R_{n_0} - (1 - \alpha)R_s - \varepsilon(R_l - \sigma T_s^4) = 0 \tag{22}$$

The 7 unknown variables to solve are:

- u_{*_0}: friction velocity
- H_0: sensible heat
- E_0: latent heat
- R_{n_0}: net radiation at the surface
- r_a: air resistance
- $\frac{1}{L_0}$: approximated definition of the reciprocal of the Monin-Obukhov length at the surface, which is the argument of the functions Φ_h,Φ_m together with y_0. Where y_0 is the first intermediate height layer, so between z_0 and z_1.
- T_s: absolute temperature at the surface

The parameters and constants needed to solve these equations are:

- z_1, z_0, z_{0h}: respectively first height layer, roughness length of the canopy and roughness length for heat
- Ψ_h: integrated form of Φ_h
- u_1: horizontal wind speed at z_1
- k: Von Karman constant
- r_s: surface resistance
- T_a: absolute temperature of air at height z_1
- *groco*: ground heat constant
- R_s, R_l: incoming short and long wave radiation
- α: albedo of the surface
- λ: vaporization or latent heat
- σ: Stephan-Boltzman constant
- e_s^*, e_a: saturated vapor pressure at T_s and vapor pressure at z_1

2.4 Boundary conditions

The boundary conditions we use to solve the atmospheric fluxes are the following:

- Bottom layer:

$$u = w = 0 \text{ at } z = z_0 \tag{23}$$
$$\theta = T_s \text{ at } z = 0 \tag{24}$$
$$q = \frac{e^*(T_s)0.622}{p} \text{ at } z = 0 \tag{25}$$

  T_s is the absolute temperature on the surface and z_0 is the roughness length.

- Top layer:
  For the top layer the Interaction law [Veldman, 2010] is used for u:

$$u_e(x) = u_{e_0}(x) + \frac{1}{\pi}\int \frac{dd^*}{d\xi}\frac{d\xi}{x-\xi} \tag{26}$$
$$d^* = y_e u_e - \int_0^{y_e} u dy \tag{27}$$

  Where u_e is the upper horizontal wind speed, u_{e_0} is upper horizontal wind speed for undisturbed flow and y_e is the height of the upper edge. The interaction law describes an approximation of the interaction between the boundary layer flow and the virtual external flow, where the external flow in this case is described as an undisturbed flow.

  For θ and q we used matched load conditions [Klaassen, 1992] written in terms H and E:

$$H(x + \Delta x, y_e) = \frac{H(x, y_e) + H(x, y_e - \Delta z)}{2} \tag{28}$$
$$E(x + \Delta x, y_e) = \frac{E(x, y_e) + E(x, y_e - \Delta z)}{2} \tag{29}$$

  Where Δx and Δz are the stepsizes between grid points in x- and z-direction, respectively. Here H and E at the next horizontal point of calculation are the means of the two upper values of H and E of the present station.

2.5 Initial profiles

The flow equations contain derivatives in the x-direction, so we need initial profiles at $x = 0$ for u, θ, q, and l. These initial profiles can be calculated form the surface fluxes using the following system of conditions at $x = 0$:

$$l = l_{ma} = \frac{kz}{1 + (kz/l_{mm})} \tag{30}$$
$$w = 0 \tag{31}$$
$$u_* = u_{*0} \tag{32}$$
$$H = H_0 \tag{33}$$
$$E = E_0 \tag{34}$$

Where l_{ma} is given by Blackadar's formula, and u_{*0}, H_0 and E_0 follow from the surface flux calculations as given in section 2.3.

2.6 Discretization

As stated in [Luppes, 1993] one-sided backwards discretization methods are used for the horizontal derivatives. We assume u is positive, if u is negative we use a FLARE condition. This will put the convective term $u\partial\phi/\partial x$ (ϕ denotes u, θ, q or l) to zero. The FLARE condition is an appropriate approximation if the absolute value of u is small when u is negative. For vertical derivatives, we use central discretization, since there is no special way in which the data is spread in vertical direction. For the mixing length equation we added an artificial diffusion (CAD) through the application of upwind discretization, because the lack of diffusion terms in this equation may cause stability problems.

The discretization methods that are used are given by [Luppes, 1993][Veldman, 2010]:

Horizontal

$$B_2 : \frac{\partial\phi}{\partial x}|_i = \frac{\phi_i - \phi_{i-1}}{x_i - x_{i-1}} + O(\Delta x); \qquad u > 0$$

$$B_3 : \frac{\partial\phi}{\partial x}|_i = \frac{3\phi_i - 4\phi_{i-1} + \phi_{i-2}}{2(x_i - x_{i-1})} + O(\Delta x^2); \qquad u > 0, \text{equidistant grid}$$

$$\text{FLARE} : u\frac{\partial\phi}{\phi x}|_i = 0; \qquad u < 0$$

Vertical

$$\text{Central } O_1 : \frac{\partial\phi}{\partial z}|_j = \frac{\phi_{j+1} - \phi_{j-1}}{z_{j+1} - z_{j-1}} + O(\Delta z^2)$$

$$\text{Central } O_1, \text{ with CAD} : w\frac{\partial\phi}{\partial z}|_j = w\frac{\phi_{j+1} - \phi_{j-1}}{z_{j+1} - z_{j-1}} - kart + CO_2(\phi) + O(\Delta z^2)$$

$$\text{Central } O_2 : \frac{\partial^2\phi}{\partial z^2}|_j = CO_2(\phi) + O(\Delta z^2)$$

with

$$CO_2(\phi) = \frac{\phi_{j+1}}{\frac{1}{2}(z_{j+1} - z_j)(z_{j+1} - z_{j-1})} - \frac{\phi_j}{\frac{1}{2}(z_{j+1} - z_j)(z_j - z_{j-1})} + \frac{\phi_{j-1}}{\frac{1}{2}(z_j - z_{j-1})(z_{j+1} - z_{j-1})}$$

$$kart = \begin{cases} |w|\frac{z_{j+1} - z_j}{2}, & w > 0 \\ |w|\frac{z_j - z_{j-1}}{2}, & w < 0 \end{cases}$$

3 Existing numerical model

The numerical model stated in [Luppes, 1993] is created in Fortran 77. The program is divided into different subroutines and functions that all interact with each other. From [Luppes, 1993] some schemes are adopted to see in which way subroutines interact with each other.

3.1 Subroutines

Here all subroutines and functions are explained in order of appearance started from the main program.

- MAIN (figure 10)
  This is the most central part of the program. It divides the program into 3 subroutines, INVOER, UITVOER and GRENSLAAGPROCES.
- INVOER
  In this subroutine all the initial and known values are defined. The two-dimensional grid is also defined in this subroutine.
- UITVOER
  This subroutine writes all solutions computed by GRENSLAAGPROCES in data files, that can be used to plot graphs.
- GRENSLAAGPROCES (figure 11)
  The data from INVOER is used here to compute the fluxes. The subroutine uses 2 different other subroutines, namely START to compute the initial profiles and PLAATSPROCES to compute the fluxes till the solutions converge and hence the boundary layer equations (1-5) have been solved.
- OTAGCALC
  This subroutine gives as output the surface dependant variables. These are the variables that depend on the type of surface that is used, and also the constants needed in equations 16-22 are provided. Also variables that depend on the type of model (single- or multi-layer) are calculated here, for example l_{ma}.
- WANDWET
  Computes the surface variables, $u_{*0}, H_0, E_0, R_{n_0}, r_a, \frac{1}{L}_0, T_s$, using equations 16-22.
- START (figure 12)
  The place where the initial profiles are calculated. It uses Subroutine OTAGCALC to search for surface dependant variables and subroutine WANDWET to compute the surface variables while using conditions 30-34.
- E0E1E2
  When the boundary layer equations are discretized a few extra variables are needed, e_0, e_1, e_2 and e_3. These variables are computed in this subroutine.
- PLAATSPROCES (figure 13)
  This subroutine combines several other subroutines to get as output the solution of the boundary layer equations (1-5) for the present iteration. It uses Subroutine E0E1E2 and subroutine OTAGCALC to find/calculate the data that is needed in PLAATSITERATIE, which was not available yet. Then PLAATSITERATIE computes the current solution and SOLMATCALC stores this solution in matrices. Finally PLAATSPREP prepares PLAATSPROCES for the next iteration in GRENSLAAGPROCES.
- PLAATSITERATIE (figure 14)
  Computes the solution of the boundary layer equations (1-5) at a given iteration. It uses five different subroutines to compute all needed variables till they all have converged. These are: UWPSOLVER, QTETASOLVER, WANDWET, MENGLENGTE and MONOBFLUX.
- UWPSOLVER (figure 15)
  Subroutine to solve the fluxes for vertical/horizontal wind velocity (w/u) and pressure (p), equations 1-5.

