MIMICA V5.1, User guide

Julien SAVRE
Meteorological Institute, Ludwig-Maximilians-Universität, Munich
julien.savre@lmu.de

Matthias Brakebusch
Dept. of Environmental Science and Analytical Chemistry, Stockholm University, Stockholm
matthias.brakebusch@aces.su.se

28.04.2021

Contents

1 Introduction

MIMICA is a local-area, 3D high-resolution atmospheric model written entirely in fortran. Current and past applications of the model include: very high-resolution simulations (Large-Eddy simulation) of boundary-layer stratocumulus clouds in Arctic and marine environments, tropical and midlatitude continental convection and its diurnal cycle, and Radiative-Convective Equilibrium over the tropical ocean.

The code is an extension of the CRM-MIT model developed by Chien Wang and Julius Chang [2]. Since then, the model has been substantially modified and upgraded by Julien Savre at Stockholm University. Most of the original code has been rewritten using modern fortran 90 and MPI standards. Previously released MIMICA versions that have already been extensively used include version V4.0 (2014 [29], no longer maintained) and V5.0 (2017). The current version (V5.1) has been released in 2021.

The present document is intended to new MIMICA users who wish to start working with the model, or more experienced users who are interested in knowing more about the code. First and foremost, the vocation of this document is to list and explain all the different options accessible through configuration files. We also give here some guidelines to compile the code and run your first simulations with a personalized configuration. In addition, you will be able to find information regarding the default numerical schemes and physical modules implemented in MIMICA. It is not intended here to give a complete description of all implementations. Instead, we wish to provide here the theoretical background needed to understand the model and some of the choices made during its development. Interested readers are invited to refer to the cited literature for more details.

MIMICA is presently developed under the git version control system and is hosted by the Bitbucket platform. The model can be found and downloaded at the following address: `https://bitbucket.org/matthiasbrakebusch/mimicav5/src/master/`. Besides, additional help and information regarding the model, how to set it up, and our monthly MIMICA community meetings can be found at: `https://uppslag.aces.su.se/mimica/mimica`.

This document is organized as follows. Section 2 gives all necessary guidelines to setup, compile and run MIMICA. Anyone interested in running the model should pay close attention to this section. Section 3 is intended to give the theoretical background needed to understand how MIMICA works and what it does. Section 4 gives a detailed description of all option flags available to control the model's configuration. It should be noted that while section 3 mostly insists on a "default" MIMICA setup, the option flags and keywords detailed in section 4 can help design simulations making use of more advanced configurations. Section 4 also includes a brief description of how initial conditions are defined. Section 5 introduces the reader to all output files produced by the MIMICA model, and is of great help for those wishing to know more about how to analyse the model's results. Finally, section 6 presents a list of available predefined test cases that can be used either for testing and debugging, or as the basis for more advanced, personalized configurations.

2 How to compile and run MIMICA

2.1 Required libraries

First and foremost, in order to compile and run MIMICA properly, you will need a recent compiler (both gfortran and intel are currently supported), as well as an MPI library for parallel simulations (*SPMD TRUE* in *start*). Before starting the compilation (see next section), make sure that the corresponding library paths are properly defined in your environmental variables. Besides, in the more general case, MIMICA needs to be link to two other external libraries: **netCDF** [27] and **fftw** [8].

The netCDF library is used to generate output files in the netCDF format. There is a good chance that the library will be available on your system. If it is properly loaded, linking is done automatically at compile time (see below). The only requirement is that you use a netCDF package compiled with a compiler that is compatible with the one used to compile MIMICA.

The fftw (which stands for Fastest Fourier Transform in the West) library is required when MIMICA is run with *ANELASTIC TRUE* in *start* (this implies that a pressure correction equation is solved using a FFT based algorithm).

- If fftw is centrally built and available on your local server, it can be directly linked to MIMICA V5 with only small modifications. In order to use the central fftw library, MIMICA should be compiled with Makefile_centralfft which must first be copied into your local work directory, and renamed Makefile. If the environment variables FFTW_INC and FFTW_LIB are not already defined on your system, you should define those yourself. Again, make sure that the loaded fftw library has been compile with a compatible compiler.

- If fftw is not available or if you do not wish to link to a centrally distributed version of fftw, you will need to download the library from the official fftw website (*http://www.fftw.org*) and compile it yourself. Once fftw has been successfully built, MIMICA can be compiled by using the default Makefile present in the main mimicav5 directory. Makefile can then be copied in your local work directory without any further modification.

2.2 Compiling MIMICA

Before going any further and proceeding with MIMICA's compilation, a new environment variable must be defined to specify the main path to your MIMICA directory. To do this, you must simply type: *export MIMICA=.....*, where you must specify your own MIMICA path name. For future uses, you can directly add this line to your *./profile* file.

Compiling MIMICA is performed using the *start* script which must be edited to compile with the desired physical and numerical packages. *start* templates are provided for a variety of idealized test cases in the *./templates* folder. The most important compilation flags that must be edited in *start* are:

- *NETCDF*: path to your netCDF directory.

- *NPX, NPY*: number of processors.

- *CMPLER*: choice of compiler. The default compiler is gfortran. If *setenv CMPLER INTEL* is uncommented, the intel fortran compiler is used (make sure that a recent intel compiler is available and loaded on your system).

- *DEBUG*: if *TRUE*, MIMICA is compiled with debug options.
- *SPMD*: if *TRUE*, MIMICA is compiled for parallel runs (*NPX* must be > 1).

All other flags in *start* control the numerical or physical configuration.

Once your *start* file is ready, you must provide one of the two Makefiles available in the parent directory as described in section 1a above. Compilation is then performed by typing: *./start*

All objects and modules are stored in *./build*. The file *compile.log* created in your work directory contains a summary of the compilation with all the different start options selected. Finally, the executable *mimicav5_XXXXXX.exe* is created in your local MIMICA directory, with XXXXXX corresponding to the current version number from which your executable has been built.

2.3 Running MIMICA

Running MIMICA can be done in serial by typing: *./mimicav5_XXXXXXX.exe*, or in parallel with (for example): *mpirun -np N mimicav5_XXXXXXX.exe* (with N the number of processors). Please consult your HPC help page to get more information on how to properly run a parallel job.

In order to run MIMICA, a minimum of 2 input files must also exist in your local directory, *cm.nml* and *out.nml*, as well as appropriate initial conditions in *./INCLUDE*.

cm.nml contains a list of model options and parameters. *cm.nml* templates are available for idealized cases in *./templates*. The file must be present (copied from a template) in your local working directory and edited as desired. *out.nml* is used to define the quantities you want to ouput using various keywords. An example is available in the main MIMICA repository.

Initial conditions will be prescribed using either a specific include file, .h, which must be present in *./INCLUDE*, or by creating your own initial sounding, again in *./INCLUDE* (the name of the initial sounding file must be prescribed in *cm.nml* under *file_init*).

2.4 Issues with MIMICA

Despite our best efforts to provide a bug-free code working in any possible configuration, we cannot guarantee that this is effectively the case. This subsection describes briefly what should be done in case MIMICA fails unexpectedly.

Before taking any special action, following these simple and straightforward steps may help resolve many issues:

1. First and foremost, you should obviously make sure that the model is configured properly. Things that need particular attention include the numerical grid, initial conditions, and stability criteria (for example, the CFL number should be kept close to 0.5).

