MIMICA V2.0, User guide

Julien SAVRE

31-01-2013

Contents

1 Coding standards

The MIMICA model has been designed as an extension of the CRM-MIT model developed by Chien Wang and Julius Chang [2]. As such, it still contains many of the features originally made available in the CRM-MIT and most of the variable names were retained. Not considering the additions regarding the numerics and physics implemented, the present LES code differs mostly from its predecessor in the coding standards adopted, the overall structure of the code being kept as close as possible to the original CRM.

Note that even though most of the model's core routines have already been modified to fully comply with the following standards, some of the optional modules (aerosol and chemistry) as well as certain routines generally used as black boxes have just been partially rewritten so far. As a result, a certain inhomogeneity in the style exists between the main routines used for physical and numerical calculations and several peripheric packages and modules.

The standards adopted try to optimize as much as possible the clarity of the code by making use of modern fortran 90/95 commands, and by adding commented lines where this seems necessary. This is sometimes done to the detriment of code performances and overall optimization, but we believe that this drawback is largely compensated by the simplicity of comprehension and easiness of coding made available to the user. The most commonly used routines of the code have been rewritten in REAL fortran 90 which enables a more condensed writting (vector operations).

The basic rules used in the new fortran 90 routines are described below:

- each routine or package is introduced by a header containing a short description of the routine and the history of all the major changes made since the first version;
- all the variables used in the calculations (winds, hydrometeors, turbulent parameters, thermodynamics....) must be invoked through the use of the corresponding modules (see section 5), followed by *implicit none*;
- local variables are defined in the local routine only; variables called in several calculations or written in output files must be defined in one of the data structure/type (*shared_data* for constants, *shared_xxxxx* for 3D fields);
- the use of *#include* statements in the declaration section must be avoided as much as possible;
- *if*, *do*, etc... statements are idented: 2 void spaces are left before each new nested block;
- if a code line is cut and appears on several lines, 4 void spaces are added before the following lines;
- no tabulation, use the space bar as much as possible;
- add commented lines introduced in the first column by the character "!" (for example to introduce new sections or delimit independent operations); the comments must be short but explicit enough to describe the purpose of the following lines of code;
- the original models must be left untouched: if new methods are implemented, create an optional environment variable in *start* and *wheader.csh*;

MIMICA is managed through the subversion system SVN. This software provides a very simple way to share our developments and get the latest updates of the code. This makes the developer's life much easier and allows everyone to follow real-time all the current developments and latest modifications, with an improved possibility to track bugs or upgrade its personal version. Basically, using SVN, a common main repository for the code is stored somewhere which includes in a condensed format all the successive versions of all the files. This repository is managed by an administrator fixing the access rules and its initial content. All the registered users having downloaded the code through SVN can either download the current version stored in the remote repository or, on the opposite, put their own local modifications into the main repository. SVN will automatically merge every modifications. Therefore, there always exist one local version of the code owned by each user that can be managed and customized at will, and one remote version including all the latest commited modifications (plus we could add all the previous commited versions stored into the remote directory). It is strongly recommended that users make sure the remote repository always contains a clean and running version of the code (that means the code must compile, run properly -no segmentation fault for example-, that there are no "print*" for debugging, no case specific branch or bypass, etc.).

As a brief introduction to SVN, here are the most basic commands to know:

- the code can be downloaded through: *svn checkout ssh://x_julsavagn.nsc.liu.se/svn_MIMICA .*
- when a working modification has been done, the development can be shared through: *svn commit file1.f90 file2.f90 ...*; an editor window will open automatically in which a short description of the commit can be written

- in order to update its own version and merge it with the main repository to account for the latest modifications, simply type: *svn update*
- differences between its own version and the main repository can be displayed via: *svn diff*.

Of course, more can be done thanks to SVN. As all the successive versions of each routine is stored, it is possible to apply these commands to one particular release of the code by adding the option *-rxxx*, where xxx is the release number. To perform the previous operations, a personal password will be required. In order for every user to follow the evolution of the code, an automatic mail is sent to the subscribers when a commit is done: the mail contains some important information such as the date, the author's name, the modified files and the short description written by the author.

2 Physical model description

In the following section, the description focuses on the main core of the model which is necessarily used in each application. The aerosol and chemistry modules are therefore excluded (temporarily).

2.1 The LES approach

In LES, the governing equations are filtered in the physical space. Depending on the kind of spatial filter used, part of the turbulent spectrum, associated with the large turbulent eddies (small wave numbers), can be directly and accurately resolved. On the other hand of the turbulent spectrum, the small dissipative turbulent structures (large wave numbers), typically smaller than the cut-off scale of the filter, cannot be resolved. Subgrid scale (SGS) models thus need to be included in the equations in order to account for the influence of the unresolved part of the spectrum on the resolved turbulent structures.

On a mathematical point of view, the filter kernel is commonly denoted by G_Δ, and the filtering operation can then be written as:

$$\overline{\Phi} = \int_{-\infty}^{+\infty} \Phi\left(\mathbf{x}\right) G_\Delta\left(\mathbf{x}-\mathbf{x}'\right) d\mathbf{x}', \tag{1}$$

where Φ is a dummy variable, $\mathbf{x}$ the position in the multidimensional space and $\overline{\cdot}$ represents a filtered quantity. The kernel filter can be given any prescribed shape in the physical or spectral space (top-hat, gaussian...). In practical LES codes, it is always more convenient to define implicitly the filter which avoids the necessity to calculate integral 146. The cut-off scale in the physical space is then defined both by the grid and by the numerical scheme (the dissipative role played by the numerical discretization is often omitted).

Based on the filtering operation 146, any local, instantaneous quantity Φ can be decomposed in a resolved filtered part and an unresolver SGS part:

$$\Phi = \overline{\Phi} + \Phi'. \tag{2}$$

Note that two important properties that are true for RANS methods, are no longer valid in LES:

$$\overline{\overline{\Phi}} \neq \overline{\Phi} \qquad\qquad \text{and} \qquad\qquad \overline{\Phi'} \neq 0. \tag{3}$$

These two inequalities complexify greatly the filtered equations.

If we now consider that the evolution of Φ is governed by an advection/local production equation,

$$\frac{\partial \rho\Phi}{\partial t} + \frac{\partial \rho u_i \Phi}{\partial x_i} = \dot{s}, \tag{4}$$

where $\dot{s}$ represents a local source or sink, and if we apply a spatial filter to this equation, we obtain:

$$\frac{\partial \overline{\rho\Phi}}{\partial t} + \frac{\partial \overline{u_i}\,\overline{\rho\Phi}}{\partial x_i} = \overline{\dot{S}} + \mathcal{R}_\Phi. \tag{5}$$

To simplify further this equation, Favre filtering can be invoked, where $\overline{\rho\Phi} = \overline{\rho}\widetilde{\Phi}$. In equation 5, $\mathcal{R}_\Phi$ contains all the various cross correlations between Φ and u appearing when the filter is applied. It reads:

$$\mathcal{R}_\Phi \quad = \quad \frac{\partial}{\partial x_i}\left(\overline{\rho u_i \Phi} - \overline{u_i}\,\overline{\rho\Phi}\right) = \frac{\partial \tau_i^\Phi}{\partial x_i}. \tag{6}$$

The exact form of $\mathcal{R}_\Phi$ is made complicated by the two properties introduced previously and won't be detailed here. It contains three main terms: one represents the interactions between the large scales, one represents the interactions between small and large scales, and the last one contains only informations on the subgrid scales. The turbulence closures discussed in section 2.3 model the SGS tensor τ^Φ.

2.2 Governing equations

The model solves for a non-hydrostatic incompressible system, in two or three dimensions. Compared to the original CRM model, the anelastic assumption constitutes one of the major differences which is reflected by a heavily modified solver algorithm (see section 3). The equations are spatially filtered as described in the previous section but the filter notation $\overline{\cdot}$ will be omitted for clarity, as this is commonly the case in the litterature.

2.2.1 Pressure decomposition

Using the anelastic, non-hydrostatic framework, the Exner pressure (used instead of the regular pressure) is decomposed into three parts (see for instance Clarks [4]): an isentropic reference state varying with altitude only, an horizontal deviation from the reference state assumed to be in hydrostatic balance with its environment, and a 3D dynamic perturbation which needs to be updated after each time step. This yields:

$$\Pi(x, y, z) = \pi(t; x, y, z) + \pi_0(z) + \pi_1(t; z), \tag{7}$$

where 1 denotes the hydrostatic part and 0 the reference isentropic part. In the above relation, the adimensional Exner pressure was used instead of the thermodynamic pressure P as commonly done in atmospheric models. Let's recall the relation between the two pressures:

$$\Pi = \left(\frac{P}{P_{ref}}\right)^{R/c_p}, \tag{8}$$

where P is the total thermodynamic pressure; $P_{ref} = 100000\ Pa$, R is the ideal gas specific constant for dry air (287 $J/K/kg$) and c_p is the specific heat capacity at constant pressure for dry air ($= 1004\ J/kg$). The same decomposition as for Π can be applied to P.

The reference pressure π_0 is held constant throughout the simulation and does not require any update. It satisfies the following relation:

$$\frac{\partial \pi_0}{\partial z} = -\frac{g}{c_p \theta_{00}}, \tag{9}$$

where θ_{00} is a reference potential temperature, usually chosen as being the surface value of θ.

The hydrostatic pressure deviation π_1 won't be evaluated in the same way if a large scale horizontal divergence is enforced ("traditional LES"), or if we consider a free simulation. In the first case, π_1 is used to force the horizontal large scale divergence: $D_{LS} = -\frac{\partial w}{\partial z}$. If we decompose the vertical velocity as $w = w_0 + w'$, with w_0 the horizontal mean of w and w' the deviation of w around w_0, then the vertical wind tendency, including advection, SGS mixing and buoyancy, may result in a vertical acceleration of the horizontal mean. In order to keep this average constant and equal to the large scale forcing w_{LS} (determined by D_{LS}), we must impose an additional constraint:

$$\left\langle \frac{\partial w}{\partial t} \right\rangle = \langle w_s \rangle = 0. \tag{10}$$

In the above, w_s is the total vertical wind tendency and $\langle . \rangle$ represents the horizontal average. In order to force the previous constraint, π_1, appearing in w_s as a vertical pressure gradient, is calculated to balance the vertical acceleration resulting from all the other physical phenomena. Noting w_s^* the tendency without the π_1 gradient, we may write:

$$\langle w_s^* \rangle - c_p \theta_{00} \frac{\partial \pi_1}{\partial z} = 0, \tag{11}$$

or:

$$\frac{\partial \pi_1}{\partial z} = \frac{1}{c_p \theta_{00}} \langle w_s^* \rangle, \tag{12}$$

By integrating this last relation over the vertical dimension, π_1 can be updated after each time step so that the horizontal average of the vertical wind always satisfies the fixed horizontal flow divergence constraint.

In the case of a free simulation, without any forcing, π_1 remains constant throughout the simulation and expresses simply the deviation of π_0 from the local hydrostatic balance, so that we can write:

$$\frac{\partial(\pi_0 + \pi_1)}{\partial z} = -\frac{g}{c_p \theta_{v0}}. \tag{13}$$

Here, θ_{v0} is the initial virtual temperature sounding, varying with altitude. π_1 can the be calculated by substracting the previous relation with equation 9. The virtual potential temperature is defined by $\theta_v = \theta(1 + 0.61 q_v)$ where q_v is the water vapor mixing ratio. The potential temperature appearing in the previous relation is defined by $\theta = T(p/p_{ref})^{R/c_p}$.

Finally, under the anelastic assumption, the dynamical pressure is updated after each time step to force the flow divergence to remain equal to 0. As explained in the next section, this requires the inversion of a Poisson equation derived from the assumption that $div(\mathbf{u^{n+1}}) = 0$ and for which π is solution.

2.2.2 Balance equations

The system includes the momentum equations, an energy equation (potential temperature equation), an equation for the total water mixing ratio and equations for two hydrometeor moments: the number concentration and mass mixing ratio (see section 2.5). The momentum equations in non conservative form read:

$$\frac{\partial \mathbf{u}}{\partial t} + u_i \frac{\partial \mathbf{u}}{\partial x_i} = -c_p \theta_{00} \frac{\partial \pi}{\partial x_i} + b\delta_{i,3} + f_k \left(\mathbf{u} - \mathbf{u}_g\right) \epsilon_{i,j,k} + \frac{1}{\rho_0} \frac{\partial \tau^u}{\partial x_i}. \tag{14}$$

In the above, $\mathbf{u}$ is the velocity vector (u, v, w), Cp is the atmosphere's heat capacity at constant pressure (≈ 1005), θ_{00} is a reference potential temperature (initial surface value), b is the vertical buoyancy flux, τ^u is the subgrid scale viscous tensor, and f_k is the Coriolis parameter with $\mathbf{u}_g$ being the geostrophic wind vector. $\delta_{i,3}$ and $\epsilon_{i,j,k}$ denote the Kronecker and Levi-Civita symboles respectively whereas ρ_0 is the reference density of the isentropic state. The buoyancy flux, appearing only in the vertical wind equation, is expressed under the anelastic assumption and following Clarks as the deviation of the virtual potential temperature from its horizontal average:

$$b = g \left(\frac{\theta - \langle \theta \rangle}{\theta_{00}} + 0.61 \left(q_v - \langle q_v \rangle\right) - q_l - q_i \right), \tag{15}$$

with q_l and q_i the total liquid and total ice mass mixing ratios.

In the MIMICA model, a governing equation for θ_{il}, the ice/liquid potential temperature, is solved as a proxy for the energy balance equation. The choice to use the ice/liquid potential temperature instead of any other temperature was made because θ_{il} is conserved during all the microphysical processes. Its balance equation reads:

$$\frac{\partial \theta_{il}}{\partial t} + u_i \frac{\partial \theta_{il}}{\partial u_i} = \frac{1}{\rho} \frac{\partial \tau^\theta}{\partial x_i} + \mathcal{Q}_{rad} + \mathcal{Q}_{prec}. \tag{16}$$

In equation 16, τ^θ is the subgrid scale contribution related to the potential temperature and $\mathcal{Q}_{rad}$ is a radiative source term (depending on the case this latter can be calculated via simplified formula or given by an atmospheric radiative transfer model). $\mathcal{Q}_{prec}$ is a precipitation related term which arises directly from the vertical redistribution of water vapor through hydrometeor precipitation. Following Wang and Chang [2] it can be written:

$$\mathcal{Q}_{prec} = \frac{\theta_{il}^2}{\theta} \frac{L_v \mathcal{P}_l + L_s \mathcal{P}_i}{c_p T}, \tag{17}$$

with $\mathcal{P}_l$ and $\mathcal{P}_i$ the total precipitation fluxes of liquid and ice water. Expressions for these two fluxes are given below. Let's recall that the ice/liquid potential temperature is related to the potential temperature through:

$$\theta_{il} = \theta \left(1 + \frac{L_v q_l + L_s q_i}{c_p T}\right)^{-1}. \tag{18}$$

For a similar reason as for θ_{il}, the total water mass q_t is defined as a prognostic quantity in the model. This ensures that water mass is perfectly conserved throughout the simulation:

$$\frac{\partial q_t}{\partial t} + u_i \frac{\partial qt}{\partial u_i} = \frac{1}{\rho} \frac{\partial \tau^{qt}}{\partial x_i} + \mathcal{S}_{prec}. \tag{19}$$

Here again, $\mathcal{S}_{prec}$ denotes a source/sink of total water due to redistribution through precipitation. The mass mixing ratio of water vapor is a diagnostic quantity which is evaluated by substracting the hydrometeors mixing ratios to q_t.

Finally, the other scalars, namely the mass mixing ratios and number concentrations of all the hydrometeors, are governed by the following equation:

$$\frac{\partial \mathbf{\Psi}}{\partial t} + u_i \frac{\partial \mathbf{\Psi}}{\partial x_i} - \frac{1}{\rho} \frac{\partial \rho \left\langle \mathbf{V}_p \right\rangle \mathbf{\Psi}}{\partial z} = \frac{1}{\rho} \frac{\partial \tau^\Psi}{\partial x_i} + \mathbf{S}_{micro} \tag{20}$$

where $\mathbf{\Psi} = (q_k, n_k)$, with k=1,....,$N_h$, N_h being the total number of hydrometeors. In the above, $\mathbf{S}_{micro}$ represents a microphysical source or sink per unit mass, τ^Ψ is the subgrid scale contribution and $\langle \mathbf{V}_p \rangle$ are the mass weighted precipitation velocities averaged over the hydrometeor populations (assumed to be 0 for vapour and cloud droplets).

All the thermodynamic quantities are related through the ideal gas law for a wet atmosphere:

$$P = \rho r_d T \left(1 + 0.61 q_v\right). \tag{21}$$

This is used to retrieve the density and close the thermodynamic system.

When the anelastic equations are solved (diagnostic pressure option), the equation system is closed by the incompressible constraint that the velocity divergence must vanish everywhere:

$$\frac{\partial \rho_0 u_i}{\partial x_i} = 0. \tag{22}$$

As explained in section 3.7, the perturbation pressure is actually diagnostically computed at the end of each time step in order to make sure that this constraint is respected.

2.3 Subgrid scale closure

SGS models are used to evaluate the subgrid scale contributions τ^Ψ of all the scalars. By analogy with molecular diffusion, a simple hypothesis enables to write:

$$\tau_i^\Psi = -\rho K_h \frac{\partial \Psi}{\partial x_i} \tag{23}$$

for the scalars and

$$\tau_{i,j}^u = -\rho K_m \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{24}$$

for momentum. Two approaches can then be used to evaluate the eddy diffusivities K_h and K_m.

The first approach is a 1st order closure based on the solution of a turbulent kinetic energy equation (Deardorff [5]). A closed form of the SGS TKE equation can be written (without further details):

$$\frac{\partial e}{\partial t} + u_i \frac{\partial e}{\partial x_i} = 2K_m S_{ij} S_{ij} + \frac{2}{\rho} \frac{\partial}{\partial x_i} \left(\rho K_m \frac{\partial e}{\partial x_i} \right) - K_h N^2 - C_\epsilon \frac{e^{3/2}}{l}. \tag{25}$$

In the above equation, e is the SGS TKE, $S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$ is the strain tensor, N^2 is the Brunt-Vaisala frequency, and l a characteristic length scale for the SGS processes. It is usually assumed that $K_h = K_m/Pr$, with Pr the turbulent Prandtl number (≈ 0.4). The characteristic length scale is defined in the present LES model as (in 3D):

$$l = \left[(\Delta x \Delta y \Delta z)^{-2/3} + (z\kappa/C_S)^{-2} \right]^{-1/2}, \tag{26}$$

with z the local altitude, κ the Von Karman constant (≈ 0.35) and C_S the Smagorinsky constant (defined below). The eddy diffusivity K_m is finally obtained following scaling arguments:

$$K_m = C_m l \sqrt{e}. \tag{27}$$

The constants C_m, C_ϵ and C_S are given the values 0.15, 0.93 and 0.21 respectively.

The second closure is based on the 0th order Smagorinsky and Lilly approach [22]. If we assume the balance between production of TKE by shear plus buoyancy and dissipation in equation 25, it follows:

$$2K_m S_{ij} S_{ij} - K_h N^2 - C_\epsilon \frac{e^{3/2}}{l} = 0. \tag{28}$$

Rearranging the terms yields:

$$K_m = (C_S l)^2 \sqrt{2 S_{ij} S_{ij}} \sqrt{1 - \frac{Ri}{Pr}}, \tag{29}$$

where the Richardson number $Ri = \frac{N^2}{|\mathbf{S}|}$ was introduced to account for convective stability.