- QTETASOLVER (figure 16)
  Subroutine to solve the fluxes for temperature (θ) and humidity (q), equations 4-8.
- MENGLENGTE
  Computes the mixing length (l), equation 15.
- MONOBFLUX
  Computes the variables: $\frac{z}{L}, Ri, \tau, H, E$, equations 6-12.

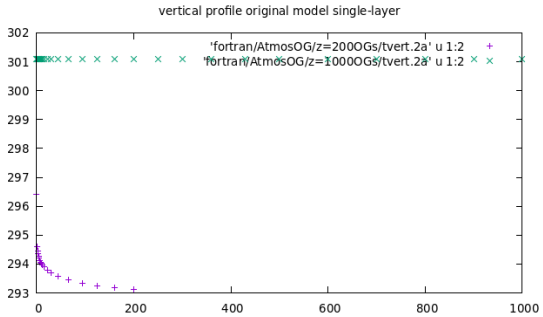
4 Adjustments and Results

4.1 Problem

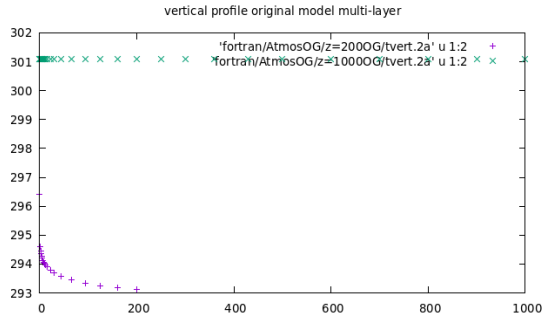
The problem that arises when running the original program is that the solutions change when we move the upper boundary layer, this is seen in figures 3 and 4. Figure 3 shows the difference in initial profiles when the top boundary layer is moved from 200 m (purple line) to 1000 m (green line). It is clear that there are big differences between the profiles, while we expect these profiles to be overlapping. In figure 4 horizontal profiles are given where also the graphs do not align with each other. The errors in the horizontal profiles are partially a consequence of the error in the initial profiles, because these horizontal profiles are created with the initial profiles as starting point. So the first step taken is repairing the initial profiles in such a way that they overlap, independent of the choice of the height of the top boundary layer. The initial profiles are calculated in subroutine START, which uses subroutines INVOER, WANDWET AND OTAGCALC as input for variables and the pressure profile is computed in subroutine SOLMATCALC. Therefore all adjustments made to fix the initial profiles are made in these subroutines. After improving the initial profiles we try mesh refinement to further improve the solution, the grids are defined in INVOER, so all adjustments for mesh refinement are made here.

For all horizontal plots we defined the landscape in x-direction as follows:

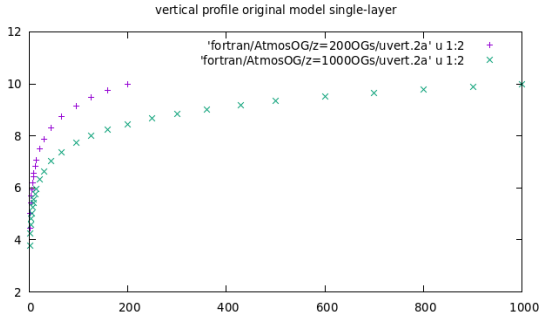
- (1) 0-300 m : single-layer grassland
- (2) 300-600 m: single/multi-layer forest
  Mainly the single-layer variant is used in the plots, because the adjustments made are almost identical for both situations and single-layer took less computation time.
- (3) 300-900 m: single-layer grassland



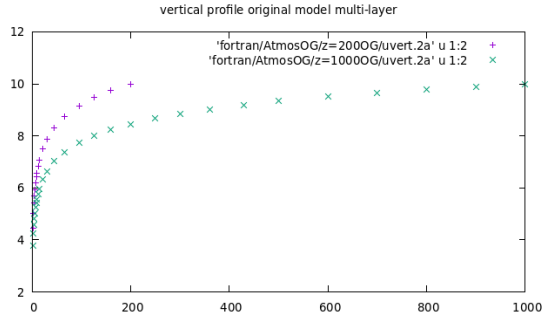
(a) Temparature: top layer at 200 m vs 1000 m



(b) Temparature: top layer at 200 m vs 1000 m

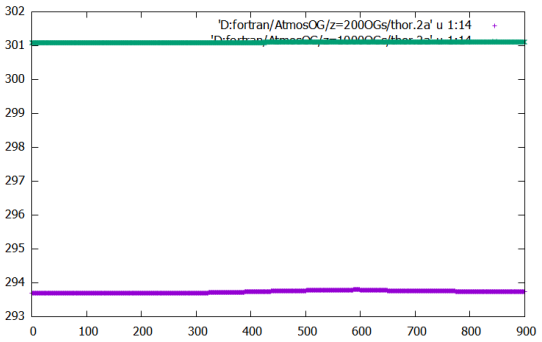


(c) Horizontal wind speed: top layer at 200 m vs 1000 m

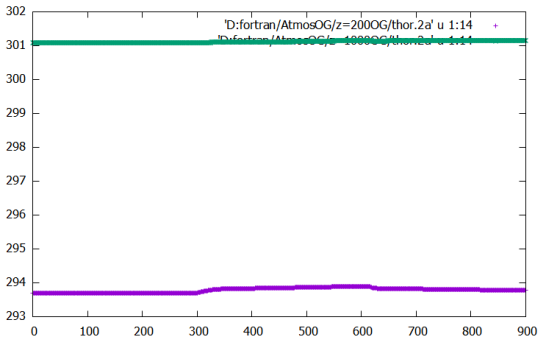


(d) Horizontal wind speed: top layer at 200 m vs 1000 m

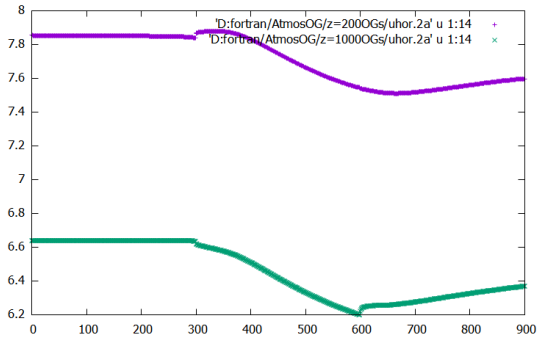
Figure 3: Initial profiles original program; Left graphs single-layer, right graphs multi-layer.



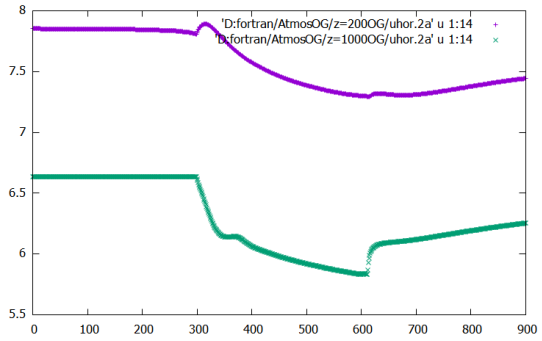
(a) Temparature: top layer at 200 m vs 1000 m



(b) Temparature: top layer at 200 m vs 1000 m



(c) Horizontal wind speed: top layer at 200 m vs 1000 m



(d) Horizontal wind speed: top layer at 200 m vs 1000 m

Figure 4: Horizontal profiles original program; Left graphs single-layer, right graphs multi-layer.