2. A simulation that crashes unexpectedly despite a correct configuration might indicate the presence of coding errors. Finding out where such errors may be can be difficult, but using debug options can help greatly. For example, setting *ldebug* to *.true.* in *cm.nml* indicates in which part of the code the model stopped by printing out a short line of text in *cm.prt* when the code enters or leaves specific routines. Besides, when MIMICA is compiled with *DEBUG TRUE* in *start*, compiler specific debug flags are turned on and a debug executable is created. When run in debug mode, the code will stop whenever an error is encountered, and the exact place where this error was found is specified in your simulation's log file.

3. If the simulation's design is complex, perhaps making use of many of the available physical parameterizations, it can be difficult to figure out what part of the model is responsible for a reported error. In this situation, it can be beneficial to simplify the problem by changing the configuration step-by-step. For example, some of the parameterizations may be turned off (radiation or aerosols), while different choices can be made for others (microphysics, numerical schemes). Similarly, downscaling the configuration (for example, going from 3D to 2D, or making each dimension smaller) can help executing your tests faster.

If following these advises allowed you to identify a bug or modelling error, you can choose to either resolve the issue yourself by correcting the code and submitting the commit to the main git repository, or report the issue for someone else to correct it. When reporting an issue, it is preferable to use the "Issues" section on MIMICA's bitbucket page (`https://bitbucket.org/matthiasbrakebusch/mimicav5/src/master/`). A detailed description of the problem should then be given. Alternatively, you can contact Julien Savre directly by email: julien.savre@lmu.de.

3 Model description

The description proposed in this section focuses on a "default" configuration of the MIMICA model. It should be reminded that MIMICA possesses many more capabilities which are not detailed below for the sake of clarity. Some of these features are however mentioned when appropriate, and more information can be found in the description of the *start* and *cm.nml* files given in section 4.

3.1 Governing equations

MIMICA solves the basic conservation equations for mass (continuity), momentum, potential temperature and total water content in 2 or 3 dimensions. Depending of the level of complexity and physical parameterizations requested, a set of additional conservation equations for microphysical quantities (typically the mass and number concentrations of certain hydrometeor classes) as well as passive tracers can also be solved.

Two different dynamical cores are available in MIMICA: **an anelastic core**, in which the density is assumed to depend on altitude only so that the continuity equation reduces to a diagnostic equation constraining momentum to be divergence free, and **a compressible core**, in which no approximation for the density is made and the continuity equation is solved explicitly. All the equations presented hereafter are written according to the anelastic approximation of [20] as this is the preferred option in MIMICA (*setenv ANELASTIC TRUE* in *start*). More detail about the compressible core are given in section 3.2.2.

The momentum equations can be written as follows:

$$\frac{\partial \rho_0 u}{\partial t} + \nabla \cdot (\rho_0 \mathbf{u} u) = -\rho_0 \frac{\partial}{\partial x}\left(\frac{p'}{\rho_0}\right) + f_0 v + \nabla \cdot (\rho_0 \tau_u) + S_u \quad (1)$$

$$\frac{\partial \rho_0 v}{\partial t} + \nabla \cdot (\rho_0 \mathbf{u} v) = -\rho_0 \frac{\partial}{\partial y}\left(\frac{p'}{\rho_0}\right) - f_0 u + \nabla \cdot (\rho_0 \tau_v) + S_v \quad (2)$$

$$\frac{\partial \rho_0 w}{\partial t} + \nabla \cdot (\rho_0 \mathbf{u} w) = b - \rho_0 \frac{\partial}{\partial z}\left(\frac{p'}{\rho_0}\right) + \nabla \cdot (\rho_0 \tau_w) + S_w \quad (3)$$

$\mathbf{u}$ is the velocity vector (u, v, w), b is the buoyancy, τ_u, τ_v, τ_w are the contributions from subgrid scale turbulent diffusion to u, v and w respectively (defined in section 3.4.1), f_0 is the Coriolis parameter (f-plane approximation), ρ_0 is the reference anelastic density (independent of time and varies only with altitude), and $p' = p - p_0$ is the perturbation pressure defined with respect to the base state hydrostatic pressure. S_u, S_v and S_w are additional source terms that may represent nudging or large-scale forcing. The buoyancy b is defined as a function of virtual potential temperature:

$$b = \rho_0 g \left(\frac{\theta_v}{\theta_{v0}} - 1\right) = \rho_0 g \left[\frac{\theta\,(1 + \epsilon q_v - q_l - q_i)}{\theta_0\,(1 + \epsilon q_{v0})} - 1\right] \quad (4)$$

with θ being the potential temperature, q_v is the water vapor mixing ratio, q_l and q_i are the liquid and ice mixing ratios respectively, g is the gravitational acceleration (constant, set to 9.81 ms^{-2}) and $\epsilon \approx 0.602$. θ_0 and q_{v0} are here defined as the base state potential temperature and vapor mixing ratio (independent of time, functions of altitude only). The different terms appearing in the momentum budget equations can be turned on or off individually using various options in *cm.nml* and *start*.

The potential temperature, total water mixing ratio and passive tracer equations read:

$$\frac{\partial \rho_0 \theta}{\partial t} + \nabla \cdot (\rho_0 \mathbf{u} \theta) = \mathcal{Q}_{rad} + \mathcal{Q}_{micro} + \nabla \cdot (\rho_0 \tau_\theta) + S_\theta \quad (5)$$

$$\frac{\partial \rho_0 q_t}{\partial t} + \nabla \cdot (\rho_0 \mathbf{u} q_t) = \nabla \cdot (\rho_0 \tau_{qt}) + \mathcal{Q}_{prec} + S_q \quad (6)$$

$$\frac{\partial \rho_0 \varphi}{\partial t} + \nabla \cdot (\rho_0 \mathbf{u} \varphi) = \nabla \cdot (\rho_0 \tau_\varphi) + S_\varphi \quad (7)$$

with $\mathcal{Q}_{rad}$ and $\mathcal{Q}_{micro}$ being source terms representing radiative and latent heating and cooling respectively, $\mathcal{Q}_{prec}$ is a sedimentation term, q_t and φ are the total water mixing ratio and passive tracer, while τ_θ, τ_{qt}, τ_φ are the contributions from subgrid scale turbulent diffusion. Again, S_θ, S_q and S_φ are additional sources including for example nudging or large-scale advection. Note that the potential temperature is defined as:

$$\theta = T \left(\frac{p}{p_{00}} \right)^{-c_{pa}/R_a}, \quad (8)$$

with T the absolute temperature, $p_{00} = 1000\ hPa$, $cp_a = 1004\ JK^{-1}kg^{-1}$ and $R_a = 288\ JK^{-1}kg^{-1}$ the dry air specific heat capacity at constant pressure and specific ideal gas constant for dry air respectively. As an alternative to the potential temperature equation, MIMICA an also operate with the frozen moist static energy (MSE) as the conserved energy variable (available by setting *setenv ISENTROPIC FALSE* in *start*). The frozen MSE is often used to simulate tropical convection as it is almost exactly conserved upon adibatic processes and phase changes, and is directly related to the enthalpy (and therefore to the first law of thermodynamics).