The Brunt-Vaisala frequency is modeled following Stevens et al. [23] to include changes in stability related to condensation/evaporation at the SGS level. N^2 is defined by:

$$K_h N^2 = -\frac{g}{\theta_{00}} \langle w' \theta_v' \rangle \approx K_h \frac{\partial b}{\partial z}. \tag{30}$$

By considering the buoyancy production generated by the mixing of two separate levels in saturated or subsaturated conditions, Stevens et al. [23] propose to model N^2 following:

$$N^2 = \frac{g}{\theta_{00}} \left[\Delta\theta_l + 0.608\theta_{00} \Delta q_t + \frac{n}{2} G \left(\Delta q_t - F \Delta\theta_l \right) \right]. \tag{31}$$

Here, n represents the number of saturated layer and the two parameters F and G are defined by : $F = \Pi dq_s/dT$ and $G = \left(L_v/c_p\Pi - \theta_{00}\left(1+r_m\right)\right)/\left(1+L_v/c_p dq_s/dT\right)$. n represents the number of consecutive saturated layers. In the model, we only consider the two neighbouring layers at a given level so that n is always set to 2 (for example within the cloud layer) or 0 (if the considered point is situated in subsaturated conditions). The above relation was developed by considering that the cloud water in saturated conditions is adjusted to saturation. This model does not account for ice phase water but we will still use it as such even when ice crystals are present.

The scalar eddy diffusivity is defined as the ratio between K_m and a turbulent Prandtl number. In MIMICA, it is possible to set Pr to a negative value, in which case K_h is assumed to decrease linearly up to an altitude of approximately 200 m over which K_h remains close to 0. This assumption is equivalent to the use of an implicit LES for scalars above the 200 m level where turbulent diffusion is only provided by numerical resolution and the dissipation of the numerical scheme. Provided that all the scalars are advected using flux-limited schemes which generate numerical diffusion in regions with strong gradients (see section 3), it can be argued that numerically induced diffusion is already sufficient so that there is no need for additional diffusion through an explicit SGS contribution.

One precision on the numerical discretization of turbulent diffusion terms must be added here. No particular procedure is applied to calculate the horizontal diffusion: the gradients are computed by simple 2nd order central differences, which is made easier by the staggered grid system, and the explicit diffusive terms are calculated at the same location as the velocity vector or scalars. However, due to the non-constant grid spacing in the vertical direction, for instance close to the surface, the velocity and scalar gradients may generate numerical waves leading to instabilities. To avoid their development, vertical turbulent diffusion for both velocities and scalars is implicited: the vertical contributions are rewritten into a tridiagonal system which can easily be solved using standard methods.

2.4 Radiation

Radiation in the LES model can be treated following to approaches: a detailed one making use of a separate radiation solver and a simplistic formulation where the radiation fluxes are expressed as functions of LWP only (very suitable for LES intercomparison studies as in Stevens et al. [7]). In *start*, the parameter RADIA controls the choice of the model, if it's true the detailed radiation solver is used whereas in the other case, the simplified model is selected.

The radiation solver coupled to MIMICA is the 2005 version of the Fu-Liou model developed by Fu and Liou [8]. The solver is a multiband 2-4 stream radiation model and requires certain input files and options to operate properly. The interface between MIMICA and the radiation model is provided by *radiag.f90* where the local data computed by the LES are stored in 1D arrays in pressure dimension. These arrays constitute the input data required by *rad_ir* and *rad_vis* the two main routines calculating respectively the longwave and shortwave radiative fluxes. The model requires the pressure array, the absolute temperature, the water vapor mixing ratio, the mass mixing ratios as well as the effective radiuses of ice particles and cloud droplets, and finally ozone soundings that can be provided by the CHEMISTRY module. The solver provides in return the LW and SW upward and downward fluxes that have to be summed and derived to determine the net heating/cooling rates. Note that the McICA option, defined by default to FALSE in *radiag.f90*, controls the multiband option of the solver: when turned on TRUE, the model solves for a single band instead of all the defined ones and extrapolates the results for the others. This allows to considerably improve the overall code performances.

The radiation solver works on 1D columns extending up to the tropopause. Because the LES domain is limited to about 1500-2000 m in altitude, the input data arrays must be completed using standard data above the LES domain. Standard atmospheres are stored in external .lay files located in the DATA directory (f.i. *kmlw.lay*). The most appropriate standard atmosphere must then be copied in the local working directory and renamed *sounding_rad.dat*. The user must be careful when using one of the predefined files and must make sure that the standard and LES data match properly at the domain top (no sharp jump in temperature, no pressure inversion...). It is of course possible for the user to build its own customized sounding file. In addition to the standard atmosphere, the radiation solver also requires two additional tables located in DATA containing informations used by the CKD (correlated k-distribution) model: *ckd.dat* and *cldwtr.dat*.

In addition to the 1D soundings extrapolated up to the tropopause, surface parameters must also be provided as inputs of the radiation solver. This includes the surface albedo and surface skin temperature, both prescribed in *cm.nml*, as well as the surface emissivity which is hard coded and set to 0.9 in *radiag.f90*. To finish with, the solar zenith angle is computed at each radiation step at the beginning of *radiag.f90*. The formula uses the prescribed latitude, the julian day, the time reference for the simulation with respect to UTC and the instantaneous time in seconds. All the necessary parameters must be set in *cm.nml*.

The second simplistic method defines the net radiative fluxes as a function of LWP only:

$$F_{rad}\left(z\right) = F_0 \exp\left[-k\left(LWP\left(z_t\right) - LWP\left(z\right)\right)\right] + F_1 \exp\left[-kLWP\left(z\right)\right] + F_3\left(z\right), \tag{32}$$

with z_t the cloud top height, and

$$LWP\left(z\right) = \int_0^z \rho_0\left(z'\right) q_l\left(z'\right) dz'. \tag{33}$$

The coefficients F_0, F_1 and k are case dependent and are usually fitted to detailed radiation calculations performed in similar conditions as the considered case. $F_3\left(z\right)$ is an additional function of altitude sometimes present to compensate for cooling due to large scale subsidence above cloud top.

Radiative heating tendencies are finally obtained following:

$$\mathcal{Q}_{rad} = -\frac{1}{\rho_0 C_p}\frac{\partial F_{rad}}{\partial z}. \tag{34}$$

2.5 Microphysics: a two moment bulk scheme

2.5.1 Basic definitions

A two-moment bulk microphysics scheme is implemented in the code for all the hydrometeors (including cloud droplets) where assumed size distributions are used, taking the form of regular gamma functions denoted $\Gamma\left(x\right)$. They are expressed as:

$$dN = N_0 D^{\alpha} e^{-\lambda D} dD, \tag{35}$$

where N is the number concentration of a given hydrometeor population having diameter D, α is a constant depending on the hydrometeor type, and N_0 and λ are two parameters which can be expressed in terms of N and Q the mass mixing ratio.

$$N_0 = \frac{N\lambda^{\alpha+1}}{\Gamma\left(\alpha+1\right)} \qquad \text{and} \qquad \lambda = \left(\frac{A_m \Gamma\left(\alpha+\beta_m+1\right) N}{\Gamma\left(\alpha+1\right) Q}\right)^{1/\beta_m}. \tag{36}$$

The parameters A_m and β_m are the constants in the particle's mass-size relation given by:

$$m = A_m D^{\beta_m}. \tag{37}$$

One important property of Γ that will be used later on must be introduced here: $\Gamma\left(x+1\right) = x\Gamma\left(x\right)$.

N and Q are defined by:

$$N = \int_0^{+\infty} f\left(D\right) dD \qquad \text{and} \qquad Q = \int_0^{+\infty} m f\left(D\right) dD, \tag{38}$$

with $f\left(D\right) = \frac{dN}{dD}$. Therefore, N is considered to be the 0th moment of distribution 35 while m is the β_mth moment. The mean diameter of a given hydrometeor population, which will be used later, is the first moment of 35: $\overline{D} = \int_0^{+\infty} D f\left(D\right) dD$. The values of coefficients α, β_m and A_m for all the hydrometeor kinds defined in the model are summarized in table 1. Note that species terminal fall speeds, V_p, are modeled using simple power laws of the diameter as well (the coefficients are gathered in table 1):

$$V_p = A_v D^{\beta_v}\left(\frac{\rho_0}{\rho}\right)^{\gamma}. \tag{39}$$

Cloud droplets are commonly assumed not to precipitate so that $A_{v,c} = 0$.

At the moment, five types of hydrometeors are described by the model: cloud droplets, rain drops, cloud ice, graupel and snow. In the following, quantities refering to a specific type of hydrometeor will be denoted by the initials c, r, i, g or s.

Ice crystals may take different shapes depending on the thermodynamic conditions under which they were nucleated. At the moment, only one type of ice crystal habit can be assumed at a time for each simulation. The proper ice habit has to be selected prior to the beginning of the calculation in *shared_hydro.f90* where a list of ice microphysical parameters is provided (hard coded). All the default ice crystal types are summarized in table 1.

The choice of the microphysics level is controled by the integer lmicro in *cm.nml*: lmicro = 0 corresponds to a dry case where no condensed water is present (no microphysics calculated), for lmicro = 1, only warm microphysics is modeled (cloud drops and rain, drizz in *cm.nml* even allows to swith on and off the formation of precipitating rain drops), for lmicro = 2, ice microphysics is included but only for pristine ice crystals (no riming and no aggregation) whereas lmicro = 3 includes the formation of snow flakes and graupel (the 5 hydrometeor species, $nh = 5$).

nh and *nhydro* must not be confused: *nhydro* is the total possible number of hydrometeors which is used to allocate and initialize all the arrays depending on microphysics, whereas *nh* is the number of modeled hydrometeors and is used for the calculations.

hydrometeor	α	βm	A_m	β_v	A_v
cloud (c)	5	3	523.6	-	-
rain (r)	0	3	523.6	0.8	836.
ice (i, rosette C2a)	2	2.26	0.102	1.225	6059.5
ice (i, columns C1f)	2	1.8	0.0093	0.488	30.14
ice (i, plates P1c)	2	2.79	1.43	0.62	17.9
ice (i, dendrites P1e)	2	2.29	0.0233	0.48	5.02
ice (i, assemblage of den. P7b)	2	2.68	0.35	0.83	62.62
graupel (g)	0	3	65	0.8	199.05
snow (s)	1	2	0.04	0.333	6.962

Table 1: *Coefficients for the microphysics scheme for all five kinds of hydrometeors. Parameters for ice crystals are taken from Propacher and Klett [18] and Khvorostyanov and Curry [14]. Units are SI (kg-m-s).*

2.5.2 Equivalence with mass distributions

It is sometimes convenient to describe hydrometeor populations in terms of mass distributions instead of size. Combining relations 37 and 35 yields a distribution under the form of a generalized gamma distribution:

$$dN = M_0 m^{\nu} e^{-\Lambda m^{\mu}} dm, \tag{40}$$

where M_0 and Λ are the new parameters that need to be redefined, and ν and μ two constants dependent on the type of hydrometeor. Comparing the mass and size distributions the following equalities come:

$$\begin{aligned} M_0 &= \frac{N_0}{\beta A_m^{\frac{\alpha+1}{\beta_m}}}, & (41)\\ \Lambda &= \frac{\lambda}{A_m^{1/\beta_m}}, & (42)\\ \nu &= \frac{\alpha+1}{\beta_m} - 1, & (43)\\ \mu &= \frac{1}{\beta_m}. & (44)\end{aligned}$$

The kth moment of this generalized distribution is given by:

$$\begin{aligned} M^{(k)} &= \int_0^{+\infty} m^k f(m)\, dm & (45)\\ &= M_0 \int_0^{+\infty} m^{k+\nu} e^{-\Lambda m^{\mu}} dm & (46)\\ &= M_0 \frac{\Gamma\left(\frac{\nu+k+1}{\mu}\right)}{\mu} \Lambda^{-\frac{\nu+k+1}{\mu}}. & (47)\end{aligned}$$

Note that using relation 47, expressions for M_0 and Λ as functions only of N, Q and the parameters ν and μ can be derived:

$$M_0 = \frac{\mu N}{\Gamma\left(\frac{\nu+1}{\mu}\right)} \Lambda^{\frac{\nu+1}{\mu}} \qquad \text{and} \qquad \Lambda = \left[\frac{\Gamma\left(\frac{\nu+1}{\mu}\right) Q}{\Gamma\left(\frac{\nu+2}{\mu}\right) N}\right]^{-\mu}. \tag{48}$$

Replacing the ν and μ by their corresponding expressions depending on α, A_m and β_m, we find:

$$M_0 = \frac{N}{\beta \Gamma(\alpha+1)} \Lambda^{\alpha+1} \qquad \text{and} \qquad \Lambda = \left[\frac{\Gamma(\alpha+1) Q}{\Gamma(\alpha+\beta+1) N}\right]^{-1/\beta}. \tag{49}$$

2.5.3 Collisions between hydrometeors

The collisional sources and sinks are obtained by integrating a collection kernel $K(m, m^*)$ over two different hydrometeor populations having masses m and m^*. The time variation of a mass distribution $f(m)$ due to collision processes can be derived, commonly called the Stochastic Collection Equation:

$$\frac{\partial f\left(m\right)}{\partial t}=\frac{1}{2}\int_{0}^{m}f\left(m-m^{*}\right)f\left(m^{*}\right)K\left(m-m^{*},m^{*}\right)dm^{*}-\int_{0}^{+\infty}f\left(m\right)f\left(m^{*}\right)K\left(m,m^{*}\right)dm^{*}. \tag{50}$$

Introducing the kth moment $M^{(k)}$ of m in the previous relation, an equation for the evolution of $M^{(k)}$ can be found [20]. From this general equation (not shown here), and after further simplifications, some simpler expressions can be derived giving the rate of change of the particle's number densities and mass mixing ratios due to collision/coalescence events.

$$\frac{\partial N}{\partial t}=\int_{0}^{+\infty}f\left(m\right)\left[\int_{0}^{+\infty}f\left(m^{*}\right)K\left(m,m^{*}\right)dm^{*}\right]dm, \tag{51}$$

and:

$$\frac{\partial Q}{\partial t}=\int_{0}^{+\infty}mf\left(m\right)\left[\int_{0}^{+\infty}f\left(m^{*}\right)K\left(m,m^{*}\right)dm^{*}\right]dm, \tag{52}$$

The previous equations represent the evolution of the number and mass of a given hydrometeor species Ψ due to collisions with particles Ψ^{*} belonging to another species. These relations can be derived without any assumption on the collection kernel or on the mass distribution.

Given that f is usually assumed to take the shape of gamma distribution, the only unknown in the previous equations is the collection kernel K which needs to be presumed. The original model uses the gravitational kernel which reads:

$$K\left(D,D^{*}\right)=\rho E^{*}\frac{\pi}{4}\left(D+D^{*}\right)^{2}\left|V_{p}-V_{p}^{*}\right|. \tag{53}$$

The coefficient E^{*} represents the collision efficiency and usually depends on the considered microphysical species and local conditions. This modelling assumption is still in use in the mixed-phase microphysics part of the LES. Warm microphysics processes are however now modeled using Seifert and Beheng's approach which makes use of a slightly different kernel as shown in the next section.

Provided that the SCE is given in terms of particle masses, it may not be adequate to represent an hydrometeor population by its size distribution. This issue is however commonly overcome by simply assuming that the previous relations for the mass and number microphysical sources can directly be extended to a size distribution. The correspondance between the two relations may not be perfectly exact as mass and diameter are related by a linear power law formula in such a way that additional terms or factors should be included. The assumption seems nevertheless acceptable and will therefore be used in the following.

2.5.4 Warm microphysics: Seifert & Beheng's scheme

Warm microphysics (involving only liquid water) will be treated using Seifert and Beheng's model [20, 21]. The microphysics parameterization uses a double moment approach for both cloud droplets and rain where mass distributions of these two species are represented by generalized gamma functions. The approach chosen by Seifert and Beheng is thus slightly different from the framework introduced previously, but the same assumptions still apply.

The authors use a simple kernel defined by a picewise polynome (Long's kernel):

$$K\left(m,m^{*}\right) = k_{c}\left(m^{2}+m^{*2}\right) \tag{54}$$

$$K\left(m,m^{*}\right) = k_{r}\left(m+m^{*}\right). \tag{55}$$

The first polynome is used when the mass of the collector m is smaller than a critical mass m_c separating cloud drops and rain drops. This mass is calculated from the separation diameter between cloud droplets and rain prescribed by the parameter *drmin* in *cm.nml*. Cloud drops can only collect smaller cloud drops so that the first polynome will only represent cloud/cloud interactions. The second one is used when the collector's mass is greater than m_c, which corresponds to rain/cloud or rain/rain interactions. The constants k_c and k_r are given the values $9.44e9$ cm3/g2/s and $5.78e3$ cm3/g/s respectively.

Integrating the mass distributions following relations 51, and using the above kernels leads to the exact expressions for the rates of autoconversion (c+c $\rightarrow$ r), accretion (r+c $\rightarrow$ r) and selfcollection (c+c $\rightarrow$ c or r+r $\rightarrow$ r). These rates can be found in [20, 21] and are recalled hereafter:

$$\left.\frac{\partial Q_{r}}{\partial t}\right|_{auto}=auq=\frac{k_{c}}{20m_{c}}\frac{\left(\nu_{c}+2\right)\left(\nu_{c}+4\right)}{\left(\nu_{c}+1\right)\left(\nu_{c}+1\right)}Q_{c}^{2}\overline{m}_{c}\left[1+\frac{\Phi_{a}\left(\tau\right)}{\left(1-\tau\right)^{2}}\right], \tag{56}$$

$$\left.\frac{\partial N_r}{\partial t}\right|_{auto} = aun = \frac{1}{m_c}\left.\frac{\partial Q_r}{\partial t}\right|_{auto} \tag{57}$$

$$\left.\frac{\partial Q_c}{\partial t}\right|_{auto} = -auq \tag{58}$$

$$\left.\frac{\partial N_c}{\partial t}\right|_{auto} = -2 \times aun \tag{59}$$

$$\left.\frac{\partial Q_r}{\partial t}\right|_{acc} = clr = k_r Q_c Q_r \Phi_{ac}(\tau) \tag{60}$$

$$\left.\frac{\partial N_r}{\partial t}\right|_{acc} = 0 \tag{61}$$

$$\left.\frac{\partial Q_c}{\partial t}\right|_{acc} = -clr \tag{62}$$

$$\left.\frac{\partial N_c}{\partial t}\right|_{acc} = clrn = -\frac{1}{\overline{m}_c} clr \tag{63}$$

$$\left.\frac{\partial Q_r}{\partial t}\right|_{sc} = 0 \tag{64}$$

$$\left.\frac{\partial N_r}{\partial t}\right|_{sc} = crrn = -k_r N_r Q_r \Phi_b(\tau) \tag{65}$$

$$\left.\frac{\partial Q_c}{\partial t}\right|_{sc} = 0 \tag{66}$$

$$\left.\frac{\partial N_c}{\partial t}\right|_{sc} = cccn = -k_c \frac{(\nu_c + 2)}{(\nu_c + 1)} Q_c^2 + 2 \times aun \tag{67}$$

where ν_c corresponds to the mass exponent in the mass distribution for cloud droplets. Seifert and Beheng suggest that this parameter should be set to 1, which is equivalent to $\alpha = -1/3$ in a size distribution (considering that $\beta_m = 3$). To remain consistent with table 1 where $\alpha = 5$, a value of $\nu_c = 0$ will be chosen (set in *shared_data*).