4.2 Initial profiles

The first adjustment made was creating an equidistant grid in the z-direction, because in this way a fixed step size can be used to compute values needed to improve the initial values. The grid in z-direction is defined in the following way:

- For the single-layer model we first define the number of grid points in z-direction, N. Then the step-size between the grid points in z-direction (Δz or DZ) is defined. Finally we use a a loop that generates the height of every grid point. This is done by using the equation:

$$z(i) = z(1) + (i-1)\Delta z \text{ for } i = 1, 2, 3 \dots N, N+1 \tag{35}$$

  Where $z(i)$ is the height at grid point i. The resulting Fortran code can be seen in the appendix in listing 1.

- When the multi-layer method is used, first the grid is defined manually until the height of the canopy is reached. Then the layers above the canopy are defined in a similar way as for the single-layer model:

$$z(i) = z(N_{can}) + (i - N_{can}) \cdot \Delta z \text{ for } i = N_{can}, (N_{can}+1) \dots N, N+1 \tag{36}$$

  Where N_{can} defines the grid point where the canopy ends. The resulting Fortran code can be seen in the appendix in listing 2.

- Then some extra variables are defined, that are needed to fix the initial profile, which are the following:

  - $zstart$
    This is the height where the initial values are known.
  - N_{00}
    The layer that corresponds to the height $zstart$, because we changed the grid in z-direction into an equidistant grid this value is easy to compute. It is defined the following way:

$$N_{00} = \frac{zstart - 1 + \Delta z}{\Delta z} \tag{37}$$

The main idea, used to solve the problem that occurred in the initial profile, was to compute the solution up- and downwards from a certain height where the solutions were known. So we first computed all values and variables needed to compute the solution at height layer N_{00} and then created 2 loops that computed the initial profile. One loop went from N_{00} to 1 and the other loop went from N_{00} to $N + 1$.

4.2.1 Wind speed

Since the vertical wind speed (w) is 0 in the initial profile, we only need to improve the initial profile of the horizontal wind speed (u). In the original model, the initial profile of u is computed using a log-ratio formula that needs the initial wind speed at the top layer (u_s) as input. Since we only know the wind speed for one specific height, we used $zstart$ in the inverted log ratio formula. In this way we could compute the top layer initial value needed to get the correct initial profile. This gave the following equation for u_s:

$$u_s = u_{zstart} \frac{\ln\left(z(N+1)/z_0\right)}{\ln\left(zstart/z_0\right)} \tag{38}$$

Where u_{zstart} is the horizontal wind speed at heigth $zstart$ and z_0 is the roughness length. The resulting Fortran code can be seen in the appendix in listing 3 and the resulting initial profile can be seen in figure 5c. As seen in figure 5c, the initial profiles with boundary layers at different heights do now overlap.

4.2.2 Pressure

First we changed the formula to compute the pressure at the bottom layer (p_{low}) in subroutines START and WANDWET. p_{low} was computed using the pressure (p_s) at height $z(N+1)$ as starting point. This was done in the following way:

$$p_{low} = p_s + \varrho g(z(N+1) - 0) \tag{39}$$

Where ϱ is the air density and g is the gravitation constant. We changed this formula in such a way that we use the pressure (p_{zstart}) at height $zstart$ as starting point:

$$p_{low} = p_{zstart} + \varrho g(z(N_{00}) - 0) \tag{40}$$

To compute the pressure profile we made changes in subroutine SOLMATCALC. In this subroutine the pressure profile was generated by looping over every grid point starting with grid point $N+1$ and ending with grid point 1. Since all starting values are now computed for heigth $zstart$, this one loop is replaced by 2 loops. The first one (loop 1) looping from grid point N_{00} to 1 and the other one (loop 2) looping from grid point $N_{00}+1$ to $N+1$. The formulas used in these loops are defined in the following way:

$$\text{loop 1: } p(i) = p_s + \varrho g(z(N_{00}) - z(i)) \text{ for } i = N_{00}, N_{00} - 1 \ldots, 2, 1 \tag{41}$$

$$\text{loop 2: } p(i) = p_s - \varrho g(z(N_{00}) + z(i)) \text{ for } i = N_{00} + 1, N_{00} + 2 \ldots, N, N + 1 \tag{42}$$

Where $p(i)$ is the pressure at vertical grid point i. The resulting Fortran code can be seen in the appendix in listing 4 and the resulting initial pressure profile is given in figure 5d. As seen in figure 5d, the initial profiles with boundary layers at different heights do now overlap.

4.2.3 Temperature and humidity

To compute the initial profiles of temperature and humidity (θ and q), we first needed to change the formula of T_s, the absolute temperature at the surface. This value is computed for height $z(N+1)$ and we need to change that to $zstart$, because we also changed the definition of t_0, now t_0 is the temperature at height $zstart$. This resulted in the following formula for T_s:

$$T_s = t_0 + (\gamma \cdot zstart) + \frac{0.5 \cdot \gamma^2 \cdot zstart^2}{t_0} \tag{43}$$

Where γ is the psychrometer constant. The resulting Fortran code is seen in the appendix in listing 5.

The initial profiles of temparature and humidity where computed using one loop in subroutine START. That looped from gridpoint $N+1$ to 1. This loop is also changed into 2 loops, loop 1 from N_{00} to 1 and loop 2 from $N_{00}+1$ to $N+1$. This resulted into the following formulas:

$$\text{loop 1: } \text{ for } i = N_{00}, N_{00} - 1 \ldots, 2, 1 \tag{44}$$

$$\theta(i) = \theta(i+1) + ((z(i+1) - z(i)) \cdot \frac{H_0 \Phi_{hi}}{\varrho c_p l_i u_{*0}} \tag{45}$$

$$q(i) = q(i+1) + ((z(i+1) - z(i)) \cdot \frac{E_0 \Phi_{hi}}{\varrho l_i u_{*0}} \tag{46}$$

$$\text{loop 2: } \text{ for } i = N_{00} + 1, N_{00} + 2 \ldots, N, N + 1 \tag{47}$$

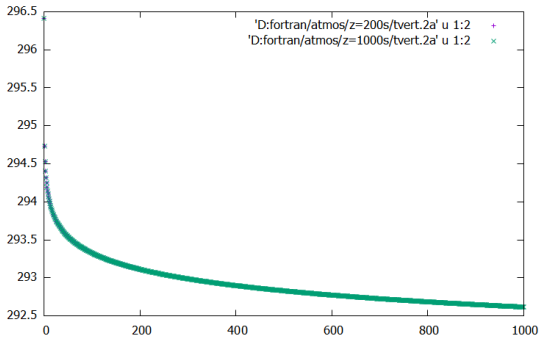
$$\theta(i) = \theta(i-1) - ((z(i) - z(i-1)) \cdot \frac{H_0 \Phi_{hi}}{\varrho c_p l_i u_{*0}} \tag{48}$$

$$q(i) = q(i-1) - ((z(i) - z(i-1)) \cdot \frac{E_0 \Phi_{hi}}{\varrho l_i u_{*0}} \tag{49}$$

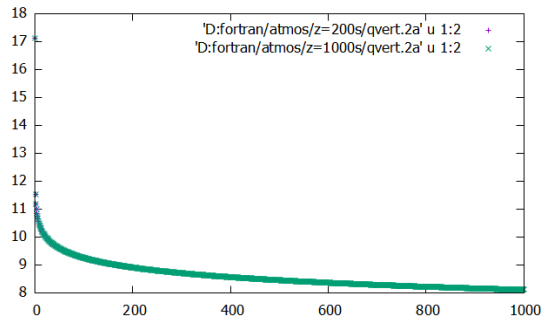
Where $l_j = l_{ma_j}$ and $\theta(i)$ and $q(i)$ are the temperature and humidity at vertical grid point i. The resulting Fortran code can be seen in the appendix in listing 6 and the resulting initial profiles are given in figures 5a and 5b. As seen in figures 5a and 5b, the initial profiles with boundary layers at different heights do now overlap.

4.2.4 Horizontal profiles

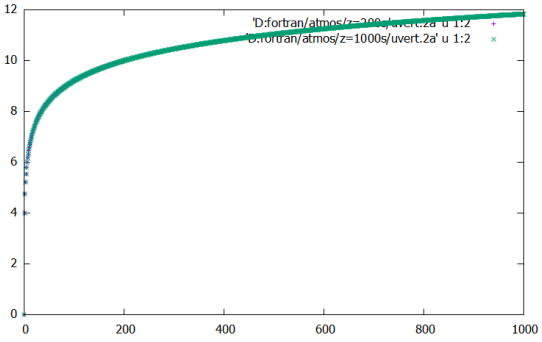
Now that the initial profiles are corrected, the horizontal profiles can be computed again. As we can see in figure 6, compared to figure 4, the gap between the horizontal profiles of two models with different top layer heights is much smaller, but unfortunately there still occurs an error. To try to correct this error we will use grid refinement, as explained in the next section.