Finally, the system of equations is closed using the ideal gas law for moist air to relate all thermodynamic properties:

$$p_0 = \rho_0 R T \quad (9)$$

with $R = (1 - q_t) R_a + q_v R_v$ the ideal gas constant for moist air ($R_v = 461.5\ JK^{-1}kg^{-1}$ being the ideal gas constant for water vapor). By default, the vapor contribution to R is neglected so that $R = R_a = 288\ JK^{-1}kg^{-1}$. This can however be changed by setting *cst_cp = 0* in *cm.nml*, making all heat capacities dependent on water vapor, liquid and ice mass mixing ratios. More information regarding thermodynamically consistent equations and relationships can be found in [28].

3.2 The anelastic and compressible cores

3.2.1 The anelastic solver

In its default configuration, MIMICA employs the anelastic approximation of [20] in which the density is assumed to vary only with altitude and to be time independent. This enables us to filter out acoustic waves to improve the model's stability and overall performance.

Before going any further, a few important notations have to be introduced. We denote $\mathbf{u}^n$ the velocity vector estimated at time t^n, that is at the beginning of the time-step. $\mathbf{u}^{n+1}$ then denotes the velocity vector estimated at time $t^{n+1} = t^n + \Delta t$. We similarly define p'^n and p'^{n+1} the pressure perturbation estimated at time t^n and t^{n+1}.

The idea behind the anelastic solver is to estimate the perturbation pressure p' in equations 1-3 implicitly (i.e. at time t^{n+1}). To do so, the anelastic continuity equation [5]:

$$\frac{\partial \rho_0 u}{\partial x} + \frac{\partial \rho_0 v}{\partial y} + \frac{\partial \rho_0 w}{\partial z} = 0 \quad (10)$$

is used as a diagnostic constraint to find p'^{n+1} and enforce mass conservation at the end of the time step. The procedure (often called projection method, [5]) can generally be decomposed into three steps:

1. The momentum equations 1-3 are first solved with all terms (including the pressure gradient term) estimated at time t^n. At this stage, the updated velocity vector $\mathbf{u}^*$ does not satisfy the continuity equation 10.

2. Next, a pressure correction δp is introduced to update $\mathbf{u}^*$ and yield the momentum vector at the end of the time step $\mathbf{u}^{n+1}$:

$$\rho_0 \mathbf{u}^{n+1} = \rho_0 \mathbf{u}^* - \Delta t \rho_0 \nabla \left(\frac{\delta p}{\rho_0} \right), \tag{11}$$

   and:

$$p'^{n+1} = p'^{n} + \delta p. \tag{12}$$

   Taking the divergence of equation 11 yields:

$$0 = \nabla \cdot (\rho_0 \mathbf{u}^*) - \Delta t \nabla \cdot \left(\rho_0 \nabla \frac{\delta p}{\rho_0} \right). \tag{13}$$

   We have used here the fact that $\nabla \cdot (\rho_0 \mathbf{u}^*)$ must be 0. The first term on the right hand side, $\nabla \cdot (\rho_0 \mathbf{u}^*)$, can be computed easily, leaving us with an elliptic equation for δp.

3. The elliptic δp equation is then solved (see details below), and $\mathbf{u}^{n+1}$ can be obtained from equation 11.

The procedure can be repeated several times (this is specified by *nsubp* in *cm.nml* which is set to 1 by default) in order to improve the accuracy of the method.

An efficient solver based on Fast Fourier Transforms (FFT) is implemented in MIMICA to solve equation 13. Let's first rewrite equation 13 as follows:

$$\nabla^2 \left(\frac{\delta p}{\rho_0} \right) + \frac{1}{\rho_0} \frac{\partial \rho_0}{\partial z} \frac{\partial}{\partial z} \left(\frac{\delta p}{\rho_0} \right) = \mathcal{F}, \tag{14}$$

with $\mathcal{F} = \nabla \cdot (\rho_0 \mathbf{u}^*) / (\rho_0 \Delta t)$, and apply a Fourier transform in the horizontal plane to give:

$$-\left(k^2 + l^2\right) \widehat{\left(\frac{\delta p}{\rho_0} \right)} + \frac{\partial^2}{\partial z^2} \widehat{\left(\frac{\delta p}{\rho_0} \right)} + \frac{1}{\rho_0} \frac{\partial \rho_0}{\partial z} \frac{\partial}{\partial z} \widehat{\left(\frac{\delta p}{\rho_0} \right)} = \widehat{\mathcal{F}}. \tag{15}$$

Using centered finite differences to approximate the first and second order derivatives of the transformed perturbation pressure yields a tridiagonal system for $\widehat{(\delta p/\rho_0)}$ which can be efficiently inverted. Once this is done, the solution $\widehat{(\delta p/\rho_0)}$ can be transformed back into the physical space to find $\delta p/\rho_0$ at each grid point. It is then easy to calculate the gradient $\nabla (\delta p/\rho_0)$ and introduce it in equation 11 to complete the correction procedure.

The procedure presented above can only estimate the pressure perturbation to within an unknown constant. This is not a problem when updating the velocity vector as the pressure perturbation gradient is not affected by the exact value of this constant. By default, the constant of integration is computed to ensure that the domain averaged value of δp is 0.

Although very accurate and efficient, the anelastic solver also comes with its limitations. It must be noted in particular that the present FFT solver assumes by default that the domain's lateral boundaries are periodic (extending the solver to open boundaries is possible but has not been done yet). In addition, when used in parallel, the FFT solver can only handle specific combinations of grid sizes and number of processors. The following combinations are recommended:

- **In 2D**, the main dimension *MAXX* must be divisible by the number of processors squared, NP^2
- **In 3D**, both dimensions *MAXX* and *MAXY* must be divisible by *NP* (the total number of processors, *MAXX*×*MAXY*).

3.2.2 The compressible solver

By setting *setenv ANELASTIC FALSE* in *start*, MIMICA solves for the fully compressible equations of motions, which implies that the density is allowed to vary both in time and space. As mentioned previously, solving the fully compressible system of equations requires much smaller time steps to satisfy the acoustic CFL stability criterion, which often translates into a significant loss of performance. The compressible solver however has its advantages: there is no *a priori* assumption on the density and it is more flexible in the sense that using periodic or open lateral boundaries does not need any extra coding effort.

In short, the following continuity equation now needs to be solved:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = \mathcal{Q}_{prec}. \tag{16}$$

The presence of a term related to precipitation ($\mathcal{Q}_{prec}$) can be understood by noting that $q_a + q_v + q_l + q_i = 0$ (q_a being the dry air mixing ratio), and that total continuity must be retrieved when summing all equations for q_a, q_v, q_l and q_i. The conservation of water mass implies that all microphysical sources and sinks add up to 0 except for the sedimentation terms.

All the equations introduced previously still need to be solved, except that ρ_0 must now be replaced by ρ. In addition, note that divergence damping is applied (an extra term in the momentum equations, see [33]) in order to improve stability. The strength of this damping is controlled by *adiv* in *cm.nml* whose value is set by default to 0.1.

To solve the new system of coupled equations, no extra step must be taken. Because the density and potential temperature are solved and known explicitly within each grid cell, the pressure can be found using the ideal gas law. It is then enough to take the gradient of the diagnosed pressure and introduce it directly in the momentum equations.