In the above, $\overline{m} = Q/N$ is a mean mass, $\tau = 1 - \frac{Q_c}{Q_c + Q_r}$ is a dimensionless scaling parameter, and Φ_a and Φ_b are autoconversion, accretion and break-up rates defined by Seifert and Beheng [21] using similarity assumptions as:

$$\Phi_a = 400\tau^{0.7}\left(1 - \tau^{0.7}\right)^3 \tag{68}$$

$$\Phi_{ac} = \left(\frac{\tau}{\tau + 5e - 5}\right)^4 \tag{69}$$

$$\Phi_b = k_b\left(\overline{D}_r - D_{eq}\right) \quad \text{when} \quad 0.35mm < \overline{D}_r < D_{eq} \tag{70}$$

$$\Phi_b = 2\exp\left[k_b\left(\overline{D}_r - D_{eq}\right)\right] - 1 \quad \text{when} \quad \overline{D}_r > D_{eq} \tag{71}$$

with $k_b = 1000$ 1/m and D_{eq} an equilibrium diameter (corresponding to a balance between break-up and selfcollection) set to 0.9 mm. Φ_b is set to 0 when $\overline{D}_r < 0.35$ mm.

2.5.5 mixed phase microphysics

Mixed-phase microphysics is treated using a similar parameterization as in the original CRM model. Three ice microphysical species are described (ice, graupel and snow) and modeled via their size distributions. The gravitational kernel 53 is used to represent their interactions with any other hydrometeor (including liquid drops).

The collision rates between two hydrometeors, 1 collecting 2, for cold microphysics are given by:

$$P_{n12} = \frac{\pi}{4}\rho E_{12}\int_0^{+\infty}\int_0^{+\infty} f(D_1)\,f(D_2)\,(D_1+D_2)^2\,|V_{p1}-V_{p2}|\,dD_1 dD_2, \tag{72}$$

$$P_{q12} = \frac{\pi}{4}\rho E_{12}\int_0^{+\infty}\int_0^{+\infty} D_2 f(D_1)\,f(D_2)\,(D_1+D_2)^2\,|V_{p1}-V_{p2}|\,dD_1 dD_2, \tag{73}$$

After tidious manipulations and simplifications, one can write the general form of the collision rates:

$$\begin{aligned}P_{n12} = \frac{\pi}{4}\rho E_{12}N_1N_2 \Bigg[&\left\langle D_1^2\right\rangle \left|\frac{(\alpha_1+\beta_{v1}+2)(\alpha_1+\beta_{v1}+1)}{(\alpha_1+2)(\alpha_1+1)}\left\langle V_{p1}\right\rangle - \left\langle V_{p2}\right\rangle\right| \\ &+2\left\langle D_1\right\rangle\left\langle D_2\right\rangle\left|\frac{\alpha_1+\beta_{v1}+1}{\alpha_1+1}\left\langle V_{p1}\right\rangle - \frac{\alpha_2+\beta_{v2}+1}{\alpha_2+1}\left\langle V_{p2}\right\rangle\right| \\ &+\left\langle D_2^2\right\rangle\left|\left\langle V_{p1}\right\rangle - \frac{(\alpha_2+\beta_{v2}+2)(\alpha_2+\beta_{v2}+1)}{(\alpha_2+2)(\alpha_2+1)}\left\langle V_{p2}\right\rangle\right|\Bigg]\end{aligned} \tag{74}$$

$$\begin{aligned}P_{q12} = \frac{\pi}{4}\rho E_{12}Q_1N_2 \Bigg[&\left\langle D_1^2\right\rangle_m \left|\frac{(\alpha_1+\beta_{v1}+\beta_{m1}+2)(\alpha_1+\beta_{v1}+\beta_{m1}+1)}{(\alpha_1+\beta_{m1}+2)(\alpha_1+\beta_{m1}+1)}\left\langle V_{p1}\right\rangle_m - \left\langle V_{p2}\right\rangle\right| \\ &+2\left\langle D_1\right\rangle_m\left\langle D_2\right\rangle\left|\frac{\alpha_1+\beta_{v1}+\beta_{m1}+1}{\alpha_1+\beta_{m1}+1}\left\langle V_{p1}\right\rangle_m - \frac{\alpha_2+\beta_{v2}+1}{\alpha_2+1}\left\langle V_{p2}\right\rangle\right| \\ &+\left\langle D_2^2\right\rangle\left|\left\langle V_{p1}\right\rangle_m - \frac{(\alpha_2+\beta_{v2}+2)(\alpha_2+\beta_{v2}+1)}{(\alpha_2+2)(\alpha_2+1)}\left\langle V_{p2}\right\rangle\right|\Bigg]\end{aligned} \tag{75}$$

The notations $\langle . \rangle$ and $\langle . \rangle_m$ respectively denote an ensemble mean over the entire hydrometeor population and a mass weighted ensemble mean. Elementary functions were created in MIMICA to calculate the mean diameters and terminal fall speeds as well as any other useful distribution averaged qantity. P_{q12} represents the mass tendency of species 1 when collected by species 2. It is also possible in a similar manner to define the mass tendency of species 2 following the same process.

Two major simplifications can be obtained in the case where two identical hydrometeors interact (selfcollection) or when the collected particle is a cloud droplet (riming). In the latter case, two assumptions can be made: that the cloud droplet terminal fall speed is 0, and that their size or mass is small compared to the collector particle. We can then rewrite P_{n11} as well as P_{n1c} under the following forms (similar simplifications are obtained for P_q):

$$P_{n11} = \frac{\pi}{2}\rho E_{11}N_1^2\left\langle D_1^2\right\rangle\left\langle V_1\right\rangle\left|\frac{(\alpha_1+\beta_{v1}+2)(\alpha_1+\beta_{v1}+1)}{(\alpha_1+2)(\alpha_1+1)} - 1\right|, \tag{76}$$

(during selfcollection, the mass of 1 is conserved in such a way that $P_{q11} = 0$)

$$P_{n1c} = \frac{\pi}{4}\rho E_{1c}N_1N_c\frac{(\alpha_1+\beta_{v1}+2)(\alpha_1+\beta_{v1}+1)}{(\alpha_1+2)(\alpha_1+1)}\left\langle D_1^2\right\rangle\left\langle V_{p1}\right\rangle. \tag{77}$$

These general relations can be applied to any type of hydrometeor provided that the coefficients for the mass and terminal fall speed formula are known.

For most of the interactions, the preceding relations are applied as such. All the possible interactions treated in the code are summarized in table 2. A total of 17 processes are modeled in the present version. Some collision processes present particularities. Riming of ice crystals and snow (resulting from the collision between ice/snow and cloud droplets) is for instance assumed to occur only for cloud droplets having a diameter larger than 15 mm (see [2]). In these cases, a partial mixing ratio of cloud droplets effectively rimmed is calculated and multiplied to the integral collection rates. This allows to exclude part of the cloud droplet spectrum from the rimming process. Similarly, autoconversion of ice particles to graupel will occur if the ice particle diameter exceeds a threshold $\hat{D}_i = 0.3$ mm. This transition can only result from deposition growth. In that case, only partial integrals must be calculated (see [2]).

hydrometeor types	cloud drops	rain	ice crystals	graupel	snow
cloud drops	c + c → c/r auq, aun, cccn 56, 57, 66, 67	c + r → r clr, clrn 63, 62	c + i → g cli, clii, clin 77	c + g → g clg, clgn 77	c + s → s cls, clsn 77
rain		r + r → r crrn (inc. break-up) 65	r + i → g cir, cirn 74, 75	r + g → g crg, crgn 74, 75	r + s → g csr, csrr, csrn 74, 75
ice crystals			i + i → s i → g (autoconv.) ais, aisn 76, aig, aign	i + g → g - efficiency = 0	i + s → s cis, cisn 74, 75
graupel				g + g → g - rebound	g + s → g csg, csgn 74, 75
snow					s + s → s cssn 76

Table 2: *Interactions between hydrometeors treated in the code.*

2.5.6 Precipitation

Precipitation fluxes appearing in equation 20 include a mean terminal fall speed for each hydrometeor kind, defined as a mass weighted average over the entier hydrometeor population. The power law 39 defines the terminal fall speed of an hydrometeor having a given mass m, including a correction term accounting for vertical density variations. The mass weighted mean terminal fall speeds are thus defined as:

$$\langle V_p \rangle = \frac{\int_0^{+\infty} m V_p(D) f(D) \, dD}{\int_0^{+\infty} m f(D) \, dD}. \tag{78}$$

After simplifications, it comes:

$$\langle V_p \rangle = A_v \frac{\Gamma(\alpha + \beta_v + \beta_m + 1)}{\Gamma(\alpha + \beta_m + 1)} \lambda^{-\beta_v} \left(\frac{\rho_0}{\rho} \right)^{\gamma}. \tag{79}$$

2.5.7 Phase transitions

Here is defined how phase transitions are treated in the model: evaporation/condensation, sublimation/deposition, melting/freezing. Following Pruppacher and Klett [18], the mass tendency of a given droplet of diameter D due to evaporation/condensation is given by:

$$\left. \frac{\partial m}{\partial t} \right|_{e/c} = 2\pi D G_{lv}(T, p) F_v \frac{s}{q_s}, \tag{80}$$

with:

$$G_{lv}(T, p) = \left[\frac{R_w T}{e_s D_v} + \frac{L_{lv}}{K_T T} \left(\frac{L_{lv}}{R_w T} - 1 \right) \right]^{-1} \tag{81}$$

$s = q_v - q_s$ is the absolute supersaturation, q_s the saturation mixing ratio over liquid water, e_s the saturation vapor pressure over liquid water, R_w the ideal gas specific constant for water vapor, D_v the diffusion coefficient for water vapor ($\approx 3.e-5$ m2/s), K_T the heat conductivity ($\approx 2.5e-2$ Jm/s/K). The ventilation factor F_v is defined following Chen and Lamb [3] as a function of the Best number $X = Sc^{1/3} Re^{1/2}$:

$$F_v = a_v + b_v X^{\gamma}. \tag{82}$$

The Schmidt number Sc is assumed to be constant and equal to 0.7 whereas the Reynolds number Re is defined by:

$$Re = \frac{\rho_0 V_p D}{\mu}, \tag{83}$$

with V_p the drop terminal fall speed and μ the dynamic viscosity of air ($\approx 1.5e-5$ kg/m/s). For $X \leq 1$, Chen and Lamb suggest that $\gamma = 2$, $a_v = 1$ and $b_v = 0.14$ whereas when $X > 1$, $\gamma = 1$, $a_v = 0.86$ and $b_v = 0.28$ for ice crystals and $\gamma = 1$, $a_v = 0.78$ and $b_v = 0.308$ otherwise. Ventilation effects are neglected for cloud droplets so that F_v is set to 1. For the other hydrometeor species, it is possible to turn on the ventilation effects by setting lvent to 1 in *cm.nml* (these effects are not included by default but might have a large impact on ice crystal growth).

In order to obtain condensation/evaporation sources for the entire drop population, one has to integer relation 97 over the assumed size distribution. To simplify integration, we assume that the final total mass tendency will take the form:

$$\left.\frac{\partial Q}{\partial t}\right|_{e/c} = 2\pi G_{lv}(T,p)\frac{s}{q_s}\int_0^{+\infty} DF_v f(m)\,dm \tag{84}$$

$$= 2\pi N G_{lv}(T,p)\langle D\rangle F_v^*\frac{s}{q_s} \tag{85}$$

where we defined a modified ventilation factor depending on an averaged Reynolds number:

$$F_v^* = a_v + b_v Sc^{1/3}\langle Re\rangle^{1/2} \tag{86}$$

$$= a_v + b_v^* Sc^{1/3}\left(\frac{\beta_v+\alpha+1}{\alpha+1}\frac{\langle V\rangle\langle D\rangle}{\nu}\right)^{1/2}. \tag{87}$$

The mass weighted terminal fall speeds are defined previously and b_v^* is a modified factor, not detailed here.

Similar relations are found for sublimation/deposition processes applied to ice crystals, graupel and snow flakes:

$$\left.\frac{\partial m}{\partial t}\right|_{e/c} = 4\pi C_i D G_{iv}(T,p) F_v\frac{s_i}{q_{si}}, \tag{88}$$

and:

$$G_{iv}(T,p) = \left[\frac{R_w T}{e_{si}D_v} + \frac{L_{iv}}{K_T T}\left(\frac{L_{iv}}{R_w T}-1\right)\right]^{-1}, \tag{89}$$

which yields the following total mass tendencies for the solid hydrometeor populations:

$$\left.\frac{\partial Q}{\partial t}\right|_{s/d} = 4\pi N C_i G_{iv}(T,p)\langle D\rangle F_v^*\frac{s_i}{q_{si}}, \tag{90}$$

C_i is the ice crystal capacitance. It takes the value 1/2 for spherical particles (graupel) and $1/\pi$ for other shapes. The assumption of a constant capacitance is actually very approximate and more advanced methods with a capacitance depending on crystals shape (through their aspect ratio for example) are more accurate.

The integration of condensation/evaporation sources for water droplets may sometimes lead to numerical instability issues if no specific treatment is applied. Most of the time, these problems lead to the apparition of spurious water mixing ratio peaks at the cloud boundaries where in-cloud air parcels are mixed with the ambient dry atmosphere. Direct integration of the condensation/evaporation sources are nonetheless necessary in two particular cases (even though simpler approaches can be used but with a dramatic drop in model accuracy): when three-phase microphysics is used so that water vapor will interact both with liquid drops and ice crystals, and when droplet activation is modeled, especially with prognostic aerosols. These issues and the solution adopted in MIMICA are discussed in the following sections.

Melting of ice particles is treated in a similar manner as diffusional growth, with the mass tendency of a single particle given by:

$$\left.\frac{\partial m}{\partial t}\right|_{melt} = \frac{2\pi}{L_{il}}\left[K_T(T-T_0)F_h + \frac{D_v}{L_{lv}}\left(\frac{p_v}{T}-\frac{p_s}{T_0}\right)F_v\right]D. \tag{91}$$

F_h is a heat ventilation coefficient approximated by $F_h = Le_v F_v$ with $Le_v \approx 0.67$ the Lewis number for water vapor, and $T_0 = 273.15$ K is the freezing temperature. Integrating over the particle population gives:

$$\left.\frac{\partial Q}{\partial t}\right|_{melt} = \frac{2\pi}{L_{il}}N\left[K_T(T-T_0)Le_v + \frac{D_v}{L_{lv}}\left(\frac{p_v}{T}-\frac{p_s}{T_0}\right)\right]F_v^*\langle D\rangle, \tag{92}$$

with F_v^* defined as previously. All the melted water is considered to be rain.

In the above, positive tendencies mean growth of drops or ice particles via condensation or deposition of water vapor. Negative tendencies mean a loss of ice or liquid water via sublimation or evaporation. Whereas in the first case the number concentration of hydrometeors is left unchanged (the existing particles only grow), evaporation and sublimation lead to a decrease in droplet and ice crystal number concentrations. These variations of number concentration for ice particles and liquid drops due to sublimation and evaporation are expressed following:

$$\left.\frac{\partial N}{\partial t}\right|_{e/s} = \frac{N}{Q} \min\left(\left.\frac{\partial Q}{\partial t}\right|_{e/s}, 0 \right). \tag{93}$$

Here, the subscripts e/s mean evaporation or sublimation as we consider both cloud droplets and ice crystals. The mass mixing ratio tendencies on the right hand side are limited by 0 as number concentrations are only allowed to decrease through evaporation/sublimation.

2.5.8 Saturation adjustment

When a detailed knowledge of supersaturation is not required, the saturation adjustment method can be employed, thus overcoming the need to integrate condensation/evaporation sources. The saturation adjustment assumption comes from the observation that in most stratiform clouds, in-cloud water relaxes very quickly to equilibrium. In other words, omitting CCN activation, it can be assumed that at any time, the in-cloud atmosphere is saturated, $q_v = q_s$ or $s = 0$, so that all the water vapor available above saturation condenses instantaneously on the existing drops. Stricly following this assumption, the liquid water mixing ratio can be diagnosed given q_v and q_s. The method is selected by setting the option SAT_ADJ to TRUE in *start*.

Let's note q_l^n and q_v^n the quantities at time t^n, before adjustment, and q_l^* and q_v^* the same quantities after adjustment. Knowing that $q_t = q_v + q_l$, the total water content, is conserved, we can write:

$$q_l^* = q_t^n - q_s\left(P, T^*\right). \tag{94}$$

Here, q_s is expressed following Rogers and Yau [19]:

$$q_s = \frac{380.1}{P} \exp\left(\frac{T - 273.15}{T - 29.65}\right). \tag{95}$$

The liquid water mixing ratio can be introduced in the previous relation by introducing the liquid potential temperature: $T = \Pi\Theta_l + \frac{L_{lv} q_l}{c_p}$. Relation 94 is then solved using Newton-Raphson iterations while updating q_l, q_v, T and Θ_l after each step. The system to be solved can be written:

$$q_l^{m+1} = q_l^m + \frac{q_t - q_l^m - q_s\left(P, T^m\right)}{1 + \frac{\partial q_s\left(P, T^m\right)}{\partial q_l^m}}, \tag{96}$$

where m denotes the current Newton step. Iterations are stopped when $\left|q_l^{m+1} - q_l^m\right| < \epsilon$, and we set $q_l^* = q_l^{m+1}$.

When the saturation adjustment scheme is applied, only the mixing ratio of cloud water is updated. It is thus assumed that all the water vapor in excess condenses over cloud droplets only. Rain mixing ratio is thus taken into account to evaluate the total water mixing ratio necessary for the resolution of the above system, but it is not modified after the procedure is completed.

When considering mixed-phase clouds, the method can still be applied with minor biases in liquid dominated clouds but large errors can appear when the ice/liquid equilibrium lies far from saturation over liquid, that is when the ice proportion within the cloud layer becomes large. In such conditions, an explicit treatment of condensation/deposition sources is required.

Saturation adjustment schemes for mixed-phase environments have been proposed in the litterature, but none of these solutions yield reasonable results. They usually fail to predict the right repartition between liquid and ice phase. As a result, when mixed-phase clouds are to be simulated using MIMICA, the direct integration of all condensation/sublimation sources is employed.

Note that the saturation adjustment scheme is always used prior to the first time step to initiate the cloud layer. The initial cloud cover/liquid water field is indeed usually diagnosed from the input potential temperature and total water mixing ratio soundings using the saturation adjustment method.

2.5.9 Direct integration of condensation/evaporation

When condensation/evaporation sources (respectively deposition/sublimation) are explicitly integrated, the implicit dependence of the diffusional sources on supersaturation over water and ice may be the source of numerical issues. The diffusional growth terms are simplified for the present demonstration:

$$C_k = \frac{s}{\tau_{lk}} \qquad\qquad (D_k = \frac{si}{\tau_{ik}}), \tag{97}$$

with $s = q_v - q_s$ ($si = q_v - q_{si}$) being the supersaturation over liquid water (ice), and τ_{lk} (τ_{ik}) are characteristic time scales of condensation/evaporation (deposition/sublimation) processes that can be deduced from relations given in 2.5.7. A detailed resolution of supersaturation evolution, which is necessary for a proper representation of condensation/evaporation processes, requires time-steps of the order of milliseconds. As this is not conceivable in LES of atmospheric flows, specific alternatives must be found to preserve the system stability as well as the accuracy of the solution.