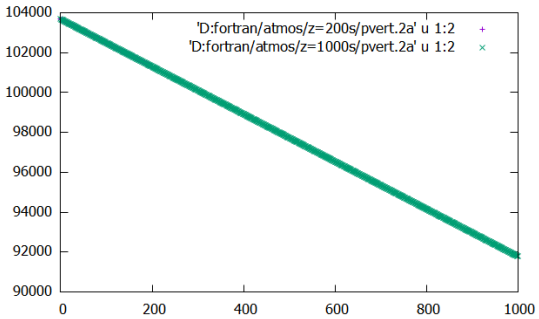
(a) Initial profiles of temperature after adjustments



(b) Initial profiles of humidity after adjustments

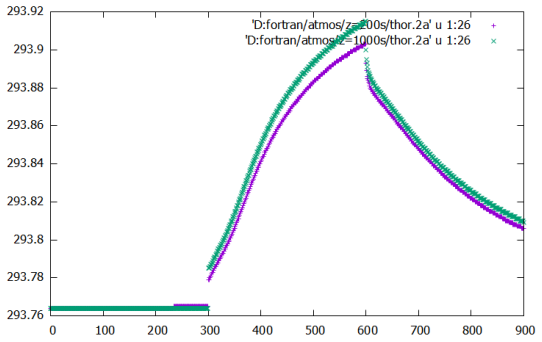


(c) Initial profiles of horizontal wind speed after adjustments

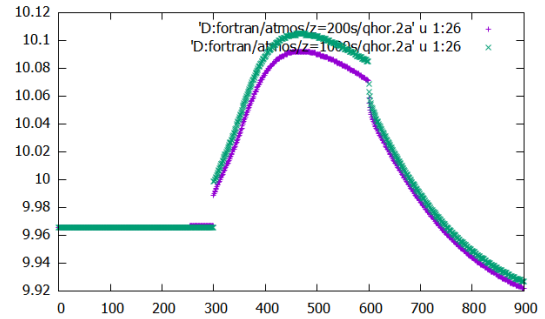


(d) Initial profiles of pressure after adjustments

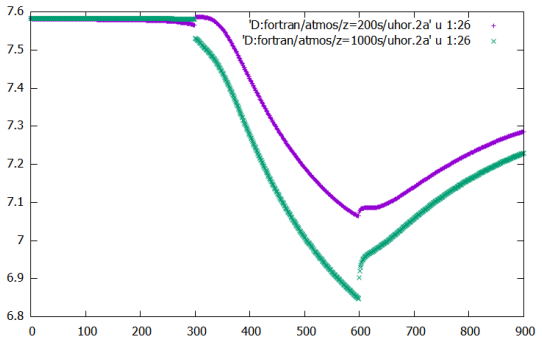
Figure 5: Initial profiles: top boundary layer at 1000 m vs 200 m.



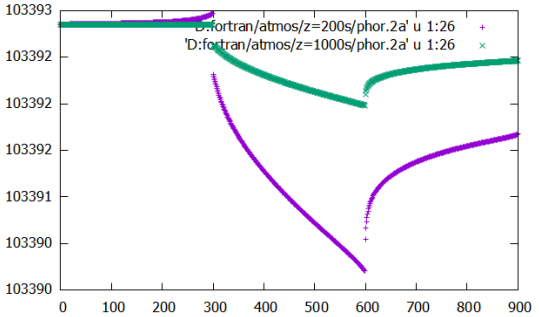
(a) Horizontal profiles of temperature after adjustments



(b) Horizontal profiles of humidity after adjustments



(c) Horizontal profiles of horizontal wind speed after adjustments



(d) Horizontal profiles of pressure after adjustments

Figure 6: Horizontal profiles: top boundary layer at 1000 m vs 200 m.

4.3 Grid refinement

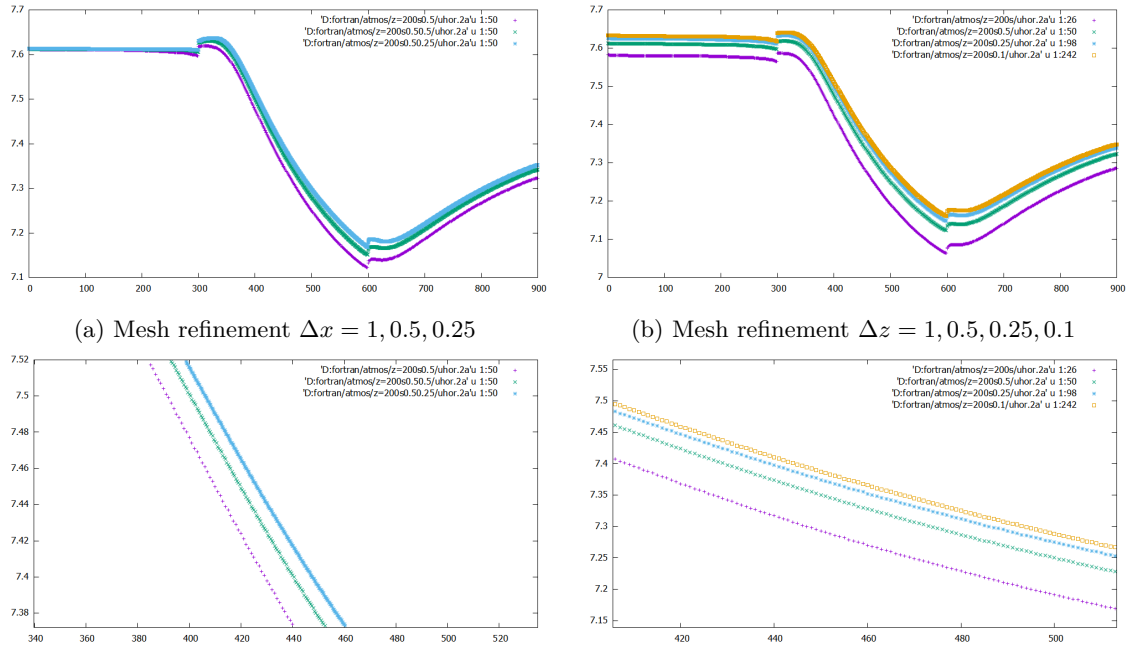
4.3.1 z-direction

In section 4.2 a grid refinement in z-direction was already used, namely the change from a manually created grid to an equidistant grid. To refine the grid even more we simply have to change the value of Δz and N, Δz is vertical step size and N is the amount of vertical grid points. For example, if we want to refine the grid from a 1-meter interval between layers to a 0.5-meter interval between layers, we change Δz from 1 to 0.5 and we change N to $2N$. So when we change Δz_{old} to Δz_{new}, we need to change N to $\frac{\Delta z_{old}}{\Delta z_{new}} * N$. The grid was refined in z-direction to $\Delta z = 1, 0.5, 0.25, 0.1$. The results of this are shown in the figures 7a and 7c. As seen in these figures, the solution still converges if we take a smaller Δz. So the grid refinement results in a more accurate solution. But if we then look at figure 9, we still see that the error occurs when we use different heights of top layers.

4.3.2 x-direction

The grid is x-direction is already an equidistant grid, so we do not need to change a lot to refine the grid. Similar to the refinement in z-direction only Δx and M are changed. In this case Δx is the horizontal step-size and M is the number of horizontal grid points. If we change Δx_{old} to Δx_{new}, we need to change M to $\frac{\Delta x_{old}}{\Delta x_{new}} * M$. We refined the grid in x-direction to $\Delta x = 1, 0.5, 0.25$. This resulted again in convergence to a more accurate solution, as seen in figures 7b and 7d. And if we look at figure 8, we see that also in this case the error between different heights of top layers still exist.

In listing 7 in the appendix, the Fortran code where the grid refinements take place is shown.