Because the stability of the fully compressible equations is constrained by an acoustic CFL criterion requiring small time-steps, a split-explicit procedure is employed to solve the system, as described in [28, 17]. The split-explicit method involves solving for slow waves and processes at the main model time-step, while fast waves (i.e. all terms supporting acoustic waves) are solved using a smaller time-step calculated to satisfy the acoustic CFL criterion. The small time-step is defined as a fraction of the main model time-step: $\Delta\tau = \Delta t / N_{step}$, with N_{step} being the number of iterations required to solve the acoustic waves.

3.3 Numerical methods

3.3.1 Numerical grid

The grid is defined in a cartesian coordinate, in 2 or 3 dimensions. The cell size is always constant in the horizontal plane (*dx* and *dy* in *cm.nml*) but a stretched grid can be used in the vertical direction by setting *setenv FINE TRUE* in *start*. Predefined vertical grids are available for specific test cases. Alternatively, the vertical grid can be provided as an external input file named *grid.dat* containing a unique column specifying each individual grid level. Note that the surface boundary in MIMICA is always flat: the surface altitude is constant across the domain, no topography can be simulated. Besides, the surface is always located at $z = 0$ in the model, but the surface pressure *psurf* can be decreased to elevate the numerical domain.

The number of cells in all 3 directions must be set by the user in *start* (*MAXX*, *MAXY*, and *MAXZ*). The domain length in X and Y is then obtained by multiplying the number of grid cells in each direction by the cell sizes *dx* and *dy*. The number of grid points along each direction, *MAXX* and *MAXY*, is *a priori* not limited. In parallel simulations however, domain decomposition imposes that *MAXX* and *MAXY* must be exactly divisible by *NPX* and *NPY* respectively (see section 3.3.3). Note also that using the anelastic solver with the FFTW library in parallel also sets additional constraints on domain size and decomposition, as explained previously.

Cartesian grids are very convenient to discretize fluid equations using finite differences or finite volume methods. In MIMICA, a staggered grid system is employed, the C-Arakawa grid, defined as follows:

- pressure, and all other scalars, are defined at the center of each grid cell
- the three velocity components are shifted by half a cell and are therefore defined at face centers: $u_{ijk} = u\left(x + \frac{\Delta x}{2}, y, z\right)$, $v_{ijk} = v\left(x, y + \frac{\Delta y}{2}, z\right)$, $w_{ijk} = w\left(x, y, z + \frac{\Delta z}{2}\right)$

In addition, the local velocity (respectively pressure) gradients are defined at the point where pressure (respectively velocity) is defined.

In practice, because of the staggering, two different sets of vertical grid spacings have to be defined: one corresponding to the spacing between two vertical velocity points, the other for the spacing between two scalar points. These two quantities are refered to as Δz_w and Δz_p respectively.

3.3.2 Boundary conditions

Boundary conditions in MIMICA are defined by default as periodic in the horizontal plane, and solid (i.e. the vertical velocity vanishes exactly at the boundary) in the vertical. Because gravity waves are allowed to propagate in the domain and are usually reflected by the top boundary, it is often recommended to apply a sponge layer near the domain top (above *zdamp* defined in *cm.nml*), whose role is to nudge the simulated state in this region to the initial conditions with a time scale *tdamp* defined in *cm.nml* (see section 3.4.7 for more detail).

Open boundary conditions have also been implemented for the lateral boundaries, but this option is not recommended in the current version of MIMICA. It is however possible to apply pseudo-open lateral boundaries by using a large horizonal domain with the default periodic conditions, and prescribing thick horizontal sponge layers as described in section 3.4.7. Similarly, it is possible to use periodic conditions in the vertical, although this only applies to specific idealized templates. Overall, it is recommended not to modify the default specification of boundary conditions accessible through parameters *bcl(1-6)* in *cm.nml*.

3.3.3 Parallelisation and domain decomposition

In parallel simulations, the numerical domain is divided into n_p sub-domains, each containing an equal number of grid points, where n_p is the number of processors used. MIMICA offers the possibility to either divide the domain along the single x direction (*DECOMP_2D* set to *FALSE* in *start*), or to divide it along both x and y directions (*DECOMP_2D* set to *TRUE*). In the first case, each sub-domain consists in a slab of the whole domain with dimensions $n_x/n_p \times n_y \times n_z$. In the second case, each sub-domain has dimensions $n_x/n_{p,x} \times n_y/n_{p,y} \times n_z$. Both $n_{p,x}$ and $n_{p,y}$ must be explicitly defined in *start* via parameters *NPX* and *NPY* respectively. In the 1D decomposition case, $n_p = NPX \times NPY$.

§When setting-up a new case, one must of course be careful and make sure that n_x is exactly divisible by $n_{p,x}$ (i.e. *MAXX* by *NPX*) and that n_y is exactly divisible by $n_{p,y}$ (i.e. *MAXY* by *NPY*).

3.3.4 Momentum advection

The momentum advection scheme must provide sufficient accuracy and stability when coupled with the time integration scheme. With no additional constraint, this leads to a very limited choice, mainly based on the easiness of implementation. We chose here to discretize momentum advection using high-order finite differences of 3rd (upwind biased) or 4th (central) order. Switching from 3rd to 4th order can be done via parameter *mom_ord* in *cm.nml*. For more detail regarding high-order conservative finite-difference discretizations, interested readers can refer to [22].

For convenience, the advective fluxes are recast into conservative form:

$$\nabla \cdot (\rho_0 \mathbf{u} u) \equiv \frac{F_{i+1/2,k} - F_{i-1/2,k}}{\Delta x} + \frac{F_{i,k+1/2} - F_{i,k-1/2}}{\Delta z_w}, \tag{17}$$

where $F_{i+1/2,k}$, $F_{i-1/2,k}$, $F_{i,k+1/2}$ and $F_{i,k-1/2}$ are the fluxes interpolated at the center of the right, left, upper and lower cell faces on the staggered grid. For simplicity, we have here considered only a 2D system where the subscript i is used in the x direction and k in the z direction. This formulation is by definition energy and mass conserving. The main issue consists now in evaluating the interpolated fluxes F. If we consider polynomial interpolations at the cell centers, the interpolated flux at the center of the right face can be approximated by:

$$F_{i+1/2,j} = \sum_{l=0}^{n-1} C_{a,l} f\left(u_{i-a+l,j}\right), \tag{18}$$

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:

$$F_{i+1/2,j} = \frac{\rho_0}{2}\left(u_{i+1,j} + u_{i,j}\right)\left[-\frac{1}{12}u_{i-1,j} + \frac{7}{12}u_{i,j} + \frac{7}{12}u_{i+1,j} - \frac{1}{12}u_{i+2,j}\right]. \tag{19}$$

Similar expressions can be obtained for all the fluxes necessary to fully discretise the advection operator $\nabla \cdot (\rho_0 \mathbf{u} u)$. A similar flux formulation can be obtained easily for the 3rd order, upwind biased discretization.

In practice, in the absence of numerical diffusion or hyperviscosity, the 3rd order scheme (*mom_ord = 3*) should be preferred as it is more stable and less likely to develop numerical oscillations.