In the present LES code, the strong implicit relation between supersaturation and condensation/evaporation is modeled using a pseudo-analytic solution of the supersaturation equation over a time step, as proposed for example by Morrison and Grabowski [17]. For the purpose of the demonstration, we simplify the entire system to account only for diffusional growth (mixing and advection were neglected):

$$\frac{\partial q_l}{\partial t} = \frac{s}{\tau_l} \tag{98}$$

$$\frac{\partial q_i}{\partial t} = \frac{si}{\tau_i} \tag{99}$$

$$\frac{\partial q_v}{\partial t} = -\frac{s}{\tau_l} - \frac{si}{\tau_i} \tag{100}$$

The system can be completed by the following relations:

$$\frac{\partial q_s}{\partial t} = \frac{\partial q_s}{\partial T}\frac{\partial T}{\partial t} = -A. \tag{101}$$

$$\frac{\partial q_{si}}{\partial t} = \frac{\partial q_{si}}{\partial T}\frac{\partial T}{\partial t} = -B. \tag{102}$$

where we define $\frac{\partial T}{\partial t}$, a temperature tendency term containing radiation, turbulent mixing as well as advection through updrafts/downdrafts. Combining the previous relations, it is possible to derive balance equations for s and si. The equation for s and si can be written as follows:

$$\frac{\partial s}{\partial t} = -\frac{s}{\tau} + \frac{1}{\tau_i}(s - s_i) + A \tag{103}$$

$$\frac{\partial si}{\partial t} = -\frac{si}{\tau} - \frac{1}{\tau_c}(s - s_i) + B, \tag{104}$$

with the characteristic phase relaxation time:

$$\tau = \left(\frac{1}{\tau_l} + \frac{1}{\tau_i}\right)^{-1}, \tag{105}$$

Equation 103 (respectively 104) can be integrated under the assumption that τ_c, τ_i and A remain constant during time t:

$$s(t) = \tau\left[\frac{s_i}{\tau_i} + \frac{s}{\tau_c} - \tau\left(\frac{B}{\tau_i} + \frac{A}{\tau_c}\right)\right]e^{-\frac{t}{\tau}} + \tau^2\left(\frac{B}{\tau_i} + \frac{A}{\tau_c}\right) + \frac{\tau}{\tau_i}\left[s - s_i + (A - B)\,t\right]. \tag{106}$$

A very similar relation can be found for si. Compared to the work of Morrison et al. [16], the dependence of $s - s_i$ on time has been considered. This yields a more complicated but more accurate solution of the supersaturation equation.

Condensation/evaporation (sublimation/deposition) sources are now evaluated using a time averaged supersaturation integrated over a timestep. Integrating relation 106 over Δt gives

$$\begin{aligned} \overline{s} &= \frac{1}{\Delta t}\int_t^{t+\Delta t} s(t^*)\,dt^* && (107)\\ &= \frac{\tau^2}{\Delta t}\left[\frac{s_i}{\tau_i} + \frac{s}{\tau_c} - \tau\left(\frac{B}{\tau_i} + \frac{A}{\tau_c}\right)\right]\left(1 - e^{-\frac{\Delta t}{\tau}}\right) + \tau^2\left(\frac{B}{\tau_i} + \frac{A}{\tau_c}\right) + \frac{\tau}{\tau_i}\left[s - s_i + \frac{\Delta t}{2}(A - B)\right]. && (108)\end{aligned}$$

The mean supersaturation obtained above is then used to calculate the condensation/evaporation sources for liquid drops whereas a mean ice supersaturation (taking a similar form) is used to evaluate the deposition/sublimation sources of all ice particles.

Although quite complicated, the method provides a very stable and accurate way of integrating condensation/evaporation sources when a detailed evolution of supersaturation is required. The issue related to supersaturation integration in high resolution models appears to be a non trivial one and finding efficient methods to overcome the issue (not mentioning the saturation adjustment scheme) is not a simple task.

The model is selected by setting lgrowth to 1 in *cm.nml*. If lgrowth is turned to 0, supersaturation for diffusional growth is evluated explicitly at time t.

2.5.10 Droplet activation

When the chemistry/aerosol modules are deactivated, an empirical parameterization of the number of activated CCN is chosen, based on a power law of the supersaturation:

$$N_{CCN} = N_{CCN,0} S^{\kappa}, \tag{109}$$

with N_{CCN} the number of activated CCN and $N_{CCN,0}$ and κ two constants. $N_{CCN,0}$ is usually defined as the background number of aerosols that can be activated, and κ varies between 0.3 to 0.7 depending on the case. The supersaturation is here defined as:

$$S = \frac{q_v - q_s}{q_s} \times 100, \tag{110}$$

where q_v is the vapor mixing ratio and q_s the saturation mixing ratio. Supersaturation is expressed in % and is explicitly limited to a value of 1% in the code (beware if larger values are expected). Note that S, the relative supersaturation, must not be confused with s, the absolute supersaturation (used in the diffusional growth theory above).

The number of droplets formed is explicitly assumed to be equal to the number of activated CCN: $N_c|_{nuc} = N_{CCN}$. All the droplets thus formed have the same mass which is equal to 4.19e15 kg (the mass of a droplet having the minimum diameter possible).

2.5.11 Ice nucleation

At the moment, only a simple ice nucleation scheme is present in MIMICA, as used in the ISDAC intercomparison for instance. Ice nucleation is defined as a simple relaxation of the ice number concentration towards a fixed background IN concentration:

$$\left.\frac{\partial N_i}{\partial t}\right|_{nuc} = \frac{N_i - IN_0}{\Delta t}. \tag{111}$$

To limit the nucleation events, this relation is only applied where liquid water is also present and where supersaturation over ice is sufficient, for example where $Si > 5\%$. All the newly nucleated ice crystals are assumed to have the same mass, defined as the minimum ice crystal mass possible. It results that:

$$\left.\frac{\partial Q_i}{\partial t}\right|_{nuc} = m_{min} \left.\frac{\partial N_i}{\partial t}\right|_{nuc}. \tag{112}$$

At the moment, m_{min} is hard coded but can be changed easily in *micro_ice.f90.*

On a long term basis, a detailed ice nucleation scheme should be implemented in MIMICA. The model will be based on nucleation theory and follow state-of-the-arts parameterization that can be commonly found in the litterature. In such works, all the different nucleation paths are usually included, namely nucleation via deposition, immersion, contact and condensation/freezing.

2.6 Boundary conditions

2.6.1 Lateral boundaries

By default, lateral boundary conditions are periodic which requires the definition of one or two (depending on the stencil used by the spatial discretization) ghost cells on both sides of the domain and in each direction. The values in the first or first two cells inside the domain at the inlet (respectively outlet) are copied into the one or two ghost cells present at the outlet (respectively inlet). This is done either by direct recopy (sequential job) or MPI communication (parallel job). The existence of ghost cells in both horizontal directions implies that all the 2D/3D arrays used in the code must be allocated accordingly. One has thus to be very careful when allocating new arrays or when exchanging data between different domains.

2.6.2 Top boundary

The standard treatment of the top boundary does not require any particular operation or model. In the last vertical layer of the domain, we will indeed consider that all the turbulent fluxes vanish (which is a reasonable assumption) so that we are only interested in the local sources and advection terms. Local sources do not present any difficulty as this is the case also for horizontal advection. Vertical advection by the w velocity can however lead to some problems, especially if we don't know if w is locally positive (directed upward) or negative (downward). In the first case, the flow goes out of the domain so that if the system of equation is well posed, it can be shown that the solution in the top layer is only determined by the initial condition (we first assume that there is no gravity wave). In other words, the flow goes out and no additional information is required to treat the boundary. On the opposite, if w is negative, it then appears that the flow comes inside the domain through the top boundary so that some values must be imposed at the "inlet" (otherwise the system would be ill-posed because information from outside the domain is required). In that case, the last vertical layer of the domain for k=nz is actually a virtual layer (as is the first layer at the surface) in which all the quantities are held constant and equal to their original values. Using these quantities, it is possible to evaluate gradients in the last physical layer in order to calculate the vertical advection.

When considering the top boundary, things are actually not as simple as that. A thorough analysis of the system of equations solved by the model can show that gravity waves can freely propagate inside the domain and will interact with inflow/outflow boundaries if we don't pay any particular attention. Indeed, classical boundaries act numerically as rigid layers on which gravity waves are able to reflect and thus propagate back into the domain. A specific treatment must then be applied in order to avoid such kind of spurious wave reflexion.

The simplest way to get rid of these gravity waves is to apply a sponge layer in the last 3-5 layers (controled by *zdamp* in *cm.nml*) of the domain in order to dampen any incoming wave. The damping function used in the model allows a fast relaxation of the local solution towards the initial reference state and is defined as:

$$\Psi_k = \Psi_k - \left(1 - \frac{z_{top} - z_k}{H}\right) \frac{\Psi_k - \Psi_k^0}{\tau_{sl}}. \tag{113}$$

The time scale τ_{sl} can be changed in *cm.nml* and should be at least of the order of 100 s. $H = z_{top} - z_{damp}$ is the height of the sponge layer whereas z_{top} is the altitude of the last scalar point in the domain. The factor depending on the altitutde allows a linear increase of the strength of the relaxation up to the top of the domain. The damping is applied to all the transported scalars and winds.

A more sophisticated approach consists in identifying exactly the gravity wave at the top layer and simply canceling the incoming component. In that case, it is common to define a simplified system of equation (non-hydrostatic, non-elastic, which remains reasonable at the top boundary) and to perform an analysis in the spectral space where waves are easily identified. A simple condition relating the vertical velocity and the pressure in the spectral space can finally be obtained that will allow upward waves to go out properly while damping downward gravity waves. Details of the derivation of so-called radiation boundary conditions can be found for example in Bougeault [1]. The condition on the pressure can be expressed as:

$$\hat{\pi} = \frac{N}{\sqrt{k^2 + l^2}} \hat{w}. \tag{114}$$

The previous relation was written for the perturbation Exner pressure π instead of the absolute pressure to remain consistent with the model formulation. k and l are the horizontal wave numbers in the k and l directions. The condition expressed above can be seen also as a dynamics adaptation of the top layer's height (in pressure coordinate) in response to incoming gravity waves (the top layer moves up and down with incoming waves).

The implementation of such a radiation boundary condition is simplified in MIMICA as a spectral solver is used to obtain a diagnostic pressure through the Poisson equation. Let's recall here that the diagnostic pressure is obtained by solving a tridiagonal system in the vertical direction while working in the spectral space in the horizontal plane. The solution of the tridiagonal system is then $\hat{p}$ so that condition 114 will appear only as a fixed boundary condition for the tridiagonal system at the top layer. The only remaining requirement is thus to obtain the transformed $\hat{w}$ in the horizontal spectral space, which can be done easily by calling the available FFT package (done only on the master processor, as for p). The radiation boundary condition can be selected through the keyword RADIA_BC in *start*.

2.6.3 Surface modeling

In LES simulations of the atmospheric boundary layer, the treatment of the bottom boundary is crucial. It must indeed provide a source of moisture and heat through evaporation processes and act as a rough surface to slow down the winds. Modeling of surface exchanges are mainly treated through the evaluation of the surface turbulent fluxes.

Symmetry is assumed in the first vertical point of the domain for velocities (slip condition):

$$\mathbf{u}\left(x, y, z = z_1\right) = \mathbf{u}\left(x, y, z = -z_1\right). \tag{115}$$

Note that the point at $z = -z_1$ is actually stored in the first layer of the grid corresponding to $k = 1$. After each momentum or scalar integration, the values in the second vertical mesh points are therefore copied into the first vertical cell. This means that the physical domain actually starts from the second vertical grid layer. Regarding scalars, symmetry is also imposed in the first vertical point:

$$\Psi\left(x, y, z = z_1\right) = \Psi\left(x, y, z = -z_1\right). \tag{116}$$

These two conditions are important to evaluate vertical gradients in the first layers of the domain.

The modeling of turbulent surface fluxes is controled through the parameter *isurf* present in *cm.nml*. *isurf* is defined as a double digit integer for which the first digit controls the method to obtain the surface friction quantities and the second digit, the surface model (fixed SST or fixed surface fluxes). *isurf* can take the following values 10, 11, 12, 20, 21 or 22.

Momentum and virtual potential temperature fluxes at the ground are evaluated using a friction velocity such as:

$$u_* u_* = \overline{u'w'} = \overline{v'w'}. \tag{117}$$

$$u_* \theta_{v*} = \overline{\theta_v' w'}. \tag{118}$$

The virtual potential temperature fluxes (idrectly related to buoyancy fluxes) are used to retrieve potential temperature and moisture surface fluxes. The two models available through *isurf* are given hereafter: u_* can either be fixed to a constant value when *isurf* equals 20, 21 or 22 (default is 0.25) or it can be computed dynamically using similarity theory for *isurf* $=$ 10 , 11 or 12. In the latter case, we must distinguish between fixed surface fluxes (isurf=11, 12) and fixed scalar surface values (isurf=10). When considering fixed surface fluxes, the sensible and latent heat fluxes must be provided explicitly by the user through parameters shf and lhf respectively in *cm.nml*. In a similar fashion, when fixed surface quantities are considered, the parameters sst and ssm must be given the appropriate values (when ssm = 0, skin surface moisture is retrieved assuming 100% saturation over liquid or ice at the skin surface temperature).

We write the turbulent flux $\overline{u'w'}$ following:

$$\overline{u'w'} = -K_m \frac{\partial u}{\partial z}, \tag{119}$$

with K_m approximated by:

$$K_m = \kappa z u_*, \tag{120}$$

where κ is the von Karman constant (≈ 0.35) and z is the altitude of the first velocity grid point. Combining these relations and integrating over z yields:

$$ln\left(\frac{z}{z_0}\right) = \frac{\kappa\, |u|}{u_*}, \tag{121}$$

from which u_* can be deduced. Here, z_0 is a surface roughness height depending on the ground nature (set to $1.e-4$ m by default). Relation 121 remains valid as long as the surface layer remains neutral. It doen't account for any TKE production through buoyancy.

For stable and unstable BL, Monin-Obukhov similarity theory can be used to derive a relation similar to 121 (after integration along z). In these cases we must define the surface buoyancy flux $\overline{\theta_v' w'}$ from the input surface parameters. The buoyancy flux will thus take different forms depending on which surface model is used: fixed SHF and LHF or fixed SST. In the case were surface heat fluxes are prescribed, we written:

$$b = \frac{g}{\theta_{00}} \overline{\theta_v' w'} = \frac{g}{\Theta_{00}} \left(\frac{SHF}{cp} + 0.608\theta_{00} \frac{LHF}{L_v} \right). \tag{122}$$

We then define the Monin-Obukhov length scale by:

$$L = -\frac{u_*^3}{\kappa b}. \tag{123}$$

Using these relations, the friction velocity can be deduced from a relation of the form:

$$ln\left(\frac{z}{z_0}\right) - \Psi_m(\zeta) = \frac{\kappa\, |u|}{u_*}. \tag{124}$$

This equations is similar to 121 derived for neutral surface layers, but the term Ψ_m appears now to account for surface buoyancy fluxes, source of stability. The ratio $\zeta = z/L$ can be viewed as a measure of ABL stability (stable for $\zeta > 0$ and unstable for $\zeta < 0$). Ψ_m is commonly deduced from observations and depends on the sign of ζ. We use here the forms proposed by Garratt [9]:

$$\text{for } \zeta > 0 \qquad \Psi_m(\zeta) = -5\zeta \tag{125}$$

$$\text{for } \zeta < 0 \qquad \Psi_m(\zeta) = 2ln\left(\frac{1+x}{2}\right) + ln\left(\frac{1+x^2}{2}\right) - 2tan^{-1}(x) + \frac{\pi}{2} \qquad \text{and} \qquad x = (1-16\zeta)^{1/4}. \tag{126}$$

As this was the case for the neutral boundary layer, u_* can be derived from relation 124. However, L depending directly on u_*, an iterative procedure is required to obtain a converged value of u_*, with a first guess given by the neutral formula. Five subiterations are used. The friction virtual potential temperature is finally deduced after the last subiteration from: $\theta_{v*} = -\frac{\theta_{00} b}{g u_*}$. This quantity is actually not used in the calculation of the surface fluxes as they are readily provided by SHF and LHF.

In the second case, when skin surface scalars are prescribed, the surface buoyancy flux is calculated from the gradient between the virtual potential temperature in the first LES grid cell above the surface and a skin surface virtual potential temperature:

$$b = \frac{g}{\theta_{00}} \left[(\theta_1 - \Pi_{00} SST) + 0.608 \theta_{00} \left(q_{t1} - SSM \right) \right] = \frac{g}{\theta_{00}} \left(\theta_{v1} - \theta_{v,skin} \right). \tag{127}$$

An initial friction potential temperature can readily be obtained under neutral conditions following:

$$ln \left(\frac{z}{z_0} \right) = \frac{k b \theta_{00}}{g Pr_t \theta_{v*}}, \tag{128}$$

with Pr_t the turbulent Prandtl number defined in *cm.nml.* We can now define as previously the Monin-Obukhov characteristic length scale, but using the surface buoyancy flux defined above:

$$L = \frac{u_*^2 \theta_{00}}{\kappa g \theta_{v*}}. \tag{129}$$

From these relations, the friction velocity and virtual potential temperature can be deduced from similar relations as previously:

$$ln \left(\frac{z}{z_0} \right) - \Psi_m \left(\zeta \right) = \frac{\kappa \left| u \right|}{u_*}, \tag{130}$$

$$Pr_t \left[ln \left(\frac{z}{z_0} \right) - \Psi_h \left(\zeta \right) \right] = \frac{\theta_{00} b}{g \theta_{v*}}, \tag{131}$$

with the ratio $\zeta = z/L$ defined as previously. Whereas Ψ_m doesn't change compared to the fixed surface fluxes case, Ψ_h is evaluated in a very similar way as a function of ζ and depending on its sign:

$$\text{for } \zeta > 0 \qquad \Psi_h \left(\zeta \right) = -7\zeta \tag{132}$$

$$\text{for } \zeta < 0 \qquad \Psi_h \left(\zeta \right) = 2ln \left(\frac{1+y}{2} \right) \qquad \text{and} \qquad y = \left(1 - 9\zeta \right)^{1/2}. \tag{133}$$

Once again, the implicit dependence on u_* and θ_* requires subiterations to yield an accurate solution. Now, both friction scalars are updated at the end of each substep. Once the final value of θ_{v*} is found, the surface fluxes of both temperature and moisture can be retrieved by simply assuming the same contribution of absolute temperature and moisture to the surface virtual potential temperature flux as for the skin surface virtual potential temperature. Therefore, SST and SSM are not used directly to compute the surface fluxes, but they are used in the Monin-Obukhov theory to determine the surface friction quantities that are in turn used in the surface fluxes expressions following 117 and 118.

In the MIMICA model the first layer above the ground is given by the second cell layer in the vertical dimension, for k=2. The layer k=1 actually corresponds to a virtual layer used to store surface quantities and therefore calculate gradients in the first layer.

2.7 Large scale tendencies (nudging)

In some cases, it can be requested that the simulated wind and scalar fields must be nudged to a given reference state. If we consider one particular quantity Φ, nudging is treated by simply adding one extra term N_Φ in the corresponding balance equation. This term allows the relaxation of the Φ-field towards the reference state Φ_0 with a given time scale τ_n. We then write:

$$N_\Phi \left(x, y, z; t \right) = -\frac{\Phi \left(x, y, z; t \right) - \Phi_0 \left(x, y, z; t \right)}{\tau_n}. \tag{134}$$

The relaxation time scale τ_n is usually case and variable dependent. It can be a constant, depend on the vertical coordinate or on the simulation time. The reference state Φ_0 is often taken to be one-dimensional only (dependent on the height only) and equal to the initial state.

In a similar fashion, large scale scalar advection can be considered. Large scale advection is simply modeled by a constant source added to the scalar tendencies (only for ice/liquid potential temperature and total water mixing ratio). These sources can possibly be time dependent (to be defined in *timevar.f90*).