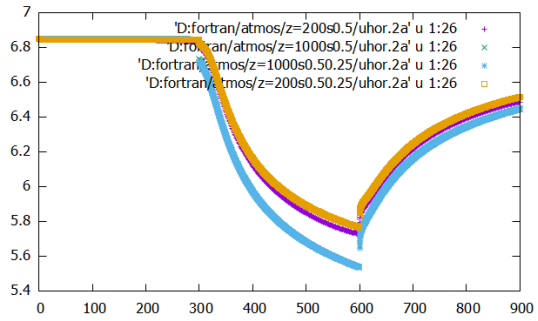
(a) Mesh refinement $\Delta x = 1, 0.5, 0.25$

(b) Mesh refinement $\Delta z = 1, 0.5, 0.25, 0.1$

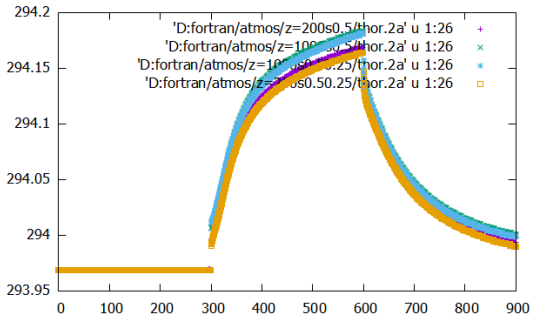
(c) Close up on 7a

(d) Close up on 7b

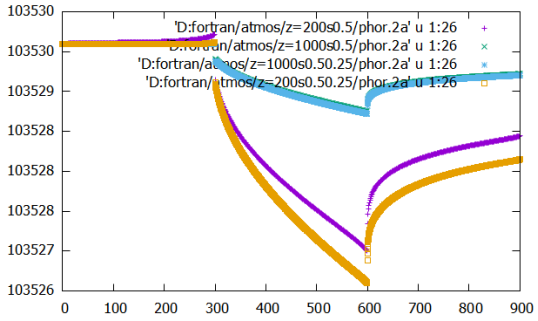
Figure 7: Mesh refinements results for u; colors of lines in order of smaller $\Delta z/\Delta x$: purple, green, blue, yellow.



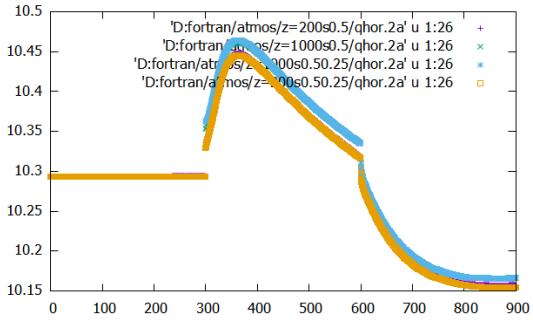
(a) Effect of mesh refinement on u $DX = 1, 0.25$



(b) Effect of mesh refinement on t $DX = 1, 0.25$

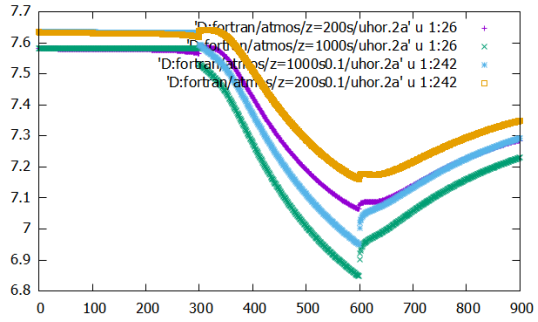


(c) Effect of mesh refinement on p $DX = 1, 0.25$

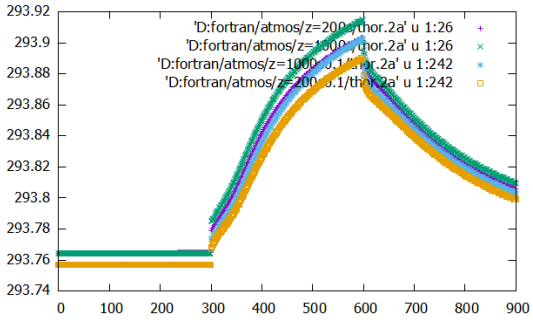


(d) Effect of mesh refinement on q $DX = 1, 0.25$

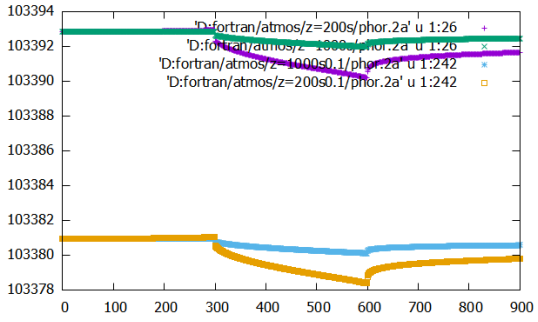
Figure 8: Top boundary layer height 1000 m (blue/green line) vs 200 m (yellow/purple).



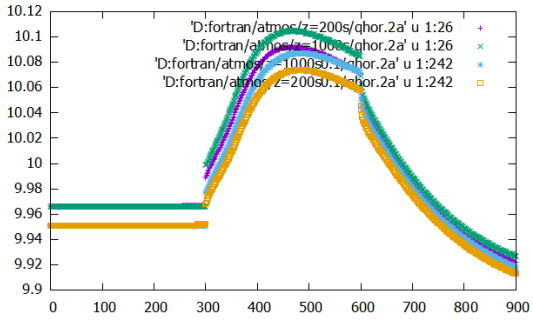
(a) Effect of mesh refinement on u $DZ = 1, 0.1$



(b) Effect of mesh refinement on t $DZ = 1, 0.1$



(c) Effect of mesh refinement on p $DZ = 1, 0.1$



(d) Effect of mesh refinement on q $DZ = 1, 0.1$

Figure 9: Top boundary layer height 1000 m (blue/green line) vs 200 m (yellow/purple).

5 Conclusion and Discussion

The goal of this study was to improve the solutions of an existing 2D atmospheric boundary layer simulation software by means of mesh refinement. To obtain this goal, adjustments were made to the initial profiles and on the definition of the existing grid. The results of these adjustments will be discussed shortly. In section 4 the adjustments that were implemented in the code were described and the results of these adjustments were shown.

- The adjustments to improve the initial profiles were successful. After the adjustments the initial profiles stayed along the same graph, while the top boundary layer height changed.
- Grid refinement showed that the solution converges if a smaller distance between grid points is used. However, grid refinement did not change the difference between graphs in case of different boundary layer heights.

Therefore we can conclude that grid refinement did give more accurate solutions but did not fully solve the problem we wanted to solve. Unfortunately the solution computed by the software still depends on the height of the upper boundary layer.

Possible causes of the problems could be in the definition of the boundary conditions at the upper boundary layer. Also the application of the FLARE condition may not be sufficient enough to handle negative values of u, which typically occur in the lower layers directly after the forest. Hence, for future research the interaction law can be revised and research could be done to find a better way to handle negative values of u. Another point for future research is the multi-layer model in combination with mesh refinement. In this project we kept the surface layers of the multi-layer model for every grid refinement the same, but it would be reasonable to redefine and refine the layers under the canopy. Since we can describe the vegetation in more detail if we put in more layers to describe the vegetation, this will probably lead to more accurate solutions.

References

[K.Gregson and Unsworth, 1990] K.Gregson, N. C. and Unsworth, M. (1990). TLM: A technique with application in the numerical solution of diffusion problems. *Agricultural Forest Meteorology*, 51:1–20.

[Klaassen, 1992] Klaassen, W. (1992). Average fluxes from heterogeneous vegetated regions. *Boundary Layer Meteorology*, 58:329–354.

[Luppes, 1993] Luppes, R. (1993). Atmospheric flows in heterogeneous vegetated regions. Technical Report W93.03, University of Groningen, Department of Mathematics.

[Valters,] Valters, D. introduction to fortran. *coding club (ourcodingclub.githhub.io)*.

[Veldman, 2010] Veldman, A. (2010). *Boundary Layers in Fluid Dynamics*. University of Groningen, Department of Mathematics. Lecture Notes.