3.3.5 Scalar advection and time-stepping

Regarding scalar advection, additional constraints must be considered in order to make sure that the advected quantities always remain within prescribed physical bounds and do not develop spurious numerical instabilities. 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, generally develop numerical oscillations in the presence of discontinuities. Consequently, specific advective schemes must be employed for scalars to avoid the development of instabilities.

To satisfy these conditions, it is generally not enough to only optimize the spatial discretization scheme. Since we are dealing with time-dependent problems, the fully-discretized equations must be considered and the choice of an appropriate time-stepping method also influences the stability of the solution. Two approaches are available in MIMICA to provide stable solutions to the time-dependent scalar advection problem: the first one is based on finite-volume formulations, while the second one relies on the so-called method of lines that allows the use of any time-stepping method provided that this latter doesn't yield unstable solutions. An extensive description of both approaches can be found in [19].

Finite volume formulation: In finite-volume methods, the continuous, space-dependent scalar function $\phi(\mathbf{x})$ is replaced by the discontinuous scalar field Φ where Φ is defined as an average over small volumes (the grid cells). This can be formally written as [19] (in one-dimension for simplicity):

$$\Phi_i = \frac{1}{V_i} \int_{\mathcal{V}} \phi(x)\, dx, \tag{20}$$

where $\mathcal{V}$ is the volume over which the integral is performed, i.e. the grid cell with spatial index i, and V_i is its volume. Applying the volume integral to the scalar advection equation $\partial \rho_0 \phi / \partial t + \partial (\rho_0 u \phi) / \partial x = 0$ and further integrating in time between t^n and $t^{n+1} = t^n + \Delta t$, yields:

$$\int_{\mathcal{V}} \phi^{n+1}(x)\, dx - \int_{\mathcal{V}} \phi^n(x)\, dx = \int_{t^n}^{t^{n+1}} f^-(\phi^n)\, dt - \int_{t^n}^{t^{n+1}} f^+(\phi^n)\, dt, \tag{21}$$

with the notations f^- and f^+ denoting the fluxes of ϕ across the grid cell boundaries (outgoing and incoming fluxes respectively). This suggest that the advection equation may be replaced by the following finite-volume approximation:

$$\Phi_i^{n+1} = \Phi_i^n - \frac{\Delta t}{\Delta x} \left(F_{i+1/2}^n - F_{i-1/2}^n \right), \tag{22}$$

with $F_{i+1/2}$ and $F_{i-1/2}$ now denoting the fluxes of ϕ across the grid cell boundaries, integrated over time. The main issue associated with finite-volume schemes therefore lies in providing estimates for the incoming and outgoing fluxes. This equation forms the basis of all finite-volume schemes implemented in MIMICA.

Among the wide variety of high-order finite-volume methods available to estimate these fluxes, three common approaches have been implemented in MIMICA and can be selected with the *scal_adv* parameter in *cm.nml*:

lw enables the (at most 2nd order) flux-limited Lax-Wendroff scheme [5]. The default flux limiter there is the Van Leer limiter.

muscl enables the (at most 2nd order) MUSCL finite volume scheme (piecewise linear, [38]). The default flux limiter there is the MC limiter.

quick enables the (at most 3rd order) QUICK finite volume scheme (piecewise parabolic, [18]).

All these schemes rely on the use of slope limiters to avoid the development of unwanted numerical instabilities in the vicinity of discontinuities. In the following, only the default MUSCL scheme is introduced (the other schemes have a similar design).

The MUSCL scheme [38] is a central method that generates minor numerical dissipation in smooth areas. Let's first rewrite the flux swept through a cell face ($F_{i+1/2}$ at $x_{i+1/2}$) during a time step as:

$$F_{i+1/2} = \int_{x_{i+1/2}-u_{i+1/2}\Delta t}^{x_{i+1/2}} \phi\left(x\right) dx. \tag{23}$$

To estimate the integral, we then need to approximate the continuous function ϕ based on the model solution Φ. Popular approximations rely on polynomial interpolations: a polynomial of order 0 means that ϕ is constant within each grid cell (1st order), a polynomial of order 1 means that ϕ is linear within each grid cell (2nd order) and a polynomial of order 2 means that ϕ is quadratic within each grid cell (3rd order). MUSCL schemes as described here make use of piecewise linear interpolations and are therefore (at most) 2nd order accurate).

Using a piecewise linear interpolant between $x_{i-1/2}$ and $x_{i+1/2}$ gives:

$$\widetilde{\phi}\left(x\right) = \phi\left(x_0\right) + s_i\left(x - x_0\right), \tag{24}$$

with s_i the slope and x_0 the location of the grid cell center. Integrating over the grid cell then gives $\phi\left(x_0\right) = \Phi_i$ and:

$$s_i = \frac{\Phi_{i+1} - \Phi_i}{\Delta x}. \tag{25}$$

Integral 23 can now be evaluated exactly:

$$F_{i+1/2} = u_{i+1/2}\left[\Phi_i + \frac{1}{2}\left(1 - |\mu|\right)\left(\Phi_{i+1} - \Phi_i\right)\right] \tag{26}$$

where we have introduced the Courant number $\mu = u_{i+1/2}\Delta t/\Delta x$. Note that on the Arakawa C-grid, because u is defined at face centers, the advective velocity $u_{i+1/2}$ is directly available and does not require any interpolation.

As mentioned previously, scalar advection often requires the use of so-called limiters to prevent unphysical values to be produced as a result of numerical errors (*limit = .true.* in *cm.nml*, default behavior). We thus rewrite equation 26 as (see for example [5]):

$$F_{i+1/2} = u_{i+1/2}\left[\Phi_i + \frac{1}{2}C_{i+1/2}\left(1 - |\mu|\right)\left(\Phi_i - \Phi_{i-1}\right)\right] \tag{27}$$

where we have now introduced the flux limiter $C_{i+1/2}$. $C_{i+1/2}$ si defined as a function of the slope ratio $r_i = \left(\Phi_{i+1} - \Phi_i\right)/\left(\Phi_i - \Phi_{i-1}\right)$. For example, the default MC limiter reads [37]:

$$C_{i+1/2} = \max\left[\min\left[2r_i, \frac{1}{2}\left(r_i + 1\right), 2\right], 0\right]. \tag{28}$$

The role of the flux limiter is to selectively switch between the low-order, upwind biased formulation (very diffusive to prevent the development of numerical oscillations) in the presence

of strong gradients, and the 2nd order central scheme (more dispersive and prone to produce wiggles) where gradients are smooth enough. For computational efficiency, limitation is not applied everywhere in the domain, but only in regions where $|\Phi_{i+1} - \Phi_i| > \lambda |\Phi_i|$, with λ set by default to 10^{-9} (this can be changed through the parameter *limit_tol* in *cm.nml*, larger values of the tolerance leading to less overall limitation).