3 Numerics

3.1 Time integration

In the current version of the model, four integration methods for the dynamics part are available: the 1st order Euler forward scheme (unstable for $CFL < 0.5$, but useful for defbugging purposes), a 2nd order Leap-Frog method with temporal filtering, a 3rd order TVD Runge-Kutta scheme and a 4th order low-storage, low-dispersion and low-dissipation Runge-Kutta scheme. The three latter methods are described below.

Considering a dummy variable Ψ whose values are known at times n and $n-1$, its value at time $n+1$ estimated through the Leap-Frog method will be given by (semi-discretized form):

$$\frac{\Psi^{n+1} - \Psi^{n-1}}{2\Delta t} = \mathcal{F}^n\left(\Psi\right), \tag{135}$$

where $\mathcal{F}^n$ is the local flux of Ψ evaluated at time n. The Leap-Frog method is second order accurate and has a stability criterion that can go up to about 0.8 in 3D depending on the choice of spatial discretization. Note that as Leap-Frog time integration requires the knowledge of all scalar fields at time $n-1$, the method cannot be used to advance the first time step. Initialization of the solution at $n = 1$ must hence be done by advancing the first time step thanks to the eplicit forward Euler method. As this is done only for the first time step, it does not significantly affect the solution's accuracy and stability.

Although quite interesting, the Leaf-Frog scheme possesses one important drawback. The fact that the integration procedure does not use the value of Ψ at time n may lead to a separation between the solutions at time n and $n+1$. Two solutions may thus exist at the same time level which can drift away from each other due to numerical errors. To avoid this, temporal smoothing of the solution following Asselin's filter is commonly used. This ensures that the solution remains consistent but it also dramatically decreases the order of accuracy by adding what can be interpreted as temporal diffusion. Asselin's filter can be written as:

$$\Phi^n = \Phi^n + \nu\left(\Phi^{n-1} - 2\Phi^n + \Phi^{n+1}\right) \tag{136}$$

The coefficient ν is fixed at 0.3 in MIMICA. Recently, an improvement of the method was proposed by Williams [24] to reach effectively 2nd order accuracy. The improved RAW filter is very simple: instead of relaxing only the solution at time n, the solution at time $n+1$ is also altered in such a way that the sum of the three successive available solutions $n-1$, n and $n+1$ is conserved during the damping process. If we note α the term added to Φ^n on the right hand side of relation 136, then the improved RAW filter reads:

$$\Phi^n = \Phi^n + k\alpha \tag{137}$$

$$\Phi^{n+1} = \Phi^{n+1} + (k-1)\,\alpha \tag{138}$$

Following [24], the optimal value of k is equal to 0.5. In the model, a value of 0.55 is fixed. Note that for $k = 1$, the classical Asselin method is retrieved. The filter is controled by the nsmooth parameter in *cm.nml*: for *nsmooth* $= 1$ the classical Asselin filter is used, for *nsmooth* $= 2$, the RAW filter is used.

The 3rd order TVD Runge-Kutta method implemented in MIMICA can be found in, for instance, Gottlieb and Shu [10]. A TVD method (for "Total Variation Diminishing", see scalar advection section below) has the property that the total variation of any transported quantity (that is the sum over all cells of the absolute difference between two successive cell values) does not increase from one time step to another. This condition ensures that the scheme is necessarily monotonicity preserving so that $\Phi^n_{i+1} \leq \Phi^n_i$. This guarantees that no spurious numerical oscillations will develop and the solution will remain free of purely numerical instabilities. Of course the numerical methods include both the time integration scheme and the advection schemes and both must must be TVD to guarantee the TVD condition for the overall integration step.

Following [10], the 3rd order TVD Runge-Kutta can be expressed under the condensed form:

$$\Psi^m = \alpha_{m1}\Psi^n + \alpha_{m2}\Psi^{m-1} + \beta_m \Delta t \mathcal{F}^{m-1}, \tag{139}$$

with m varying from 1 to 3. The initial conditions are given by: $\Psi^0 = \Psi^n$ and $\mathcal{F}^0 = \mathcal{F}\left(\Psi^n\right)$. The optimal values for coefficients α_{m1}, α_{m2} and β_m are given in table 3. At the end of the sub-steps, we obtain: $\Psi^{n+1} = \Psi^3$.

The so-called Runge-Kutta 44-2S method, as proposed by Ketcheson [13], belongs to the familly of low-storage Runge-Kutta schemes which improve the computer memory usage and efficiency compared to classical Runge-Kutta formulations, while conserving the same stability and accuracy properties. It is refered as "44" because it is 4th order

i	α_{i1}	α_{i2}	β_i
1	1	0	1
2	3/4	1/4	1/4
3	1/3	2/3	2/3

Table 3: *Coefficients of the TVD Runge-Kutta 3 scheme.*

accurate and it requires only 4 sub-steps to reach this condition. The scheme can be written in a condensed form as follows:

$$w^m = w^{m-1} + \delta_i \Psi^{m-1} \tag{140}$$

$$\Psi^m = \gamma_{m1}\Psi^{m-1} + \gamma_{m2} w^m + \beta_m \Delta t \mathcal{F}^{m-1}, \tag{141}$$

where m is the indice denoting Runge-Kutta sub-steps (here $m = 1, ..., 4$). The algorithm is initialized with $\Psi^0 = \Psi^n$, $\mathcal{F}^0 = \mathcal{F}(\Psi^n)$ and $w^0 = 0$. At the end of the sub-steps, we obtain: $\Psi^{n+1} = \Psi^4$. The values of the four Runge-Kutta coefficients are given in the table below.

i	δ_i	γ_{i1}	γ_{i2}	β_i
1	1.0000000	0.0000000	1.0000000	1.1937439
2	0.2176833	0.1210985	0.7217817	0.0992799
3	1.0658413	-3.8438337	2.1212093	1.1316780
4	0.0000000	0.5463789	0.1986530	0.3106657

Table 4: *Coefficients of the Runge-Kutta 4S scheme.*

Like every Runge-Kutta methods, the two present schemes require the evaluation of all scalar sources and advection fluxes three/four times within each time step. Compared to the Leap-Frog method, the CPU cost of the time integration is thus multiplied by a factor 3/4. However, whereas Leap-Frog, coupled with Asselin's filtering method, has an order of accuracy around 1, the present Runge-Kutta methods are respectively 3rd and 4th order accurate and do not need any filtering. Note that due to the necessity of performing successive subiterations within one Runge-Kutta step, time advancement of winds and scalars cannot be decoupled and must be performed simultaneously during each RK substep. Consequently, whereas winds and scalars are integrated in two separate routines for the Leap-Frog method, they are gathered in a single routine when either Runge-Kutta scheme is selected.

3.2 Numerical grid

The mesh used by the LES model is defined in a cartesian coordinate only, in 2 or 3 dimensions. The cell size is fixed constant in the horizontal plane but some refinement and stretching can be applied in the vertical direction. This operation is performed in *refine.f90* where user-defined functions can be entered to characterize the non-constant grid spacing. Some basic functions already exist and can be used at will.

The domain length and number of cells in all 3 directions can be set by the user in the *start* file. Note that extra cells were initially defined for each velocity component in their respective directions. Although they are no longer used in the model, they still exist and need to be considered.

Cartesian grids are very convenient to discretize fluid equations using finite differences. In the present code, a staggered grid system is employed, the so-called C-Arakawa grid, defined as follows:

- pressure, $p_{ijk} = p(x, y, z)$, and all the other scalars, are cell-centered
- the three velocity components are shifted by half a cell size downtream of p in their respective direction: $u_{ijk} = u\left(x + \frac{\Delta x}{2}, y, z\right)$, $v_{ijk} = v\left(x, y + \frac{\Delta y}{2}, z\right)$, $w_{ijk} = w\left(x, y, z + \frac{\Delta z}{2}\right)$

Using such a staggered grid system can largely increase the accuracy of the advection scheme without requiring the use of high order schemes [6]. In addition, the local velocity (respectively pressure) gradients will be defined exactly at the point where pressure (respectively velocity) is defined, using a simple first order approximation without interpolation. This improves the code's readability by making the evaluation of such gradients straightforward.

In practice, due to grid staggering, two different sets of vertical grid spacings have to be defined when vertical refinement is used: one corresponding to the spacing between two vertical velocity points, the other defined as the spacing between two scalar points. These two quantities will be refered to as Δz_w and Δz_p respectively.

3.3 Momentum advection

In LES models, and more generally in every small or large scale atmospheric models, the advection scheme is always considered as a key element that needs to be treated carefully. As explained in the next section, conservation of scalars and momentum during the advection step is enforced by the inversion of a Poisson equation for pressure that ensures that the flow remains divergence free everywhere. We start by describing momentum advection and then scalar advection which requires additional constraints.

The momentum advection scheme must provide sufficient accuracy and stability when coupled with the integration scheme. With no additional constraint, this leads to a very limited choice, mainly based on the easiness of implementation. Based on this, we chose to discretize momentum advection terms following central finite differences. These methods usually provide good accuracy given that they generate no numerical diffusion, and are in addition easy to implement. For clarity sake, the advective fluxes are recast into a conservative form:

$$u\frac{\partial u}{\partial y} \equiv \frac{F_{i,j+1/2} - F_{i,j-1/2}}{\Delta y}, \tag{142}$$

where $F_{i,j+1/2}$ and $F_{i,j-1/2}$ are the fluxes interpolated respectively at the center of the right and left cell faces on the C-Arakawa grid (in the y direction). For simplicity, we have here considered only a 2D system where the subscript i is used in the x direction and j in the y direction. This formulation is by definition energy and mass conserving and will therefore provide a better stability than any other formulation. The issue now consists in evaluating the interpolated fluxes. If we consider polynomial interpolations at the cell centers, the interpolated fluxes can be expressed as:

$$F_{i,j+1/2} = \sum_{l=0}^{n-1} C_{a,l} f\left(u_{i,j-a+l}\right), \tag{143}$$

where n represents the order of the interpolation, a the first point considered for the interpolation, f the local flux and $C_{a,l}$ the coefficients of the polynom. Selecting $a = 1$ and $n = 4$ yields a 4th order central approximation for the interpolated flux given by:

$$F_{i,j+1/2} = \frac{1}{2}\left(v_{i+1,j} + v_{i,j}\right)\left[-\frac{1}{12}u_{i,j-1} + \frac{7}{12}u_{i,j} + \frac{7}{12}u_{i,j+1} - \frac{1}{12}u_{i,j+2}\right]. \tag{144}$$

Considering the grid is staggered, an interpolation of the advective velocity v at the cell face is required. For the 2nd order central scheme, the above formulation becomes:

$$F_{i,j+1/2} = \frac{1}{2}\left(v_{i+1,j} + v_{i,j}\right)\left[\frac{1}{2}u_{i,j} + \frac{1}{2}u_{i,j+1}\right], \tag{145}$$

where we set $a = 0$ and $n = 2$. In practice, it is possible to switch from 2nd order to 4th order by setting the ADV4 parameter in *start* to TRUE.

The biggest advantage of central schemes is also their biggest drawback. Their lack of numerical dissipation may indeed lead to the amplification of small dispersive errors that are free to propagate within the field and cause numerical instabilities. It is therefore often necessary to filter these short wavelengths in order to damp the possibly growing instabilities. Following Durran [6], a 2nd order or 4th order filter operator is applied to horizontal advection of momentum to add artificial diffusion. We write:

$$v\frac{\partial u}{\partial y} \equiv \mathcal{F}^n_{i,j} + \gamma_n \widehat{u_{i,j}}^{nj}, \tag{146}$$

where $\mathcal{F}^n_{i,j}$ represents the centered nth order (n equals 2 or 4 here) semi-discrete advection operator, and $\widehat{\cdot}^{nj}$ denotes the nth order spatial filter applied along j defined by:

$$\widehat{u_{i,j}}^{4j} = u_{i,j+2} - 4u_{i,j+1} + 6u_{i,j} - 4u_{i,j-1} + u_{i,j-2}. \tag{147}$$

and:

$$\widehat{u_{i,j}}^{2j} = -u_{i,j+1} + 2u_{i,j} - u_{i,j-1}. \tag{148}$$

On the staggered grid system, the filtered terms are expressed at the same locations as the velocity vector components. In 146, γ_n represents the strength of the filter. If we fix $\gamma_4 = |u_{i,j}|/12\Delta x_j$ (resp. $\gamma_2 = |u_{i,j}|/2\Delta x_j$) along with the use of 4th order (resp. 2nd order) central differences, the resulting advection scheme (including artificial diffusion) is exactly equivalent to the 3rd order (resp. 1st order) upwind scheme. In practice, we multiply γ_4 or γ_2 as described previously by a constant C_{diff}, with $0 \leq C_{diff} \leq 1$ (in *cm.nml*) controling the strength of artificial diffusion.

3.4 Scalar advection

Regarding scalar advection, additional constraints must be considered in order to make sure that the advected quantities always remain within prescribed bounds and do not develop spurious numerical instabilities or reach unphysical values. For instance, it is obvious that mass mixing ratios (for vapor or hydrometeors) must always remain positive thus requiring a so-called positive-definite scheme. It must also be realized that low dissipation central difference schemes, as presented above for momentum advection, may develop numerical oscillations (wiggles) in regions where large gradients are encountered. This is particularly the case in the vicinity of an inversion layer. Consequently, specific advective schemes must be employed for scalars to avoid the development of instabilities. In the present model, a wide variety of schemes was implemented giving the user a large choice of methods, each having its particularities. All the schemes are at most 3rd order accurate which implies that they all generate little numerical dissipation.

As for momentum advection, we use a conservative flux formulation for scalar advection:

$$u_i \frac{\partial \Phi}{\partial x_i} \equiv \frac{F_{i+1/2} - F_{i-1/2}}{\Delta x_i}. \tag{149}$$

In the following are introduced the five different options available for scalar advection. The choice of the method is controled by the keywords ADV_LW, ADV_TVD, ADV_WENO and ADV_MUSCL in *start*. When either of these switches is set to TRUE,the corresponding scheme is selected. On the opposite, if they are all set to FALSE (default option), the central finite difference scheme with raw limitations is selected. When not using the ADV4 option for momentum advection (2nd order accuracy), it is recommended to use the 2nd order Lax-Wendroff method described hereafter.

3.4.1 Upwind finite differences

A first method implemented in the code relies on 3rd order upwind finite differences (it reduces to the 1st order upwind method when ADV4 is turned off in *start*) for which the flux balance inside each grid cell is limited to make sure that 1) no local extremum is created and 2) the transported scalar does not drop below a minimum value (typically 0). In order to account for the changing velocity sign inside the domain, we define two 3rd order scalar interpolations at the faces using two different stencils. The 3rd order interpolations are written:

$$\Phi^{+}_{i+1/2} = \frac{1}{6}\left(2\Phi_{i+1} + 5\Phi_i - \Phi_{i-1}\right), \tag{150}$$

when the wind velocity along i is positive, and:

$$\Phi^{-}_{i+1/2} = \frac{1}{6}\left(-\Phi_{i+2} + 5\Phi_{i+1} + 2\Phi_i\right), \tag{151}$$

when the velocity is negative. The upwind fluxes are finally expressed following:

$$F_{i+1/2} = \frac{1}{2}\left[\left(\Phi^{+}_{i+1/2} + \Phi^{-}_{i+1/2}\right) u_{i+1/2} - \left(\Phi^{+}_{i+1/2} - \Phi^{-}_{i+1/2}\right) \left|u_{i+1/2}\right|\right]. \tag{152}$$

This enables to select the proper scalar interpolation at the cell face depending on the velocity sign. This 3rd order scheme is used as such in the horizontal plane. In the vertical direction, the presence of an inversion layer sometimes requires more numerical dissipation than what is imposed by a 3rd order approximation. It is thus possible to switch to a 1st order upwind scheme for vertical advection only, which guarantees a high stability (the selection of the vertical advection method is hard coded at the moment and the default is the 1st order method). The finite difference formulation described here is very easy to implement and it doesn't contain any difficulty. However, when strong discontinuities are treated and the 1st order approximation is required in the vertical direction, the scheme generates excessive numerical dissipation which can deteriorate the solution's accuracy. In order to avoid this dramatic accuracy loss, more sophisticated methods have to be employed.

3.4.2 2nd order flux limited Lax-Wendroff method

Following the flux limiter approach, we seek to rewrite fluxes under the following form:

$$F_{i+1/2} = F^{LO}_{i+1/2} + \frac{1}{2}\phi_{i+1/2}\left(F^{HO}_{i+1/2} - F^{LO}_{i+1/2}\right), \tag{153}$$

with superscripts LO and HO denoting low-order and high-order fluxes respectively, and ϕ is the flux limiter function. Therefore, the fluxes are expressed as a combination of a high-order formulation and a monotone low order scheme (usually the upstream method) where the flux limiter ϕ acts as a switch selecting the monotone scheme in the region of high gradients and the high order approximation where the solution is smooth.

In the Lax-Wendroff method, following Durran, the upstream scheme is used as the monotone approximation and the 2nd order Lax-Wendroff method provides the basis for the high order part. The upstream method reads:

$$F_{i+1/2}^{up} = \frac{u_{i+1/2}}{2}\left(\Phi_{i+1} + \Phi_i\right) - \frac{\left|u_{i+1/2}\right|}{2}\left(\Phi_{i+1} - \Phi_i\right), \tag{154}$$

and the Lax-Wendroff flux can be written:

$$F_{i+1/2}^{lw} = \frac{u_{i+1/2}}{2}\left(\Phi_{i+1} + \Phi_i\right) - \frac{u_{i+1/2}u_{i+1/2}\Delta t}{2\Delta x}\left(\Phi_{i+1} - \Phi_i\right). \tag{155}$$

Replacing the two formulae into 153 yields:

$$F_{i+1/2} = \frac{u_{i+1/2}}{2}\left(\Phi_{i+1} + \Phi_i\right) - \frac{1}{2}\left[\left(1 - \phi_{i+1/2}\right)\left|u_{i+1/2}\right| + \phi_{i+1/2}\frac{u_{i+1/2}u_{i+1/2}\Delta t}{\Delta x}\right]\left(\Phi_{i+1} - \Phi_i\right). \tag{156}$$

The flux limited Lax-Wendroff method is at most 2nd order accurate in regions of smooth gradients.

Starting from the original definition of the flux limiter, ϕ must be defined as a function of the ratio r between two successive gradients:

$$r_{i+1/2} = \frac{\Phi_{i+1+a} - \Phi_{i+a}}{\Phi_{i+1} - \Phi_i}. \tag{157}$$

The indice a is introduced to account for the flow direction: $a = -sign\left(u_{i+1/2}\right)$. The flux limiter function is then defined so that ϕ can, under certain conditions, yield upwind biased fluxes which are efficient to smoothen the solution where this is necessary and avoid the development of numerical oscillations. The values reached by the flux limiter are bounded so that ϕ possesses certain properties (the method must be TVD or Total Variation Diminishing, which guarantees that no local extremum is generated). One popular approach is given by the MC limiter:

$$\phi_{i+1/2} = max\left(0, min\left(2r_{i+1/2}, 0.5\left(1 + r_{i+1/2}\right), 2\right)\right). \tag{158}$$

Replacing relation 158 into 156, one finds the exact expression for the flux limited Lax-Wendroff method. In the model, several flux limiter functions were coded. The choice of the method is however hard coded and this is the minmod limiter that is used by default (this is the most dissipative limiter possible, thus limiting the appearance of spurious oscillations).