6 Appendix

6.1 Subroutine schemes

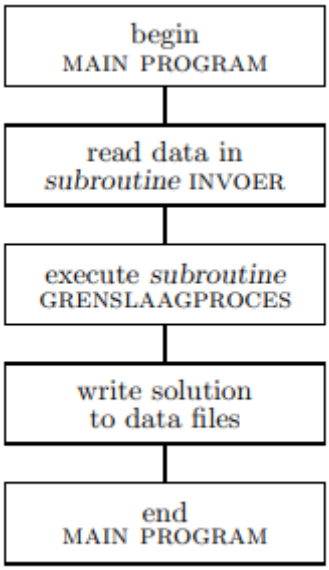


Figure 10: subroutine MAIN

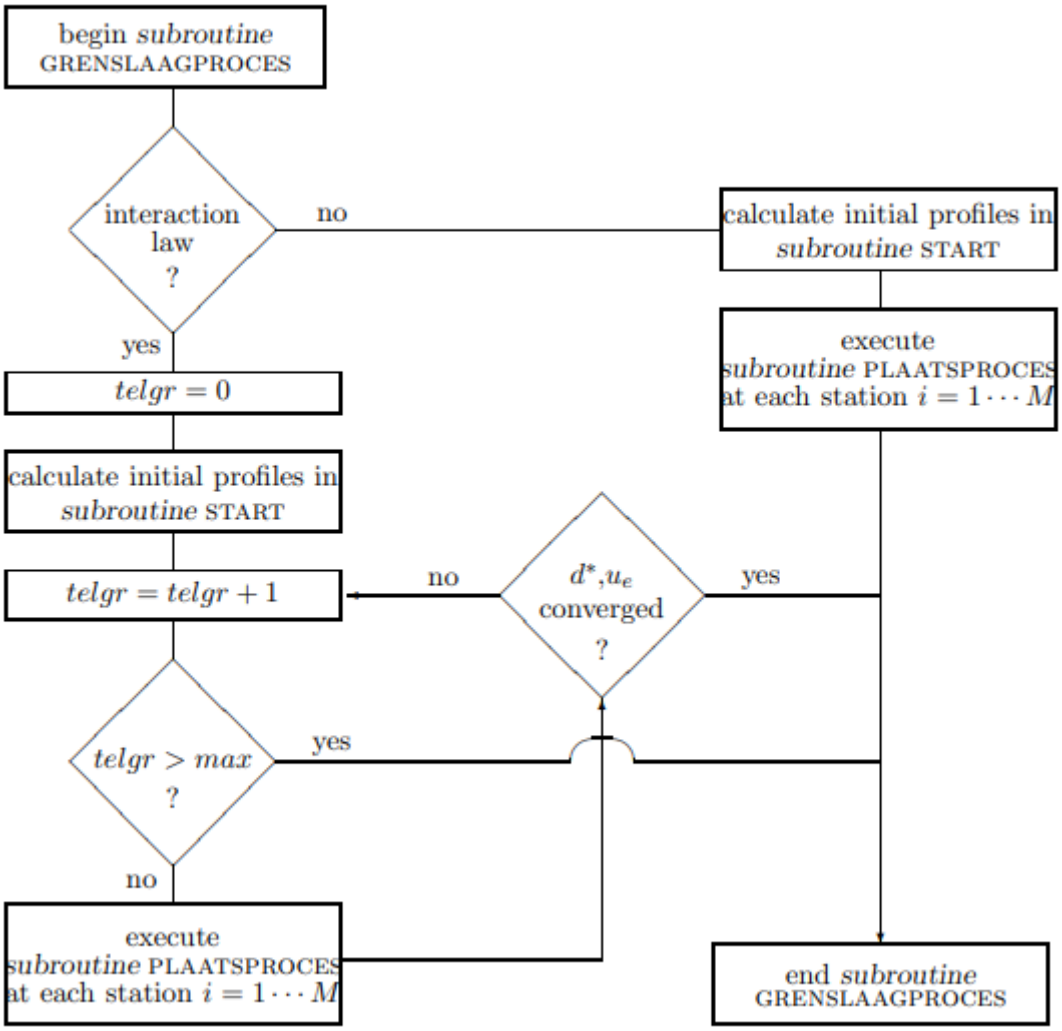


Figure 11: subroutine GRENSLAAGPROCES

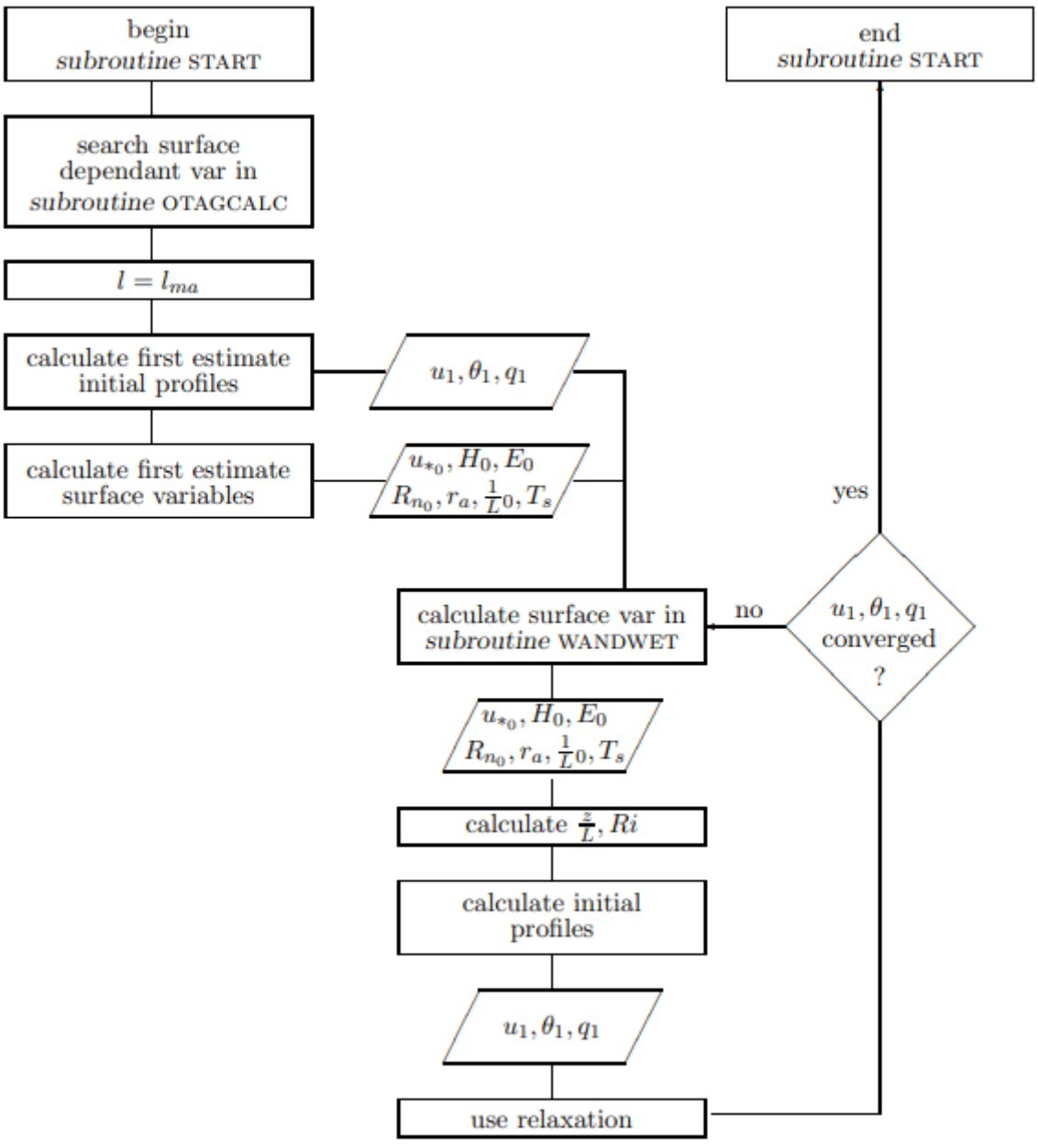


Figure 12: subroutine START

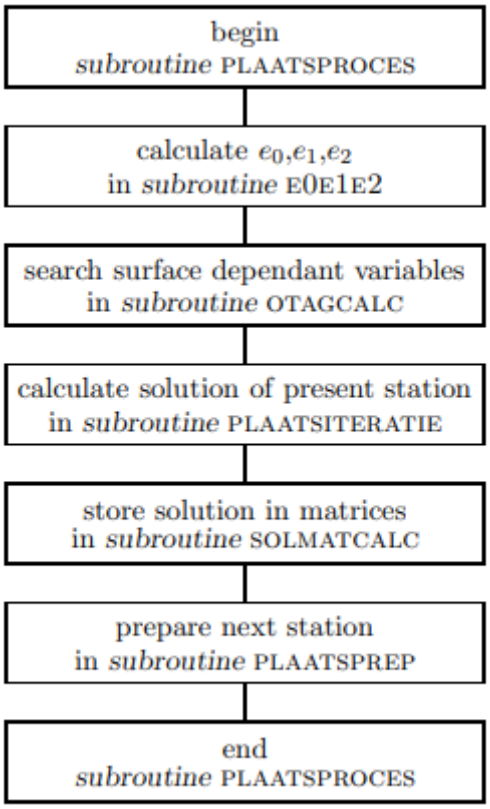


Figure 13: subroutine PLAATSPROCES

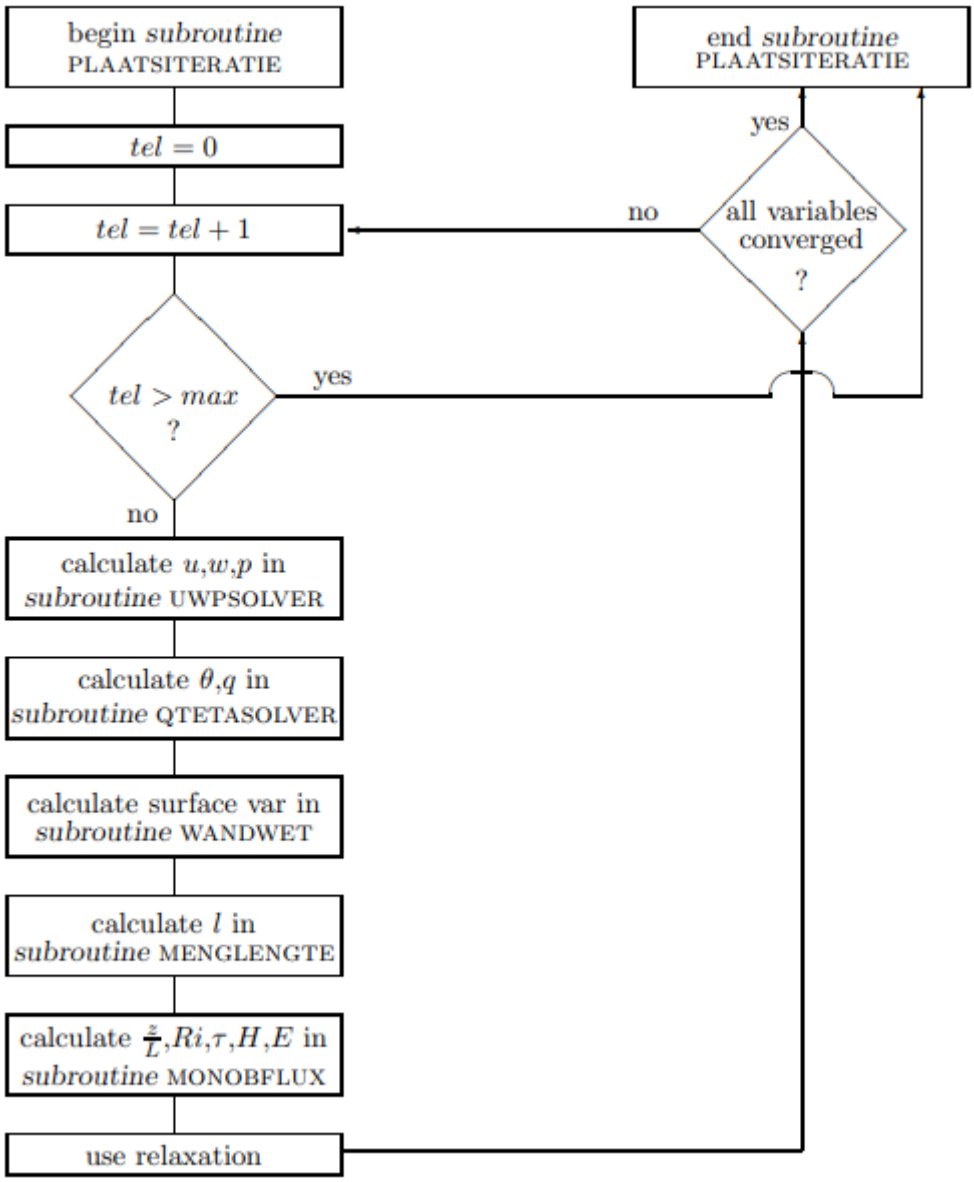


Figure 14: subroutine PLAATSITERATIE

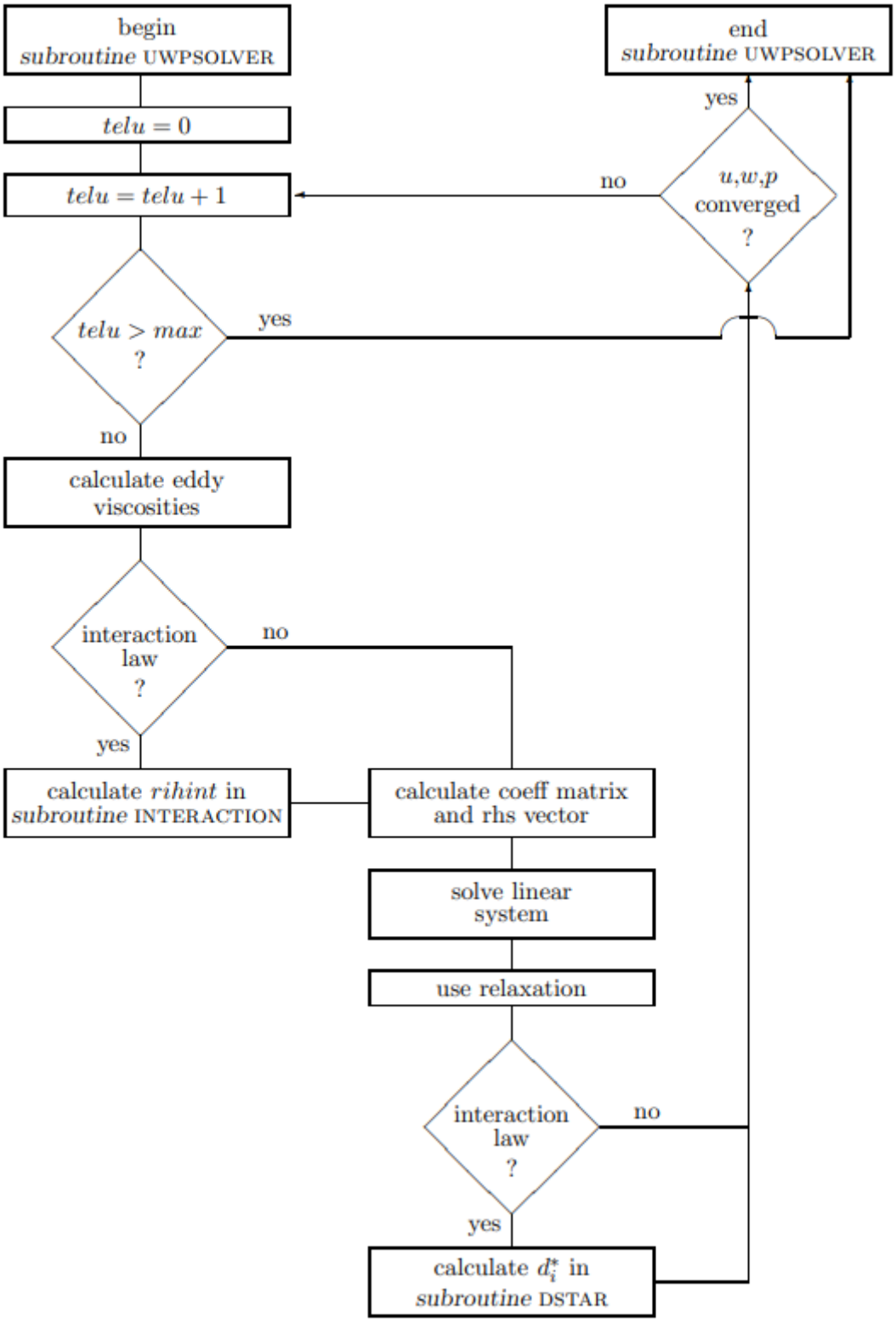


Figure 15: subroutine UWPSOLVER

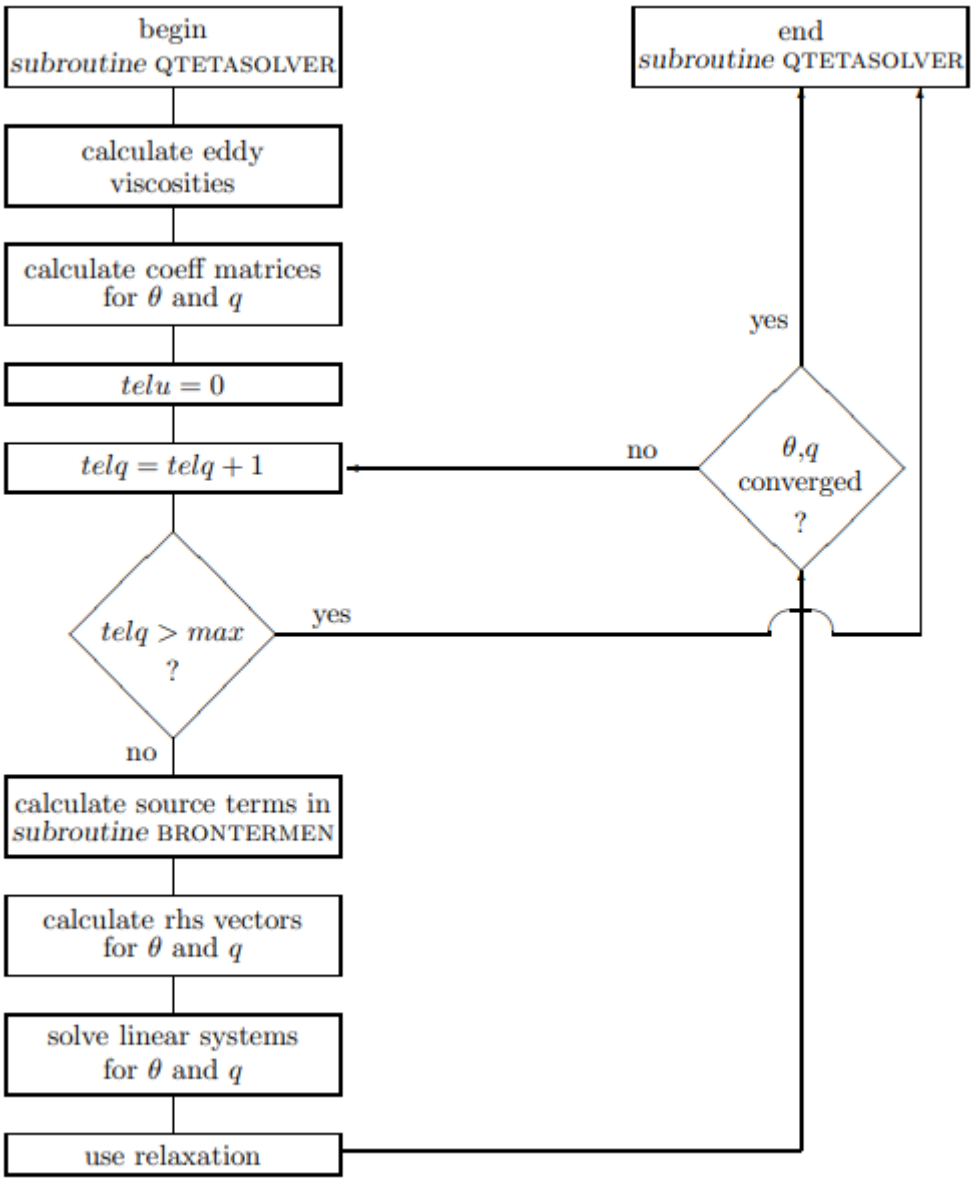


Figure 16: subroutine QTETASOLVER

6.2 Code listings

Listing 1: Definition of the Grid

```
c ********** define grid **********
c
c denk om de maxima !
      N=999
      M=900
      DX=1.0
      DZ=1.0
c denk om z(1)>max(z0opp)
c en de laatste moet z(N+1) zijn !
c N>N00!
c
      z(1)=1.0
      do  nc=2,N+1
        z(nc)=z(1)+(nc-1)*DZ
      end do
```

Listing 2: Multi-layer definition of the Grid

```
c to still have the layers in the multi-layer model on the right
c  height another way of creating the grid is needed:
      if (areatype(2) .eq. 3) then
      z(1)=1.5