In the situation where $u_{i+1/2} < 0$, the flow comes from the right side of the cell and the upwinding must be reversed:

$$F^-_{i+1/2} = \frac{1}{\Delta t} u_{i+1/2} \left[\Phi_{i+1} - \frac{1}{2} C^-_{i+1/2} (1 - |\mu|) (\Phi_{i+2} - \Phi_{i+1}) \right], \tag{29}$$

and $C^-_{i+1/2}$ is now a function of $r_{i+1} = (\Phi_{i+1} - \Phi_i) / (\Phi_{i+2} - \Phi_{i+1})$. By denoting $F^+_{i+1/2}$ the flux defined by equation 27, we can finally write, for a velocity of arbitrary sign:

$$F_{i+1/2} = 0.5 \left[1 + sign\left(1, u_{i+1/2}\right) \right] F^+_{i+1/2} + 0.5 \left[1 - sign\left(1, u_{i+1/2}\right) \right] F^+_{i+1/2}. \tag{30}$$

The function $sign\left(1, u_{i+1/2}\right)$ has the sign of $u_{i+1/2}$ and an absolute value equal to 1.

Method of lines: In contrast to finite volume methods where the integral formulation 23 naturally leads to a discrete equation in both time and space, the method of lines allows the use of distinct distretization schemes for the time derivative and advection term [19]. This provides more flexibility as different combinations of high-order discretization schemes for these two terms can be employed. We must however be careful in doing so as the basic requirements for the global scheme to remain stable and not produce over- or under-shoots (especially in presence of discontinuities) limits the choice of possible combinations.

To discretize the advection term, some of the methods proposed above (MUSCL and QUICK) can also be employed. The main difference between the finite-volume method and method of lines concerns the streaming terms (involving μ) that stem from the time integrals in 23 and are therefore absent in the method of lines formulation. The equivalent MUSCL fluxes with flux limiters thus become:

$$F_{i+1/2} = u_{i+1/2} \left[\Phi_i + \frac{1}{2} C_{i+1/2} (\Phi_i - \Phi_{i-1}) \right]. \tag{31}$$

To complement the MUSCL (or QUICK) advection scheme, the time derivative should be discretized using a method of (at least) equivalent order of accuracy, and that doesn't generate spurious oscillations or spurious extrema. Given these conditions, a 2nd order Runge-Kutta method (Heun's method) appears as a good choice [12]. This scheme can be viewed as a predictor corrector scheme consisting in two stages:

$$\Phi^* = \Psi^n + \Delta t \mathcal{F}(\Psi^n) \tag{32}$$

$$\Phi^{n+1} = \Psi^n + \frac{1}{2} \Delta t \left[\mathcal{F}(\Psi^n) + \mathcal{F}(\Psi^*) \right], \tag{33}$$

with $\mathcal{F}$ denoting the discrete estimates of the advective terms.

Time-step limitations: The stability of the fully discrete equations is generally assessed in terms of the Courant-Friedrichs-Lewy (CFL) criterion that constrains $\mu = |u| \Delta t / \Delta x$ to remain below a threshold value. For both the method of lines and finite-volume method, stability is generally

ensured when $\mu \approx 0.5$. Threshold values larger than 0.5 are possible, but a precise estimate of the maximum allowed value is hard to determine in practice due to the complexity of the problem involving 3 spatial dimensions, flux limiters, and other physical tendencies.

In MIMICA, setting *ldtfix = 0* in *cm.nml* allows the effective time-step Δt to be computed dynamically so that the maximum CFL number in the numerical domain, μ_{max}, always remains within prescribed bounds: $cfl\_min < \mu_{max} < cfl\_max$, with *cfl_max* and *cfl_min* defined in *cm.nml*. In 2 or 3 dimensions, the maximum CFL number is calculated as:

$$\mu_{max} = \sup\left(\Delta t \max\left(\frac{|u_i|}{\Delta x_i}\right)\right) \tag{34}$$

where sup corresponds to the absolute maximum within the entire domain, while max denotes a local maximum over all 3 dimensions ($i = \{1, 2, 3\}$).

In the fully compressible case, the acoustic waves propagating at a speed $c = \sqrt{\gamma p/\rho}$ must be resolved, giving $\mu = (|u| + c)\,\Delta t/\Delta x$. Since c is generally an order of magnitude larger than $|u|$ in most atmospheric applications, a compressible solver will typically require time steps that are at least 10 times smaller than with the anelastic solver.

3.4 Physical parameterizations

3.4.1 Subgrid scale turbulence

Turbulence models are used to evaluate the subgrid scale (SGS) turbulent fluxes of all transported quantities, and are denoted τ_Ψ. Typically, a simple gradient hypothesis is used to model these terms, by analogy with molecular diffusion [36]:

$$\tau_{\Psi,i} = -K_h \frac{\partial \Psi}{\partial x_i} \tag{35}$$

for scalars and

$$\tau_{u,i,j} = -K_m \left(rac\partial u_i \partial x_j + \frac{\partial u_j}{\partial x_i}\right) \tag{36}$$

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

The first approach, selected by setting *setenv TKE TRUE* in *start*, is a 1st order closure based on the solution of a turbulent kinetic energy (TKE) equation [36, 21]. A closed form of the SGS TKE equation can be written:

$$\frac{\partial e}{\partial t} + u_i \frac{\partial e}{\partial x_i} = 2K_m S_{ij} S_{ij} + 2\frac{\partial}{\partial x_i}\left(K_m \frac{\partial e}{\partial x_i}\right) - K_h \frac{g}{\theta_v}\frac{\partial \theta_v}{\partial z} - C_\epsilon \frac{e^{3/2}}{l}. \tag{37}$$

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, and l a characteristic turbulent length scale. The characteristic length scale can be defined as (in 3D):

$$l = \min\left(l_\Delta, 0.76\sqrt{\frac{e}{\mathcal{N}^2}}, 0.5 C_l \Delta z\,(1)\right), \tag{38}$$

with the characteristic cutoff scale given by:

$$l_\Delta = (\Delta x \Delta y \Delta z)^{1/3}. \tag{39}$$

z is the altitude, κ the Von Karman constant (≈ 0.41),C_l a constant of the model (given below), and $\Delta z\,(1)$ the size of the first model level. Using this formulation for l, turbulent mixing is limited close to the surface and in very stable conditions where turbulence is inhibited and the size of the turbulent eddies is small. The eddy diffusivity K_m is finally obtained using simple scaling arguments:

$$K_m = C_m l \sqrt{e}. \tag{40}$$

The constants C_l, C_ϵ and C_m are set to 0.845, 0.845 and 0.0856 respectively. It is then usually assumed that $K_h = K_m/Pr$, with Pr the turbulent Prandtl number. Pr can be defined in *cm.nml* (*pran*) and a recommended value is ~ 0.4. Note that if *pran* is negative, turbulent mixing is only applied up to *zdec* (to be defined in *cm.nml*) with Pr =*pran*.

The second closure is based on the 0th order Smagorinsky and Lilly approach [34] (*setenv TKE FALSE* in *start*). If we assume a balance between production of TKE by shear plus buoyancy and dissipation in equation 37, it follows that:

$$2K_m S_{ij} S_{ij} - K_h \mathcal{N}^2 - C_\epsilon \frac{e^{3/2}}{l} = 0. \tag{41}$$

Rearranging the terms yields:

$$K_m = (C_S l)^2 \sqrt{2 S_{ij} S_{ij}} \sqrt{1 - \frac{Ri}{Pr}} \tag{42}$$

with $C_S = 0.18$. We have here introduced the Richardson number $Ri = \mathcal{N}^2/|\mathbf{S}|$ to account for convective stability. In this case, the Brunt-Vaisala frequency $\mathcal{N}^2$ is modeled following [35] to include changes in stability related to condensation/evaporation at the SGS level, and:

$$K_h \mathcal{N}^2 = -\frac{g}{\theta_{00}} \left\langle w' \theta_v' \right\rangle \approx K_h \frac{\partial b}{\partial z}. \tag{43}$$

Note that applying turbulent diffusion to microphysical quantities (mass and number concentrations) as well as to aerosol concentrations is optional. This is controlled by parameter *micro_dif* in *cm.nml*: if it is set to *false*, these quantities are not affected by turbulent diffusion (default is *true*). Turbulent diffusion can also be switched off altogether by setting *with_dif* to *false* in *cm.nml* (this also turns off fluxes at the surface).