In practice, evaluating the limiter ϕ in each grid cell of the domain appears to be quite CPU expensive, especially considering that a large part of the domain may actually not exhibit strong gradients. As a way to improve the method, Durran [6] proposes to use a criterion λ inspired by the WENO schemes (see below) to determine regions where flux limiting is necessary. By doing so, it is possible not only to improve the code's efficiency but also to improve the overall accuracy of the scheme as the damping provided by the limiter will only be applied where necessary. Durran defines a first set of positive quantities:

$$\gamma_l = \left(\Phi_{i+l} - \Phi_{i-1+l}\right)^2 + \left(\Phi_{i-1+l} - \Phi_{i-2+l}\right)^2, \tag{159}$$

with l varying from 0 to 2 in the present case. From there, the criterion λ is defined by:

$$\lambda_{i+1/2} = \frac{max\left(\gamma\right)}{min\left(\gamma\right) + \epsilon}. \tag{160}$$

The flux limiter will be calculated and applied only when λ exceeds a threshold value set to 10 (the value can be modified in *shared_dat.f90*).

3.4.3 3rd order TVD flux limited scheme

The 3rd order TVD method relies on the same principles as the Lax-Wendroff scheme introduced previously. The only difference is that the high-order flux $F_{i+1/2}^{HO}$ is now defined using the so-called $\kappa = 1/3$ scheme:

$$F_{i+1/2}^{HO} = u_{i+1/2}\left[\Phi_i + \frac{1-\kappa}{4}\left(\Phi_i - \Phi_{i-1}\right) + \frac{1+\kappa}{4}\left(\Phi_{i+1} - \Phi_i\right)\right]. \tag{161}$$

The coefficient κ varies between 0 and 1, but for a value of 1/3, a 3rd order upwind biased approximation is retrieved. The above relation is only true for a positive velocity. The flux associated to a negative velocity can be found by taking the exact symmetric of the above with respect to $i+1/2$. Using the upstream method as the low-order monotone scheme and recasting the 3rd order method into a flux-limited form, we can find the following simple interpolation of ϕ at $i+1/2$:

$$\Phi_{i+1/2} = \Phi_i + \frac{1}{2}\phi_{i+1/2}\left(\Phi_i - \Phi_{i-1}\right). \tag{162}$$

Let's note that if the flow velocity is negative, the equivalent relation will be written:

$$\Phi_{i+1/2} = \Phi_{i+1} - \frac{1}{2}\phi_{i+1/2}\left(\Phi_{i+2} - \Phi_{i+1}\right). \tag{163}$$

Following Hundsdorfer et al. [11], the flux limiter will include the κ coefficient as ϕ will be defined as a function of $K_{i+1/2}$:

$$K_{i+1/2} = \frac{1-\kappa}{2} + \frac{1+\kappa}{2} r_{i+1/2}. \tag{164}$$

The slope ratio r is still defined as the ratio between two successive gradients. Finally, K must be introduced into the flux limiter formulation to obtain ϕ for the $\kappa = 1/3$ scheme:

$$\phi_{i+1/2} = max\left(0, min\left(2r_{i+1/2}, 0.5\left(1 + K_{i+1/2}\right), 2\right)\right). \tag{165}$$

In order to improve the method, the same selective procedure as the one proposed for the Lax-Wendroff scheme can be applied. λ is defined exactly in the same way in both cases.

3.4.4 3rd order MUSCL scheme

The MUSCL scheme [15] is a central method so that it generates minor numerical dissipation which is a great advantage compared to the usual upwind biased flux limited methods. We start from relation 171 and express each flux in the difference as the average between a left and right flux interpolation at the cell face:

$$F_{i+1/2} = \frac{1}{2}\left(F_{i+1/2}\left(\Phi^R\right) + F_{i+1/2}\left(\Phi^L\right)\right). \tag{166}$$

The $*$ symbol denotes here the MUSCL reconstructed flux and R and L are respectively the right and left interpolated fluxes. These right and left fluxes can be expressed in various manners, and we choose here to use 3rd order polynomial reconstructions using a 3 point stencil extending from $i-1$ to $i+1$ for the left flux and i to $i+2$ for the right flux. F^R and F^L are respectively functions of Φ^R and Φ^L the interpolated states on the left and right sides of the face. Using the 3rd order ENO reconstructions we write:

$$\begin{aligned}
\Phi^L_{i+1/2} &= -\frac{1}{6}\Phi_{i-1} + \frac{5}{6}\Phi_i + \frac{1}{3}\Phi_{i+1} && (167)\\
\Phi^R_{i+1/2} &= \frac{1}{3}\Phi_i + \frac{5}{6}\Phi_{i+1} - \frac{1}{6}\Phi_{i+2}. && (168)
\end{aligned}$$

Further informations on ENO reconstructions and a table of the various coefficients are available for example in Durran [6]. The principle of the MUSCL method consists in rewriting these interpolated scalar quantities by making use of flux limiters. Rearranging 167 and 168 into a flux limited form, both expressions become:

$$\begin{aligned}
\Phi^L_{i+1/2} &= \Phi_i + \frac{1}{2}\phi_{i+1/2}\left[\frac{1}{3}\left(\Phi_i - \Phi_{i-1}\right) + \frac{2}{3}\left(\Phi_{i+1} - \Phi_i\right)\right] && (169)\\
\Phi^R_{i+1/2} &= \Phi_{i+1} - \frac{1}{2}\phi_{i+3/2}\left[\frac{1}{3}\left(\Phi_{i+2} - \Phi_{i+1}\right) + \frac{2}{3}\left(\Phi_{i+1} - \Phi_i\right)\right], && (170)
\end{aligned}$$

where ϕ is the flux limiter function defined previously. When $\phi = 1$, which is expected in smooth parts of the flow field, Φ^L and Φ^R reduce to the 3rd order ENO interpolations so that the total reconstructed flux F^* given by 166 yields a 4th order central method. On the opposite, when $\phi = 0$, reconstructed fluxes F^* are approximated by 2nd order central differences. Finally, using the reconstructed scalars at the faces, we compute the advective fluxes following:

$$F^*_{i+1/2} = u_i \Phi^*_{i+1/2}, \tag{171}$$

where no velocity interpolation at the face is required because of the staggered grid and with $\Phi^*_{i+1/2}$ calculated following relation 166.

Where stronger gradients are present, a more severe upwinding is necessary. However, due to the fact that the wind speed may change sign within the domain, upwinding must be done properly accounting for wind direction.

Using MUSCL reconstruction 166, Kurganov and Tadmor [15] proposed an upwind biased scheme based on the simple Lax-Friedrichs flux splitting which will provide sufficient numerical dissipation. We then write:

$$F^*_{i+1/2} = \frac{1}{2}\left(F^R_{i+1/2} + F^L_{i+1/2}\right) - a_{i+1/2}\left(\Phi^R_{i+1/2} - \Phi^L_{i+1/2}\right). \tag{172}$$

The velocity $a_{i+1/2}$ can be interpreted in the present case as a CFL based criterion and is defined by: $a_{i+1/2} = \max\left(|u_i|, |u_{i+1}|\right)$. This simplification is possible because the flow is non-divergent. This upwinding method is only applied in the vertical direction where the strongest gradients are expected. In the horizontal plane, the original central method is retained in order to limit numerical dissipation.

3.4.5 3rd order WENO scheme

Another high resolution method, which provides a little bit more dissipation than the MUSCL scheme, is based on WENO reconstructions. Following the approach described by Jiang and Shu [12], a 3rd order WENO method can be constructed by first defining 2nd order scalar interpolations at the cell faces on two successive stencils:

$$\Phi^1_{i+1/2} = -\frac{1}{3}\Phi_{i-1} + \frac{2}{3}\Phi_i \tag{173}$$

$$\Phi^2_{i+1/2} = \frac{1}{2}\Phi_i + \frac{1}{2}\Phi_{i+1}. \tag{174}$$

The scalar advective flux at the face is then evaluated using a weighted combination of these two interpolations:

$$F_{i+1/2} = \left(\omega_1\Phi^1_{i+1/2} + \omega_2\Phi^2_{i+1/2}\right)u_i. \tag{175}$$

The weights ω_1 and ω_2 are then computed as follows:

$$\omega_k = \frac{\alpha_k}{\alpha_1 + \alpha_2}, \qquad \alpha_k = \frac{C_k}{\left(\beta_k + \epsilon\right)^2}, \tag{176}$$

with k equals to 1 and 2, ϵ a small quantity (typically equals to 10^{-6}) to avoid division by 0, $C_1 = 1/3$ and $C_2 = 2/3$ the optimal weights and β_k smoothness functions that indicate whether steep gradients are located in the vicinity of the cell. The smoothness functions are given by:

$$\beta_1 = \left(\Phi_i - \Phi_{i-1}\right)^2 \qquad \text{and} \qquad \beta_2 = \left(\Phi_{i+1} - \Phi_i\right)^2. \tag{177}$$

The above describes the simplest form of the 3rd order WENO method, for a flow possessing a positive velocity.

In a similar fashion as for the MUSCL method, we must account for the fact that wind may change direction within the domain thus requiring a different upwinding. If the flow velocity is negative, we must then evaluate the fluxes as the exact symmetry of those described above with respect to the point $x_{i+1/2}$. As a result, we use:

$$\Phi^1_{i+1/2} = \frac{1}{2}\Phi_i + \frac{1}{2}\Phi_{i+1} \tag{178}$$

$$\Phi^2_{i+1/2} = \frac{2}{3}\Phi_{i+1} - \frac{1}{3}\Phi_{i+2}, \tag{179}$$

that are weighted in a similar manner as previously. At each cell face, we can hence define a left and right WENO reconstruction (recalling the MUSCL method) that can be combined through the Lax-Friedrichs flux splitting 172 to yield the correct upwinding.

The WENO schemes are known to be rather dissipative also in regions where only weak gradients are present. Several methods were proposed in order to correct this behaviour and select more properly the regions where the weighting procedure should be applied (this can be done f.i. by changing the formulation of the β_k coefficients). In the MIMICA model, we chose to use a smoothness indicator λ that will allow to identify the disconinuities in the scalar field. The WENO procedure is then used only in regions where λ is greated than a given threshold λ_{max}, whereas the β_k coefficients are set to 0 elsewhere. The strong indicator from Xu and Shu [25] was selected here. If we define the coefficients $\beta_i = \left(\Phi_i - \Phi_{i-1}\right)^2 + \epsilon$, we can the construct the indicator as follows:

$$\lambda_i = \frac{\alpha_i}{\alpha_i + \gamma_i}, \tag{180}$$

with:

$$\alpha_i = \left(\frac{\beta_i}{\beta_{i-1}} + \frac{\beta i + 1}{\beta i + 2} \right)^2 \qquad \text{and} \qquad \gamma_i = \frac{\left(\Phi_{max} - \Phi_{min}\right)^2}{\beta_i}. \tag{181}$$

In the above, ϵ is a small constant to avoid divisions by 0 (we take $\epsilon = 10^{-6}$). Following this definition, the indicator is a normalized quantitity that will take a value close to 0 in smooth region and tends to unity close to strong gradients when $\alpha_i >> \gamma_i$. The threshold value λ_{max} is set by default to 0.005. The main advantage with using this definition is that λ is defined as varying between 0 and 1 so that the threshold value can be set independently on the considered scalar (several orders of magnitude separate the largest and smallest transported scalars so that it can be hard to properly identify discontinuities with a single indicator and threshold).

3.5 Multidimensional advection

Untill that point, we have only considered one dimensional scalar advection. In multiple dimensions, the direct application of flux limited methods can cause some implementation difficulties so that an explicit direction splitting method was employed in this code. Scalars are hence transported successively along each direction before the total fluxes are reconstructed. In three dimensions, four successive steps are thus required:

- $$\Phi^1_{i,j,k} = \Phi^n_{i,j,k} + \Delta t \frac{F_{i+1/2,j,k}\left(\Phi^n_{i,j,k}\right) - F_{i-1/2,j,k}\left(\Phi^n_{i,j,k}\right)}{\Delta x_i} \tag{182}$$
- $$\Phi^2_{i,j,k} = \Phi^1_{i,j,k} + \Delta t \frac{F_{i,j+1/2,k}\left(\Phi^n_{i,j,k}\right) - F_{i,j-1/2,k}\left(\Phi^n_{i,j,k}\right)}{\Delta x_j} \tag{183}$$
- $$\Phi^3_{i,j,k} = \Phi^2_{i,j,k} + \Delta t \frac{F_{i,j,k+1/2}\left(\Phi^n_{i,j,k}\right) - F_{i,j,k-1/2}\left(\Phi^n_{i,j,k}\right)}{\Delta x_k} \tag{184}$$
- $$\mathcal{F}^n_{i,j,k} = \frac{\Phi^3_{i,j,k} - \Phi^n_{i,j,k}}{\Delta t}, \tag{185}$$

with $\mathcal{F}^n_{i,j,k}$ representing the total advective flux in a given grid cell for scalar Φ.

3.6 Precipitation

Considering any hydrometeor moment (N or Q) denoted Ψ, the related precipitation flux reads:

$$P^\Psi \equiv -\frac{1}{\rho} \frac{\partial \rho \left\langle V_p \right\rangle \Psi}{\partial z}, \tag{186}$$

with $\langle V_p \rangle$ the mass weighted averaged terminal fall speed of the chosen hydrometeor kind. This velocity is expressed following relation 39 and exhibits a non-linear dependence on the two hydrometeor moments. The modeled precipitation flux P^Ψ is thus a non-linear derivative term which can generate inherent numerical instabilities in the vicinity of discontinuities or sharp gradients (this is a property of non-linear PDEs). In order to avoid the development of such instabilities, precipitation fluxes are discretized using upstream 1st order differences which introduce large numerical dissipation damping any numerical wave. We write (to simplify, only the vertical index k is shown):

$$P^\Psi_k = \frac{1}{\rho_k} \frac{\rho_{k+1} \left\langle V_p \right\rangle_{k+1} \Psi_{k+1} - \rho_k \left\langle V_p \right\rangle_k \Psi_k}{\Delta z_p}. \tag{187}$$

Note that $\langle V_p \rangle$ being an hydrometeor property, it is calculated at the scalar location and is thus cell-centered.

3.7 The pressure solver

3.7.1 Diagnostic pressure

Compared to the original CRM model, the LES code solves for the purely incompressible equations. Two main reasons motivated this choice. First, by getting rid of acoustic perturbations inside the domain, the stability criterion is fixed by the advective velocity instead of the acoustic one. This latter being of the order of 300 m/s in the atmosphere, the stability criterion is much more constraining in a compressible flow. Very short time steps or time-splitting procedures (not really efficient and accurate) are therefore required in order to fulfill the stability requirements. The second reason is related to the advection discretization described previously. In the CRM version of the code, a pseudo-anelastic form of the equations are solved where a divergence free advection term is discretized and a pressure equation is solved jointly (the system is over-constrained). This type of method will be conservative only if the flow is actually divergence free, condition that is not properly respected if a pressure equation is solved. Forcing the flow to remain divergence free by the method presented below ensures that scalars and momentum are conserved when using the chosen advection schemes.

When the anelastic approximation is made, the velocity divergence vanishes and the continuity equation is no longer required. As a result, the perturbation Exner pressure has to be calculated differently than by solving a specific time-dependent balance equation. Combining the momentum equation 14 and the anelastic constraint 22 yields an elliptic equation for the perturbation pressure that can easily be solved during each time step.

Let's take the divergence of the momentum equation 14. Given that the incompressible flow is divergence free, the divergence of the momentum tendency vanishes and we obtain:

$$\frac{\partial}{\partial x_i}\left[-u_j\frac{\partial u_i}{\partial x_j} - Cp\theta_{00}\frac{\partial \pi}{\partial x_i} + b\delta_{i,3} + f_k\left(u_j - u_{gj}\right)\epsilon_{i,j,k} + \frac{1}{\rho}\frac{\partial \tau^u}{\partial x_j}\right] = 0. \tag{188}$$

An equation for the perturbation pressure is then easily derived:

$$\frac{\partial^2 \pi}{\partial x_i^2} = \frac{1}{Cp\theta_{00}}\frac{\partial}{\partial x_i}\left[-u_j\frac{\partial u_i}{\partial x_j} + b\delta_{i,3} + f_k\left(u_j - u_{gj}\right)\epsilon_{i,j,k} + \frac{1}{\rho}\frac{\partial \tau^u}{\partial x_j}\right]. \tag{189}$$

To solve this elliptic equation, the pressure solver uses a mixed method with FFTs in the horizontal plane and the inversion of a tridiagonal system in the vertical direection. Assuming that the lateral boundaries are periodic (which is a prerequisite for FFT in the horizontal plane), it is possible to transform equation 189 into the Fourier space where second order derivatives become simple multiplications:

$$k^2\hat{\pi} + l^2\hat{\pi} + \frac{\partial^2 \hat{\pi}}{\partial z^2} = \hat{\mathcal{F}}, \tag{190}$$

where $\hat{.}$ denotes the Fourier transform of a given variable and k and l are the wave numbers in the x and y directions respectively. Approaching the second order derivatives along z by central differencies yields a tridiagonal system which can be efficiently resolved using standard methods. Transforming the solution $\hat{\pi}$ back into the physical space gives a very accurate estimation of the perturbation pressure.

Various methods can be employed to solve the coupled pressure/velocity system within one time-step: a simple explicit method was chosen in MIMICA. The pressure term appearing in the momentum equation is not added during the first integration step. Instead, we calculate an intermediate value of the velocity vector at time $n+1$, denoted $\mathbf{u}^*$, with $\frac{\partial u_i^*}{\partial x_i} \neq 0$. The momentum tendency divergence is computed during this step. At the end of the time step, equation 189 is solved to yield a diagnostic pressure which ensures a divergence free flow. This new pressure at time $n+1$ is then used to correct the intermediate velocity field $\mathbf{u}^*$ and find the final velocity field at $n+1$. The complete procedure is summarized below, where Euler forward is used for time integration:

$$\frac{u_i^* - u_i^n}{\Delta t} = \mathcal{F}_i^n \qquad \text{with } \frac{\partial u_i^*}{\partial x_i} \neq 0 \tag{191}$$

$$\frac{\partial^2 \pi^{n+1}}{\partial x_i^2} = \frac{1}{Cp\theta_{00}}\frac{\partial \mathcal{F}_i^n}{\partial x_i} \tag{192}$$

$$\frac{u_i^{n+1} - u_i^*}{\Delta t} = -Cp\theta_{00}\frac{\partial \pi^{n+1}}{\partial x_i} \tag{193}$$

3.7.2 Prognostic pressure

The MIMICA model must be used both as a CRM and as a LES model. As a result, it was decided that although the diagnostic pressure method is far more efficient, a prognostic pressure option should be kept in order to keep the main CRM features intact.

When the prognostic pressure option is turned on (PRESS_DIAG is FALSE in *start*), an equation for the dynamic pressure perturbation is solved along with the other governing equations. Following Klemp and Wilhelmsson, an equation for the dynamic pressure can easily be derived and after further simplifications, the following equation is obtained:

$$\frac{\partial \pi}{\partial t} = -\frac{c^2}{\rho_0 Cp\theta_{v0}^2}\frac{\partial \rho_0\theta_{v0}u_i}{\partial x_i}, \tag{194}$$

where the subscript 0 denotes the initial reference state and $c^2 = \sqrt{\gamma r T}$ is the sound speed in the atmosphere (γ is the ratio between the constant pressure and constant volume heat capacities and is set to 1.4 in the atmosphere).

Although written in a quite simple form, the above pressure equation introduces numerical difficulties that require the introduction of temporal substeps. The pressure equation coupled to the momentum equations allow acoustic waves to propagate within the domain. As a result, numerical stability can only be achieved if these acoustic waves are properly temporally resolved (see next section). Considering the acoustic CFL criterion must be verified to ensure stability, and knowing that the speed of sound in the atmosphere is of the order of 300 m/s, it comes that Δt must be reduced by a factor close to 300 to maintain stability compared to a similar simulation with diagnostic pressure.