      z(2)=2.5
      z(3)=3.5
      z(4)=4.5
      z(5)=5.5
      z(6)=7.0
      z(7)=8.5
      z(8)=9.7
      z(9)=12.0
      z(10)=15.0
      do 19 j=11,N+1
        z(j)=z(10)+(j-10)*DZ
   19 continue
      endif
```

Listing 3: initial values of t_0, p_s, r_h at height zstart and the value of u_s

```
c**>  metereological measurements (reals)
c     t0=temp. at z(N00)=zstart,x=0 in Kelvin
c     (use 273.15 in calculation),
c     now it it the temp. at z(200)=200
c     us=hor. velocity at z(N+1),x=0 in m/s
c     rh=rel. humidity at z(N00),x=0 in digits, not in %
c     ps=pressure at z(N00),x=0 in N/m2
c         10.0 is intitial windspeed at zstart
c     0.03 is the roughnesslength of areatype(1)

      zstart=200.0
      N00=(zstart-1+DZ)/DZ
      t0=291.15
      zrat=zstart/0.03
      zratt=z(N+1)/0.03
      us=(10.0*dlog(zratt))/dlog(zrat)
      rh=0.70
```

```
      ps=101300.0
```

Listing 4: pressure definitions needed for initial profiles

```
      plow=p+(dens*grav*(z(N00)-0.0))

*******************************************************

      do 50 j=1,N00
         pvsol(i,j)=p+(dens*grav*(z(N00)-z(j)))