3.4.2 Surface conditions

The surface in MIMICA is treated as a solid wall upon which the vertical velocity vanishes. The first and most basic condition enforced at the surface is thus:

$$w\,(x, y, z = 0) = 0. \tag{44}$$

Next, exchanges of energy, moisture and momentum (this latter can be optionally turned off by setting *momsf = 0* in *cm.nml*) between the surface and the atmosphere due to turbulent mixing need to be parameterized. The method adopted can be selected via *isurf* in *cm.nml*. *isurf* is an integer which can take any value between 0 and 4:

0 Prescribed surface fluxes

1 Prescribed skin surface temperature (SST)

2 Prescribed SST with fixed drag coefficient

3 Similar to 2 but with u^* defined as in [13]

4 Fixed SST, with SST computed interactively to result in prescribed average surface fluxes.

Only cases 0, 1 and 2 are described below.

Options 0 and 1 rely on Monin-Obukhov similarity theory according to which momentum and virtual potential temperature fluxes at the ground are proportional to the friction velocity u^* [36]:

$$u^*u^* = \overline{u'w'} = \overline{v'w'}. \tag{45}$$

$$u^*\theta_v^* = \overline{\theta_v' w'}. \tag{46}$$

In the simple case where surface sensible and latent heat fluxes are prescribed ($isurf = 0$), there is no need to estimate θ_v^* as $\overline{\theta_v' w'}$ is readily given, but u^* still needs to be computed to calculate the surface momentum flux. Although u^* can be set as a constant in *cm.nml* (variable *ust*), it must generally be computed interactively.

The turbulent surface fluxes are now rewritten using the usual gradient hypothesis:

$$\overline{u'w'} = -K_m \frac{\partial u}{\partial z}, \tag{47}$$

with K_m approximated by:

$$K_m = \kappa z u^*, \tag{48}$$

where κ is the von Karman constant (≈ 0.41) and z is the altitude of the first velocity grid point. Combining these relations and integrating over z yields the classical logarithmic shape of the velocity profile inside the surface layer:

$$ln\left(\frac{z}{z_0}\right) = \frac{\kappa\,|u|}{u^*}, \tag{49}$$

from which u_* can be deduced. Here, z_0 is a surface roughness height depending on the ground nature. z_0 is set by default to $zrough = 1.e-4\ m$ in *cm.nml*, and must be adjusted depending on the simulated surface type. Relation 49 remains valid as long as the surface layer remains neutral. For stable and unstable BL, Monin-Obukhov similarity theory is used to derive relations similar to 49. This first requires the evaluation of a surface buoyancy flux which can take different forms depending on the selected option.

***isurf* = 0: Fixed surface fluxes** The surface buoyancy flux is defined as:

$$\overline{b'w'} = \frac{g}{\theta_{00}}\overline{\theta_v' w'} = \frac{g}{\rho_0 \theta_{00}}\left(\frac{SHF}{cp} + \epsilon\theta_{00}\frac{LHF}{L_v}\right), \tag{50}$$

Where SHF and LHF must be prescribed in *cm.nml* (*shf0* and *lhf0* respectively). We then define the Monin-Obukhov length scale by:

$$L = -\frac{u^{*3}}{\kappa \overline{b'w'}}, \tag{51}$$

and deduce the friction velocity from a relation similar to 49:

$$ln\left(\frac{z}{z_0}\right) - \Psi_m(\zeta) = \frac{\kappa\,|u|}{u^*} \tag{52}$$

in which Ψ_m (the stability function) has been introduced to account for stability/instability at the surface. The ratio $\zeta = z/L$ can be viewed as a measure of surface layer stability (stable for $\zeta > 0$ and unstable for $\zeta < 0$). Ψ_m is commonly obtained from observations and depends on the sign of ζ. We use here the forms proposed by Garratt [11]:

$$\text{for } \zeta > 0 \qquad \Psi_m(\zeta) = -5.3\zeta \tag{53}$$

$$\text{for } \zeta < 0 \qquad \Psi_m(\zeta) = 2ln\left(\frac{1+x}{2}\right) + ln\left(\frac{1+x^2}{2}\right) - 2tan^{-1}(x) + \frac{\pi}{2} \qquad \text{and} \qquad x = (1-16\zeta)^{1/4} \tag{54}$$

As this was the case for the neutral boundary layer, u^* can be derived from relation 52. However, L depending directly on u^*, an iterative procedure is required to obtain a converged value of u^*. The friction virtual potential temperature can then be deduced from: $\theta_v^* = -\frac{\theta_{00} b}{g u^*}$. Of course, the applied sensible and latent heat fluxes are directly given by the prescribed SHF and LHF, and do not need to be recomputed.

isurf **= 1: Fixed surface properties** The surface buoyancy flux is now calculated from the gradient between the virtual potential temperature in the first grid cell above the surface and a skin surface virtual potential temperature:

$$b = \frac{g}{\theta_{00}}\left(\theta_{v1} - \theta_{v,skin}\right) = \frac{g}{\theta_{00}}\left[\left(\theta_1 - \Pi_{00} SST\right) + 0.608\theta_{00}\left(q_{t1} - SSM\right)\right]. \tag{55}$$

SST and SSM are the prescribed skin surface temperature and surface moisture (where it is generally assumed that the near surface air is saturated) which can be prescribed in *cm.nml*. An initial friction potential temperature can readily be obtained under neutral conditions using:

$$Pr_t ln\left(\frac{z}{z_0}\right) = \frac{b\theta_{00}}{g\theta_{v*}^*}, \tag{56}$$

with Pr_t the turbulent Prandtl number defined in *cm.nml*. Under stable/unstable conditions, we can write:

$$Pr_t\left[ln\left(\frac{z}{z_0}\right) - \Psi_h(\zeta)\right] = \frac{\theta_{00} b}{g\theta_v^*}. \tag{57}$$

The friction velocity is still estimated following 52. Ψ_m doesn't change compared to the fixed surface fluxes case, and Ψ_h is given by:

$$\text{for } \zeta > 0 \qquad \Psi_h(\zeta) = -5.3\zeta \tag{58}$$

$$\text{for } \zeta < 0 \qquad \Psi_h(\zeta) = 2ln\left(\frac{1+y}{2}\right) \qquad \text{and} \qquad y = (1-16\zeta)^{1/4}. \tag{59}$$

with $\zeta = z/L$ and:

$$L = \frac{u^{*2}\theta_{00}}{\kappa g \theta_v^*}. \tag{60}$$

Once again, the dependency of L on u^* and θ_v^* requires subiterations. Once a converged value of θ_v^* is obtained, it can be used to determine the surface latent and sensible heat fluxes.

isurf **= 2: Fixed surface drag** The option *isurf* = 2 implies a very simple surface flux formulation whereby friction quantities at the surface are obtained using:

$$u^* = C_{d,m}\left|U\right|, \tag{61}$$

and:

$$\theta_v^* = \frac{C_{d,s}}{Pr_t} \left(\theta_{v1} - \theta_{vs}\right). \tag{62}$$

θ_{vs} is the surface value of the virtual potential temperature calculated using a fixed SST (prescribed in *cm.nml*) and assuming near surface air is saturated. The drag coefficients $C_{d,m}$ and $C_{d,s}$ can be prescribed in *cm.nml*. Their default values are 1.2×10^{-3} and 1.1×10^{-3} respectively. Again, sensible and latent heat fluxes can be recomputed from θ_v^* and u^*. $|U|$ is the magnitude of the surface winds. This latter can be limited to a minimum value equal to *min_w* in *cm.nml* in order to guarantee that surface fluxes are non-zero even under quite conditions.