The prognostic pressure formulation is therefore quite time consuming, unless specific methods are used which reduce considerably the total CPU time for each step. We will consider that all the other physical processes evolve with characteristic time scales much larger that the acoustic waves and are sufficiently well resolved by the large convective time steps. The use of the small acoustic time steps can therefore be limited to the resolution of the coupled velocity/pressure system. Following Klemp and Wilhelmsson we introduce a time splitting method where a small enough time step is used to solve the above pressure equation as well as the momentum equations whereas all the other physical processes, including the transport of all the other scalars, are solved using the original convective time step. Subiterations are then performed for which only the advective terms in the pressure and momentum equations as well as the pressure term in the momentum equations are updated each time.

The validity of both methods in a given case is usually hard to assess. The diagnostic pressure method introduces a large dependence of the results on the initial field as the Poisson solver needs to assume an initial pressure solution to solve the problem. This may lead to results converging towards a different stationnary state than expected. However, in the prognostic pressure method case, the pressure equation presented above is in fact simplified and normally contains several extra terms. These simplifications also introduce some kind of dependency of the results on initial assumptions, so that the computed converged field may also be erroneous. Both methods can thus be viewed as equivalent in terms of accuracy but not in terms of CPU efficiency: the subiterations introduced in the prognostic pressure case dramatically reduce the model's performances.

3.8 Criterion for numerical stability

Time integration performed in the MIMICA model uses only explicit schemes and are therefore conditionally stable. This means that to ensure numerical stability throughout the computation and avoid the development of spurious oscillations, the time-step used for temporal integration must remain within prescribed bounds.

Without giving complicated details, spectral analysis of the time integration scheme leads to the elaboration of the so-called CFL criterion which is usually expressed under the form:

$$\left(\frac{|U|}{\Delta L} + N \right) \Delta t < CFL_{max}. \tag{195}$$

In the above condition, gravity waves were considered so that N, the Brunt-Vaisala frequency, appears as an additional term in the CFL number. The velocity $|U|$ and ΔL are obviously chosen in one-dimensional cases, but when multidimensional simulations are performed, several cases can be considered and the proper definition of U is no longer trivial. Using a staggered grid system actually simplifies the problem and enables to somewhat reduce the limitation imposed by the CFL number. As a matter of fact, in such a case, it can be shown that the velocity used for the criterion reduces to (here in 3D):

$$\frac{|U|}{\Delta L} = \max \left(\frac{|u|}{\Delta x}, \frac{|v|}{\Delta y}, \frac{|w|}{\Delta z} \right) \tag{196}$$

In 195, CFL_{max} is a threshold over which the numerical scheme starts to get unstable. Its value depends both on the integration scheme as well as on the choice of method made for wind and scalar advection.

In MIMICA, the recommended typical values for CFL_{max} depending on the integration scheme are: for Leap-Frog time differencing, we usually take $CFL_{max} \approx 0.7$, for the Runge-Kutta 3 method, $CFL_{max} \approx 1.33$ and for the Runge-Kutta 44, $CFL_{max} > 2$. The values indicated here were determined experimentally using at least 4th order central schemes for advection. They may thus differ significantly from theoretical criteria often found in the litterature and corresponding to ideal cases.

In practice, CFL_{max} can be prescribed by the user in *cm.nml*. We recommend to choose values of the threshold slightly lower than the one suggested above. The CFL criterion is used to determine the maximum time-step allowed during the simulation. If CFL_{max} is exceeded at at least one particular location inside the domain, the time-step is automatically decreased to verify relation 195 (done in *check.f90*).

When pressure is solved in a prognostic manner, that is when a pseudo-anelastic set of equations is solved, sound waves are allowed to propagate inside the numerical domain and this is reflected through the CFL criterion. A spectral analysis can show that the stability criterion for the numerical scheme reduces in that case to:

$$\frac{|U + c|\,\Delta t}{\Delta x} < CFL_{max}, \tag{197}$$

where c is the sound celerity. Given the fact that c is usually very large compared to U (of the order of 300 m/s), and assuming that the N term arising from gravitational waves can be neglected, it appears that the CFL criterion in presence of acoustic waves is much more contraining than in a fully compressible case.

When considering the CFL number as a way to assess numerical stability of a scheme, one has to keep in mind that this kind of criterion has been originally derived from one-dimensional problems in ideal cases. Stability may thus be much more sensitive to other parameters than just the CFL number (for example the geometry or the advection schemes). As an example, Durran [6] shows that although combining a 4th order central finite-difference advection scheme with Leapfrog time differencing should yield a stability criterion of 2, it is actually observed in practice that the CFL should not exceed 0.73. As a result, the CFL method described above to limit the time step provides only a very simplistic solution which may not be true for the values of CFL_{max} recommended.

3.9 MPI communications and domain decomposition

The LES model is parallelized using the Message Passing Interface MPI. This requires the decomposition of the numerical domain in a given number of elements within which the governing equations will be solved by a single CPU. Parallelization requires the addition of ghost cells on the sides of each local domain. The informations contained in these cells must then be communicated between neighbouring domains in order to allow a proper calculation of gradients, even at the internal boundaries. At the moment, domain decomposition can be performed along the two horizontal directions using the *DECOMP_2D* switch in *start* (if one wants 2D decomposition, otherwise, default is decomposition along x only). This represents a major improvement compared to the original CRM model as 2D domain decomposition allows the use of more CPUs at the same time thus increasing the efficiency of the code.

When performing a parallel run, one must be aware of some limitations which have to be kept in mind:

- the domains created after decomposition all contain the exact same amount of internal points. The total number of cells prescribed in *start* must then be a multiple of the number of processors/domains;
- as suggested at the end of the manual for output files, some of the MPI communications done are unnecessary and lead to an excessive increase of the CPU cost: f.i. before each output operation, all the data are gathered on a single processor before being written;
- the use of a spectral method in the horizontal plane to solve the Poisson equation (diagnostic pressure) has to be performed on a single domain/processor as the Fourier transforms are hard to parallelize; this requires that the momentum tendencies have first to be gathered on a single processor and that the resulting diagnostic pressure has to be broadcast to all the processor after the calculation;
- more generally, the implementation of MPI communications may not be properly optimized.

A number of communication modules and interfaces have been created to simplify the use of MPI communications. Thereby, instead of calling directly the MPI routines, it is often more appropirate to call the corresponding internal interface: *collect* allows to gather the data located on all the CPUs on the master processor, *distribute* performs the opposite operation by sending informations located on one processor to all the CPUs, *exchange* communicates the data contained in the ghost cells of two neighbouring domains (treatment of internal and lateral periodic boundaries) and *distexch* combines *distribute* and *exchange.*

4 Input/Output

4.1 *cm.nml* and *out.nml*

The namelist file *cm.nml* (situated in the working directory) contains some model parameters and configurations that can possibly be changed from one case to another or before a restart. (some additional important parameters that are presently hard-coded should be included in *cm.nml*). The file starts with the *$cm_in* line and ends with *$END*. All the lines below are considered as comments. All the parameters appearing in the file are defined and stored during the run in the *shared_data* module, but they also need to be defined in the header section of the main *mimica.f90* routine as belonging to the *cm_in* namelist. The reading of this file is treated automatically as a namelist where the values indicated are affected to the corresponding variable name. The order of the entries in the namelist doesn't matter.

Table 6 lists all the parameters appearing in *cm.nml* and their role in the model. The default values assigned to all these parameters are defined in *INCLUDEdefault.h*. For binary codes, the value 1 activates the process.

out.nml is a second namelist file present in INCLUDE that is used to prescribe which variables must be written in the output files (see the Output subsection). The correspoding *ou_in* namelist is defined in the header section of *mimica.f90* and must be used exactly as the cm namelist.

4.2 *start*

The *start* file contains switches (environement variables) that activate or deactivate certain modules, models or code features. It controls also the type of model executable created during compilation (regular, debug, parallel...). The file is closely related to *wheader.csh* which reads *start*, interpretes the different environement variables and enables or disbales the compilation of the corresponding parts of the code. These parts can be easily identified within the model between the *#ifdef* and *#endif* statements. A list of all the environement variables with their current status is generated after each compilation and written in *INCLUDE/ctrparam.h*. Note that the switches control the compilation of these blocks, not their execution during the run: if a bug has been introduced in one of these blocks but the switch is not turned on, the compiler won't produce any error.

Table 7 summarizes all the existing environement variables and which features they control. Regarding the choice of numerical method, only one environment variable among all the ADV_XXXX options as well as among LF, RK3 and RK4 must be set to TRUE at the same time. When They are all FALSE, the default time integration scheme is the forward Euler method and scalar advection is performed using upwind biased finite differences. For the other parameters, a switch set to TRUE enables the corresponding option.

4.3 Output files

4.3.1 Overview

Outputs are controled by the *output.f90* routine. Four main types of output files are written: global outputs which contain the entire 2D/3D field for each selected variable (see below), 1D vertical profiles where every selected quantities are averaged horizontaly, time series of these same vertical profiles and time series of scalars defined in *calmean.f90* and usually corresponding to domain extrema or domain ensemble averages. The frequence for these outputs is controled by *iax*, *its* and *iprof* respectively. *its* controls at the same time the output of the scalar time series and 1D sounding time series.

The total field files are simply given the same name as the output variable considered (not considering the possible extension notifying the output time). The 1D soundings (instantaneous or time series) possess the name of the variable with the .pro extension. The scalar time series are all written in a single file named T_S. All these files are stored in the OUTPUT folder of the local directory.

The selection of the output variables is done through the namelist *ou_in* contained in *out.nml*. The namelist is defined in the exact same manner as *cm_in* discussed in the previous section. *ou_in* contains a list of logical variables out_xxx explicitly defining which quantities are to be written in the outputs. To select one variable, its corresponding logical switch has to be set to TRUE in *out.nml*. Two files can be created for each selected quantity: one containing the whole fields and one for the vertical profiles (if the variable PROFILE is turned on). The names and unit numbers associated to each output variable can be found in *INCLUDE/assign_out.h*. Note that one file is created for each 2D/3D quantity, even for multiple processor jobs: this implies that slow MPI communications are requested for each output variable and during each output step which negatively affects the model performances.

Variable name	Default value	Meaning
dt	1 s	Main time step for dynamics/microphysics
ds	1 s	Small time step for prognostic pressure
cfl_max	0.75	Maximum allowed CFL (smaller than max. expected for stability)
cfl_min	0.5	Minimum allowed CFL (time step is increased if CFL < cfl_min)
ldtfix	0	If 1, the time step *dt* is kept constant, CFL depdendent otherwise
nstart	1	Loop index of the first time step
nstop	1	Final simulation time (default = sec.)
nunit	1	Unit for nstop (1 = sec, 60 = min...)
iax	10	Output frequency in sec. for entire fields (if <0, no output)
iav	10	Output frequency in sec. for the restart file
its	10	Output frequency in sec. for time series
ipro	10	Output frequency in sec. for profiles
ires	10	Output frequency in sec. for restart files
iradx	-5	Frequency for radiation solver (if <0, no call)
i_creact	1	Frequency for chemistry calculation (if <0, no call)
isurf	0	Surface fluxes modelling (see Boundary conditions)
dx	2000 m	Cell size in x
dy	2000 m	Cell size in y
dz	500 m	Reference cell size in z
z1; z2	500; 900 m	Refinement bounds in z
sratio	1	Refinement ratio in z
zdamp	1000.	Altitude where sponge layer is applied
tdamp	300.	Time scale for damping in sponge layer
u0shift	0	Translation velocity in x
v0shift	0	Translation velocity in y
pran	0.33	Turbulent Prandtl number (if <0, no SGS for scalars above 100m)
cdiff	0	Strength of artificial diffusion for momentum advection
lmicro	0	Microphysics level: 0=no microphysics, 1=warm micro only, 2=pristine ice micro, 3=ice
lndrop	1	Turns on prognostic cloud droplet number concentration ($N_c = cst$ otherwis
drizz	1	Activation of rain conversion
c_sed	0	Activation of cloud droplet sedimentation
lvent	0	Activation of ventilation effects for codensation/deposition
lgrowth	0	Direct integration of diffusional growth (0=explicit, 1,2=pseudo-analytic)
lfreeze	0	Turns on production of ice
drmin	70	Separation diameter between cloud droplets and rain drops (in micro m)
shf	20	Sensible heat flux at the surface
lhf	80	Latent heat flux at the surface
sst	295	Skin surface temperature
ssm	0.	Surface moisture (if 0, ssm calculated from sst at 0 supersaturation)
zrough	1.e-3	Surface roughness height
alb	0.6	Surface albedo for radiation solver
Ddiv	3.75e-6	Large scale horizontal divergence forcing
nsmooth	1	Temporal smoothing of the solution (0 by default in not Leap-Frog)
ksmooth	0.2	Smoothing parameter in Asselin's filter
xnc0_c	50.e6	N_{CCN0} parameter in CCN activation power law
xnc0_k	0.7	κ parameter in CCN activation power law
xn_ccn0	50.e6	Initial/reference CCN value
xn_in0	100.e3	Initial/reference IN value

Table 5: *Model parameters defined in cm.nml.*

4.3.2 Output options

Both 2D/3D fields and 1D profiles can be written either in formatted ASCII files or unformatted binary files. The choice is made via the FORM switch present in *start* (the output will be written in ASCII if FORM is set to TRUE).

Variable name	Default value	Meaning
dpt	1.e-4	Maximum random perturbation amplitude on Pt
dqv	1.e-5	Maximum random perturbation amplitude on qv
if_filter	60	Frequency for filtering after integration
nbulb	1	Number of initial warm bubbles
mx; my; mz	375; 0; 6	Initial poisition of the warm bubble
mx1; my1	3; 1	Initial extent of the warm bubble
mx2; my2	3; 1	Initial extent of the warm bubble
ioe; joe	2; 1	Bubble size parameters ?
mz1	1	Vertical position of the bubble & first layer for random perturbations
mz2	50	Vertical position of the bubble & last layer for random perturbations
koe	1	Bubble size parameter ?
j_day	67	Day of the year
ctr_lat	-4	Latitude of the case (if <0, not relevant)
t_utc	0	local time as compared to UTC
casename	xxxx	Case name
file_xxxx	-	File names and locations

Table 6: *Model parameters defined in cm.nml (continued).*

Binary files are less memory demanding and thus recommended but it is often convenient to be able to directly read the output files to track for instance the location of a singular point within the flow field. In addition, for these two types of outputs, it is possible to write either the instantaneous results or time averaged data depending on the status of MEANDATA in *start*. If time averaged data are written, the averages are calculated over a time lapse set in *cm.nml* by the parameter *iav*.

In the *start* file, the parameters OVERWRITE and OUTPUALL control the behavior of the model when several output times are met during the simulation. Three possibilities can be chosen depending on the combinations between these two switches. When OUTPUTALL is set to TRUE, each of the different selected output variables (in *out.nml*) is written in one separate file for each output event. The files contain the extension .xxxx, where xxxx is the physical time in seconds at which the file has been created. This regards both the total 2D-3D fields and the 1D soundings. For the profile time series, only one single file for each output variable is created, independently on the OUTPUTALL switch. This file possesses the extension .0000 to be easily identified. The OUTPUTALL option has the disadvantage to create a large amount of output files but it allows a simple post-treatment of these files.

When OUTPUTALL is turned to FALSE, only one file is created for each variable at the beginning of the run. Depending on the value of OVERWRITE, these single files can be either overwritten during each output (TRUE) or the new output can simply be written at the end of the file (if FALSE) so that all the different outputs are still present within one file.

4.3.3 Post-processing

A small post-processing tool has been developed in fortran: *post_process.f90* to help processing the raw data files generated by MIMICA for a given number of selected quantities (defined in *file_list.dat*). The program merges all the different outputs and generates a single file, in a prescribed file format. When outputs were performed at different simulation times, the tool can only handle the situation if one file have been created after each output (OUTPUTALL = TRUE). At the moment, the software can't read block output files written with the OVERWRITE switch turned to FALSE.

The post_process tool requires an input list *file_list.inp* to select the files to be processed. This list is made of two parts: the first one, introduced by the keyword time, contains a list of times that we want to process. The figures written below must exactly correspond to the time extensions of the raw output files. If required, the profile time series can also be processed. In that case, the time list must also contain a line 0000 which is the extension for the profile time series files. The second part of the *file_list.inp* file is introduced by the keyword name and contains a list of variable names to process. It is possible to comment one name line by adding # in the first column. Using the keywords, it is possible to select just a limited subset of results to process, at specific output times.

The post-processing software will create one single file for each output kind and for each selected time. The full fields will be written in *les_outptu.tec* files, to which is added an extension for the output time considered. The 1D

Variable name	Role
NP	Number of processors/domains
CMPLER INTEL/PGI	Choice of intel or portland group compiler (default: gfortran)
R8	Real type size (8 bits)
DEBUG	Builds a severe debug executable (recommended after heavy developments)
PROF	Builds an executable for profiling
SPMD	Builds an executable for parallel applications (with NP$\neq$1)
M3D	Three dimensional case if activated
DECOMP_2D	Enables 2D domain decomposition for parallel runs
MAXX3D; MAXX2D	Total number of cells along x in 3D and 2D (with ghost cells at lateral boundaries)
MAXY3D; MAXY2D	Total number of cells along y in 3D and 2D (with ghost cells at lateral boundaries)
MAXZ3D; MAXZ2D	Total number of cells along z in 3D and 2D
NHYDRO	Total number of hydrometeors (different from nh, the number of modeled hydrometeors)
FINE	Allows grid refinement in the vertical direction
NEWRUN	Starts a completely new run if activated or restarts from previous results
AUTO_INIT	Initializes the flow field using hard-coded functions if activated (reads in file otherwise)
CORIOLIS	Activates Coriolis forces
RADIA	Activates the radiation module
CHEM	Activates the chemistry module
AEROSOL	Activates the aerosol module
ADV_AEROCHEM	Activates advection/mixing of chemistry and aerosol scalars
FORCE_SUB	Assumes a forced large scale horizontal divergence if activated
SGS	Activates
TKE	1st order SGS closure with transported TKE
NUDGE	Nudging of the transported quantities
LSADV	Adding large scale scalar tendencies
MEANDATA	Outputs averaged results if true
FORM	Outputs results in formatted files if true
PROFILE	Outputs mean vertical profiles (horizontally averaged results)
PROF_TS	Outputs time series of mean vertical profiles
OVERWRITE	Overwrites previous outputs if true: only the last output is kept
OUTPUTALL	Outputs one separate file after each output event
PRESS_DIAG	Diagnostic pressure evaluation if true (prognostic otherwise)
ADV4	4th order advection scheme
ADV_LW	Scalar advection using a 2nd order Lax-Wendroff limited scheme
ADV_SMO	Scalar advection using Smolarkiewicz's MPDATA algorithm
ADV_TVD	Scalar advection using a TVD 3rd order limited scheme
ADV_MUSCL	Scalar advection using MUSCL 3rd order reconstruction
ADV_WENO	Scalar advection using 3rd order WENO scheme
RK3	3rd order TVD Runge-Kutta
RK4	4th order Runge-Kutta
LF	Leap-Frog integration method
SAT_ADJ	Saturation adjustment method if true (no direct integration of diffusional growth)
SEIFERT	Uses Seifert & Beheng warm microphysics
ICE_NUC	Activates detailed ice nucleation scheme (not yet completed)
CCN_ACTIV	Detailed CCN activation modelling
caseid	Case name

Table 7: *Environement variables defined in start.*

soundings will be written in *profiles.dat* files (again with an extension specifying the time) whereas the profile time series will be written in a single file named *profiles.2d.tec.* Except for the 1D profiles, all the other files are output in Tecplot format. These files can be read directly by Tecplot but also by other visualization softwares such as ParaView. More formats can be added easily.