   50 continue
      do 51 j=N00+1,N+1
         pvsol(i,j)=p-(dens*grav*(-z(N00)+z(j)))
   51 continue
      pvsol(i,0)=p+(dens*grav*(z(N00)-0.0))
```

Listing 5: calculation of T_s, q_s, ϱ

```
c ***** calculation part *****
      dens=(mwa*ps)/(gas*t0)
      klad=rh*svp(t0)
      qs=(0.622*klad)/(ps+((0.622-1.0)*klad))
      tets=t0+(gam*zstart  )+(0.5*gam*gam*zstart  *zstart  /t0)
c ***** end of calculation part *****
```

Listing 6: Loops that generate the initial profiles of θ and q

```
c taking z(N00)=zstart as starting point,
c to create the initial profile in 2 directions
    do 31 j=N00-1,1,-1
      u(j)=u(j+1)-((z(j+1)-z(j))*ustar0*cm/l(j))
      tet(j)=tet(j+1)
      q(j)=q(j+1)+((z(j+1)-z(j))*EE0*ch/(dens*l(j)*ustar0))
 31 continue

    do 32 j=N00+1,N+1
      u(j)=u(j-1)+((z(j)-z(j-1))*ustar0*cm/l(j-1))
      tet(j)=tet(j-1)-((z(j)-z(j-1))*HH0*ch/(dens*cp*l(j-1)*ustar0))
      q(j)=q(j-1)-((z(j)-z(j-1))*EE0*ch/(dens*l(j-1)*ustar0))
 32 continue
```

Listing 7: grid refinement

```
c ********** define grid **********
c
c denk om de maxima !
      N=999
      M=900
      DX=1.0
      DZ=1.0
```