4.4 Restart files

When the NEWRUN option in *start* is turned to FALSE, the MIMICA code can start a simulation from results obtained during a previous run. A restart file is required for this operation, called *restart.dat* (in the folder DATA), which is automatically generated after each *ires* second or at the end of a successful run. *restart.dat* is a binary file where all the necessary quantities, that is all the prognostic and diagnostic results of the previous run, are gathered in a single data block. When NEWRUN is selected, the usual field initialization is skipped and *restart.dat* is read.

For a restart simulation, a second file is required containing the initial soundings used to initialize the original run but also as reference states: PT0. This file is created and employed in the same way as *restart.dat* but contains only the reference state under the form of 1D soundings.

5 Main routines and data structures

5.1 Main subroutines and overall hierarchy

Without drawing the entire tree chart of the code, here is described the role played by some of the more important subroutines and their interlinks with the rest of the model (we exclude all the shared and typedef modules as well as most of the routines used by the detailed chemistry and aerosol modules). Three other public packages used by the code could also be added to this list: the LSODE solver, the Fast Fourier Transform package (fftpack) and the tridiagonal system solver. In addition, some elementary functions to calculate derived thermodynamical and microphysical properties can be found in the *typedef_thermo* and *typedef_hydro* structures.

- **mimica.f90** is the main routine of the program. Its only role is to define and read the input parameters stored in a common block and to provide an interface for *modelctl*.

- **gridno.F** is created automatically when the code is compiled. Its content is defined in the *wgrid.csh* script and consists mainly in some fixed dimensions such as the domain size.

- **modelctl.f90** calls the subroutines in charge of the allocation of the data structures (defined as pointers in the parallel case), the initialization, the time-stepping (in *time_step.f90*) and the outputs. It constitutes the main driving routine of the model.

- **mpi_prepare.f90** performs a certain number of preparatory tasks for MPI simulations: array and data structure allocation, communication of the initial parameters to all the procs, MPI types definition and domain decomposition along 1 or 2 dimensions.

- **setrun.f90** initializes the simulation. Different options are available: the 2D or 3D fields can be initialized using prescribed soundings defined at the end of *initial.f90* (stored in pt0, w0, p0.... in *shared_data*) or by using output file produced by a previous run in *restart.f90*.

- **initial.f90** creates the initial fields. At the moment this can only be done through prescribed functions for all the necessary prognostic quantities. The initialization is of course case dependent, see *init_pro* at the end of the file for examples. The initial soundings can be perturbed in order to initiate natural convection either by creating a thermal bubble inside the domain or using weak random perturbations in the ptil and qv fields.

- **time_step.f90** contains the interface for the choice of the numerical integration among 3 methods: *normal_phys_integ*, *rk4_phys_integ* and *forward*. The three solvers are built in the same way where the different stages of the integration step are treated separately: first, the prognostic thermodynamic quantities are evaluated for the current time step by *cal_thermo* (in *funcpack*), then momentum equations are solved by *udcomp* or *ud_tq_rk*, the scalar equations (for pt, qv and the hydrometeor moments) are solved during the next step by *tqcomp* or in *ud_tq_rk*,and finally the pressure solver *pressure_solver* is invoked to evaluate the diagnostic pressure and update the velocities.

- **ud.f90** integrates the momentum equations using the Leap-Frog method. The SGS term is first computed by *efud* (in *subgrid.f90*), and the advective tendencies are then evaluated by the routines contained in *windadv.f90*. The Coriolis forces are added if required and the buoyancy term for the vertical velocity is calculated by *buoyancy* (in *funcpack*). The boundary conditions are then imposed via *vbc* and *lbc* before the equations are integrated. Smoothing and damping at the top are provided if necessary. A prognostic equation for the TKE is also solved on the same model by *tkecomp* if required. The divergence of the momentum tendencies used by the pressure solver are also calculated here by *gradtend*. Note that the wind velocities wind1 and wind2 are not yet updated, the time step is completed only after the pressure solver.

- **subgrid.f90** contains a package of subroutines used to calculate the sub-grid scale fluxes for all the variables. *kcomp* evaluates the eddy diffusivity K (Smagorinsky-Lilly model or prognostic equation for the TKE), and *efud* and *eff* evaluate the SGS term for momentum and scalars respectively.

- **sgspack.f90** calculates the SGS terms for all the quantities using the precalculations made in *subgrid.f90*. Calculates also the surface turbulent fluxes using either fixed heat fluxes or fixed surface temperature.

- **funcpack.f90** is a package containing several useful functions called within the model. It includes for example the sponge layer damping, the Asselin filter and some functions used to calculate diagnostic properties (such as fluxes and horizontal averages).

- **integpack.f90** is a package used for time integration. It contains the RK3 and RK4 formula as well as the interface for the calls to LSODE (although not used any more).
- **bc.f90** provides the different routines used to treat the vertical and lateral boundary conditions.
- **tq.f90** solves governing equations for pt, qv and the hydrometeor moments following the same pattern as *udcomp* (Leap-Frog method). The subroutines called are however specific to the treatment of scalars (f.i., scalar advection is calculated by *advq_xxxx.f90*). There is no decoupling of dynamics and microphysics so that all the tendencies are collected and integrated at the same time and all the scalars are updated at the end of the routine.
- **advq_xxx.f90** represent a set of routines performing scalar advection. xxxx must be replaced by the name of the schemes, namely lw, tvd, muscl, weno and smo.
- **ud_tq_rk.f90** integrates momentum and scalar equations at the same time following the Runge-Kutta 44-2S method. Because four subiterations are required to complete the time-step, the momentum and scalars have to be integrated in parallel to ensure 4th order accuracy. Although this differs significantly from the ud and tq routines, the overall procedure for integration (calculation of the different contributions) remains the same.
- **windadv.f90**, **advq_fd.f90**, **advq_lw.f90**, **advq_tvd.f90**, **advq_muscl.f90** and **advq_weno.f90** compute the advective fluxes for momentum and scalars respectively. For scalars, the different schemes are selected in *start*.
- **microphysics1.f90** calculates the sources and sinks related to hydrometeor collision-coalescence processes, for both mass mixing ratios and number concentrations. The subroutines called (corresponding to each hydrometeor interaction) are included in *micropack1.f90*.
- **microphysics2.f90** calculates sources and sinks related to evaporation/condensation and sublimation/deposition. When the saturation adjustment procedure is selected, rain drops and cloud droplet sources are not updated here. The analytic implicit approximation for supersaturation is also calculated here. All the subroutines called are contained in *micropack2.f90* (including the saturation adjustment scheme). The sources calculated here are added to the overall tendencies for integration in tq.
- **micropack1.f90** and **micropack2.f90** contain the subroutines evaluating each individual microphysical sources resulting from collision-coalescence processes or phase transitions. They make use of several elementary functions defined in *shared_hydro* and calculating some characteristic properties of each hydrometeor distribution (like mean diameters).
- **micro_diag.f90** contains all the "diagnostic" microphysics. This includes CCN activation, drop/droplet freezing (heterogeneous ice nucleation) and the saturation adjustment scheme.
- **micro_ice.f90** calculates all the processes related to ice microphysics only and especially ice nucleation.
- **pressure_solver.f90** contains all the routines needed to evaluate diagnostically the pressure and update the velocities after the time step. The pressure solver calls the fast fourier transform code *fftpack.F* and the tridiagonal solver *tridiag.f90*, both defined externally. This routine should be considered as a black box.
- **grad.f90** calculates the pressure gradient term for momentum equations, the momentum tendency divergence for the pressure solver and the velocity divergence.
- **check.f90** checks the potential temperature as well as the CFL criterion after each time-step to determine if the simulation is stable. If not, either the time-step is decreased so that the max CFL becomes lower than the one prescribed in *cm.nml* or the run is stopped.
- **timevar.f90** updates time varying quantities. It is called once per timestep and used predefined functions to determine the values of parameters such as surface fluxes and large scale scalar tendencies as functions of time.
- **calmean.f90** calculates the spatial and temporal averages that can be output if requested.
- **output.f90** controls all the possible outputs: instantaneous and mean fields, vertical profiles, time series... *outputpack.f90* contains the subroutines called by output.
- **radiag.f90** is the main routine calling the radiation solver *radiatov.f90* or computing the case dependent radiative fluxes.

- **chemfront.f90** is the interface for the progress of the chemical concentrations. The advection of chemical species is performed by *advc_m*.
- ă**advcm.f90, advcsub.f90** are two routines used to transport the prognostic aerosols, chemical species or nuclei. Advection is performed using Bott's integral scheme and SGS mixing using the same method as the other scalars. These two processes can be switched off by setting ADV_AEROCHEM to FALSE in *start*.
- **exchange.f90**, **collect.f90**, **distribute.f90** and **distexch.f90** provide the tools for MPI communications.

5.2 data structures

The model uses several data structures for all the prognostic quantities as well as for some important diagnostic quantities. The use of data structures in fortran has many advantages and contributes to the improvement of the code's readability. In the present case, data structures are invoked by calling *USE shared_xxxx* at the beginning of a routine, where *shared_xxxx* is a module contained in the corresponding .f90 routine. They are made of two parts: the type definition, called by *USE typedef_xxxx*, containing the different elements of the data structure (f.i. in the wind structure, the type *wind* includes the three components *u, v, w*), and the structure definition, where different names are declared and assigned to the given type.

The structures are usually 2D or 3D arrays (4D for hydrometeors) where the three dimensional fields or 2D horizontal slices (for example at the surface) of major prognostic and diagnostic quantities are stored. All the structures defined in the code are saved and shared so that one can use the content of any given structure anywhere in the code simply by making use of the required shared module. The use of such structures limits possible errors related to bad array declarations/allocations.

Note that the original fortran modules, containing the chemistry and aerosol properties, are used in different manner by some specific routines where array declarations are performed through the inclusion of .h files (the array names still need to be passed in the subroutine calls). This way of coding was generalized in the CRM version of the model but should be changed in a near future to follow the new coding standards.

The data structures defined so far are described in the following table. The structure *shared_data* is not referenced here. It contains the definition of many constants, indices, input or dimensional parameters used throughout the model. Recently, the most common universal physical constants used throughout the code were also defined in that file. As a result, physical constants should no longer explicitly appear in the model: they should all be defined first in *shared_data.da* and used directly through the use of this data structure.

Note that the library *typedef_hydrometeor* defines several hydrometeor types, namely *hydrometeor*, *hydro_second* and *hydro_param* (the structure names associated with these types are defined in *shared_hydro*). Each new type has its own specific purpose: *hydrometeor* defines the pure hydrometeor quantities transported by the model, that is the mass mixing ratio and number concentration of each kind; *hydro_second* defines secondary hydrometeor quantities, related to n and q, but only necessary for microphysical process calculations (λ parameter in the size distribution, the terminal fall speeds for precipitation and the index ifp locating the presence of each hydrometeor kind); finally, *hydro_param* defines all the constants used for each kind in the microphysics, that is for the mass-diameter law, for the fall speed-diameter law and for the size distribution. The definition of these three distinct hydrometeor types has led to an improved readability of the microphysical module (especially for the collision processes) and more flexibility for the hydrometeor's laws (new hydrometeor kinds can be used simply by changing the *hydro_param* coefficients).

In addition, two shared modules, namely *shared_hydro* and *shared_thermo*, contain also elementary functions used to calculate some derived properties of the given type. For example, *shared_hydro* includes functions to evaluate the regular and mass weighted averaged terminal fall speeds, averaged hydrometeor diameters, the size distribution parameter λ and several important constants. Note that these functions only use the constant parameters declared in *hydro_param* so that a change in the hydrometeor properties will be automatically taken into account in these calculations. *shared_thermo* allows the calculation of the potential temperature, the absolute pressure, the saturation vapor pressures and mixing ratios and their derivatives with respect to temperature. These functions can be used anywhere in the model to evaluate these derived quantities as long as the corresponding module is in use.

Structure name	dimensions	elements	meaning	purpose
shared_wind	nux*nvy*nwz	u (m/s)	wind vel. along x	Contains the wind velocities. wind1 refers to the
		v (m/s)	wind vel. along y	instantaneous winds at time t ($t-1$ in
		w (m/s)	vertical wind vel.	LeapFrog), wind2 at time t and wind0 at $t=0$ (ini
shared_state	nx*ny*nz	p (-)	Exner perturbation pressure	Prognostic and diagnostic state variables.
		ptil (K)	Ice/liquid potential temperature	state1 refers to the instantaneous wind at time
		T (K)	Temperature	($t-1$ in Leap Frog), state2 at time t
		qv (kg/kg)	Vapor mixing fraction	Diagnostic
		qt (kg/kg)	Total water mixing fraction	Prognostic
shared_hydro	nx*ny*nz*nhydro	hydromtr%n (1/kg)	number density	Parameters for the hydrometeor populations
		hydromtr%q (kg/kg)	mass density	(droplet, rain, ice, graupel, snow)
		hydro_second%lambda	parameter of the size distribution	
		hydro_second%vp	terminal fall speeds	
		hydro_second%ifp	presence of hydrometeors (integer)	
		hydro_param%cm, ctv, am, btv, nu, cvp1, cvp2	Coefficients for microphysics laws	
shared_nuc	nx*ny*nz	ccn (kg/kg)	mass density of CCNs	Mass densities of CCNs and INs if the
		in (kg/kg)	mass density of INs	aerosol module is not used
shared_surf	nx*ny	rain (kg/m2)	rain flux at ground level	Horizontal fluxes calculated at ground
		grau (kg/m2)	graupel flux at ground level	level for different precipitating hydrometeors
		snow (kg/m2)	snow flux at ground level	
		ustar (m/s)	Surface friction velocity	
shared_thermo	nx*ny*nz	pp (-)	Total Exner pressure	Diagnostic thermodynamics properties, used
		pt (K)	potential temperature	mainly by the microphysics scheme
		ptv (K)	virtual potential temperature	thermo1 is evaluated using state1 variables
		sat (kg/kg)	Absolute supersaturation over liquid	
		sati (kg/kg)	Absolute supersaturation over ice	
		qsw (kg/kg)	saturation vapor content (water)	
		qsi (kg/kg)	saturation vapor content (ice)	
		dens (kg/m3)	Local density	Calculated using perfect gas law
shared_turbu	nx*ny*nz	fk (m2/s2)	Momentum eddy diffusivity	
		fh (m2/s2)	Scalar eddy diffusivity	$fh = fk/Pr$
		kres (m2/s2)	resolved TKE	related to the dynamics and turbulent model: resol
		ksgs (m2/s2)	SGS TKE	and SGS quantities
		def (kg/kg)	squared deformation tensor	
		N2 (1/s)	Brunt-Vaisala frequency	
		buoy	Buoyancy	Vertical acceleration
		bfres	Buoyancy production of TKE (resolved)	Diagnostic
		bfsgs	Buoyancy production of TKE (SGS)	Diagnostic
		wvar	Variance of vertical velocity	
		wske	Skewness of vertical velocity	
shared_rad	nx*ny*nwz	dtnet (K/m3)	net temperature tendency	through simplified or detailed model
		fluxsu (K/m2)	upwind shortwave radiative flux	
		fluxsd (K/m2)	downwind shortwave radiative flux	
		fluxiru (K/m2)	upwind longwave radiative flux	

References

[1] P. Bougeault. A non-reflective upper boundary condition for limited-height hydrostatic models. *Monthly Weather Review*, 111:420–429, 1983.

[2] J.S. Chang C. Wang. A three-dimensional numerical model of cloud dynamics, microphysics, and chemistry: 1. concepts and formulation. *Journal of Geophysical Research*, 98:14,827–14,844, 1993.

[3] J.-P. Chen and D. Lamb. The theoretical basis for the parameterization of ice crystal habits: growth by vapor deposition. *Journal of Atmospheric Sciences*, 51:1206–1221, 1994.

[4] T.L. Clark. Numerical simulations with a three dimensional cloud model: lateral boundary condition experiments and multicellular severe storm simulations. *Journal of Atmospheric Sciences*, 36:2191–2215, 1979.

[5] J.W. Deardorff. Three dimensional numerical study of turbulence in an entraining mixed layer. *Boundary-Layer Meteorology*, 7:199–226, 1974.

[6] D.R. Durran. *Numerical methods for fluid dynamics with applications to geophysics, 2nd edition.* Texts in Applied Mathematics, Springer, 2010.

[7] B. Stevens et al. Evaluation of large eddy simulations via observations of nocturnal marine stratocumulus. *Monthly Weather Review*, 133:1443–1462, 2005.

[8] Q. Fu and K.-N. Liou. Parameterization of the radiative properties of cirrus clouds. *Journal of Atmospheric Sciences*, 50:2008–2025, 1993.

[9] J.R. Garratt. *The atmospheric boundary layer.* Cambridge University Press, 1994.

[10] S. Gottlieb and C.-W. Shu. Total variation diminishing runge-kutta schemes. *Mathematics of Computation*, 67:73–85, 1998.

[11] W. Hundsdorfer, B. Koren, M. Van Loon, and J.G. Verwer. A positive finite-difference advection scheme. *Journal of Computational Physics*, 117:35–46, 1995.

[12] G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted eno schemes. *Journal of Computational Physics*, 126:202–228, 1996.

[13] D.I. Ketcheson. Runge-kutta methods with minimum storage implementations. *Journal of Computational Physics*, 229:1,763–1,773, 2010.

[14] V.I. Khvorostyanov and J.A. Curry. Terminal velocities of droplets and crystals: Power laws with continuous parameters over the size spectrum. *Journal of Atmospheric Sciences*, 59:1872–1884, 2002.

[15] A. Kurganov and E. Tadmor. New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations. *Journal of Computational Physics*, 160:241–282, 2010.

[16] H. Morrison, J.A. Curry, and V.I. Khvorostyanov. A new double-moment microphysics parameterization for application in cloud and climate models. part i: Description. *Journal of Atmospheric Sciences*, 62:1,665–1,667, 2005.

[17] H. Morrison and W.W. Grabowski. Modeling supersaturation and subgrid-scale mixing with two-moment bulk warm microphysics. *Journal of Atmospheric Sciences*, 65:792–812, 2008.

[18] H.R. Pruppacher and J.D. Klett. *Microphysics of clouds and precipitation, 2nd edition.* Kluwer Academic Publishers, 1997.

[19] R.R. Rogers and M.K. Yau. *A short course in cloud physics, 3rd edition.* Butterworth-Heinemann, 1989.

[20] A. Seifert and K.D. Beheng. A double moment parameterization for simulating autoconversion, accretion and selfcollection. *Atmospheric Research*, 59-60:265–281, 2001.

[21] A. Seifert and K.D. Beheng. A two-moment cloud microphysics parameterization for mixed-phase clouds. part i: Model descroption. *Meteorology and Atmospheric Physics*, 92:45–66, 2006.

[22] J. Smagorinsky. General circulation experiments with the primitive equations: 1. the basic experiment. *Monthly Weather Review*, 911:99–164, 1963.

[23] D.E. Stevens, A.S. Ackerman, and C.S. Bretherton. Effects of domain size and numerical resolution on the simulation of shallow cumulus convection. *Journal of Atmospheric Sciences*, 59:3285–3301, 2002.

[24] P.D. Williams. A proposed modification to the robert-asselin time filter. *Monthly Weather Review*, 137:2538–2547, 2009.

[25] Z. Xu and C.-W. Shu. Anti-diffusive flux corrections for high order finite difference weno schemes. *Journal of Computational Physics*, 205:458–485, 2005.