3.4.3 Radiation

The radiation solver coupled to MIMICA is a version of the Fu-Liou-Gu model [9, 10, 14]. The solver is a multiband, 4-stream radiation model that requires multiple input files and options to operate. Running MIMICA with interactive radiation is done by setting *setenv RADIA TRUE* in *start*. Because the radiative transfer model is the slowest part of all the MIMICA model, radiation is only calculated every *iradx* seconds (can be modified in *cm.nml*). Note that short-wave radiation can be turned off setting *rad_sw = 0* in *cm.nml*.

The radiation solver works on 1D columns extending up to the top of the atmosphere. Because MIMICA's domain is limited in altitude, the input data must be completed by a standard atmosphere above the numerical domain. Standard soundings are stored in *.lay* files located in the *DATA* directory. The most appropriate standard atmosphere must then be renamed *sounding_rad.dat* to be read by MIMICA. 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 these 1D soundings, surface parameters must also be provided as inputs to the radiation solver. These parameters are: the surface albedo (*alb* in *cm.nml*, set by default to 0.07), the surface skin temperature (*sst* in *cm.nml*, set by default to 295 K), and the surface emissivity (*emi* in *cm.nml*, set by default to 0.984). To finish with, the solar zenith angle is computed at each radiation step. The formula uses the prescribed latitude, julian day, starting time of the simulation (UTC) and the local simulation time. All these parameters can also be set in *cm.nml*.

Note that in some special cases, for example in the DYCOMS and ISDAC templates, radiation is computed as a simple function of the liquid water path only, and the full radiative transfer model is not needed.

3.4.4 Microphysics

Two bulk microphysics schemes are readily available in MIMICA. The more advanced and complete choice is given by the Seifert-Beheng two-moment microphysics model [30, 31] which can be selected by setting *SEIFERT TRUE* in *start*. When *SEIFERT* is *FALSE*, a simpler and more computationally effective one-moment scheme developed by [13] is used.

The Seifert-Beheng (SB) scheme: In the available SB scheme, at most 6 classes of hydrometeors are defined: cloud droplets, rain drops, ice crystals, graupel, snow particles and hail stones. The *lmicro* keyword in *cm.nml* controls the number of hydrometeors modeled: *lmicro = 0* deactivates microphysics altogether (a similar effect is obtained by setting *with_micro* to *false*), *lmicro = 1* considers warm cloud microphysics only (cloud droplets and rain drops), *lmicro = 2* adds a simple ice category, *lmicro = 3* also includes graupel and snow particles, while with *lmicro = 4*,

all hydrometeors are considered, including hail stones. However, in the following, microphysics level up to 3 only are described: The implemented hail microphysics is still experimental.

Each hydrometeor category is characterized by a particle **mass** distribution $f(m)$ assumed to take the form of a generalized gamma distribution:

$$f(m) = A_0 m^{\nu} \exp(-\lambda m^{\mu}). \tag{63}$$

While both exponents ν and μ are typically held constant for each hydrometeor type (see table 1), A_0 and λ must be computed to fully determine the hydrometeors' size distributions. This can be achieved by relating these two parameters to known moments of the gamma distribution, typically M^0 and M^1. This yields:

$$N = M^0 = \int_0^{\infty} f(m)\, dm = \frac{A_0 \Gamma\left(\frac{\nu+1}{\mu}\right)}{\mu \lambda^{\frac{\nu+1}{\mu}}}, \tag{64}$$

and:

$$q = M^1 = \int_0^{\infty} m f(m)\, dm = \frac{A_0 \Gamma\left(\frac{\nu+2}{\mu}\right)}{\mu \lambda^{\frac{\nu+2}{\mu}}}. \tag{65}$$

Combining these two equations, one can estimate A_0 and λ as functions of N and q:

$$\lambda = \left[\frac{\Gamma\left(\frac{\nu+1}{\mu}\right)}{\Gamma\left(\frac{\nu+2}{\mu}\right)} \frac{q}{N}\right]^{-\mu} \tag{66}$$

and:

$$A_0 = \frac{\mu N}{\Gamma\left(\frac{\nu+1}{\mu}\right)} \lambda^{\frac{\nu+1}{\mu}}. \tag{67}$$

From the full knowledge of all particles' mass distributions, all microphysical processes (phase changes, collision-coalescence, riming...) can be calculated.

The number and mass concentrations of each hydrometeor category are obtained by solving the following equations:

$$\frac{\partial \rho_0 N_k}{\partial t} + \nabla \cdot (\rho_0 \mathbf{u} N_k) = \mathcal{Q}_{N,k} + \nabla \cdot (\rho_0 \tau_{N,k}) + S_N \tag{68}$$

$$\frac{\partial \rho_0 q_k}{\partial t} + \nabla \cdot (\rho_0 \mathbf{u} q_k) = \mathcal{Q}_{q,k} + \nabla \cdot (\rho_0 \tau_{q,k}) + S_q \tag{69}$$

with N_k and q_k the number and mass concentrations of particle k, $\mathcal{Q}$ the sum of all microphysical contributions including collision-coalescence, phase changes and precipitation, τ the subgrid scale turbulent mixing terms (optional) and S possible additional sources.

In general, the rates of change due to collisions are treated by integrating the particles' distributions over a collection kernel K. For instance, the collision between particles j and k results in:

$$\left.\frac{\partial N_j}{\partial t}\right|_{jk} = \int_0^{+\infty} f_j(m_j) \left[\int_0^{+\infty} f_k(m_k^*)\, K(m_j, m_k^*)\, dm_k^*\right] dm_j, \tag{70}$$

and:

$$\left.\frac{\partial q_j}{\partial t}\right|_{jk} = \int_0^{+\infty} m_j f_j\left(m_j\right)\left[\int_0^{+\infty} f_k\left(m_k^*\right) K\left(m_j, m_k^*\right) dm_k^*\right] dm_j. \tag{71}$$

For warm microphysics (involving liquid water only), a simple kernel defined as a picewise polynomial (Long's kernel) is used:

$$K\left(m, m^*\right) = k_c\left(m^2 + m^{*2}\right) \tag{72}$$

$$K\left(m, m^*\right) = k_r\left(m + m^*\right). \tag{73}$$

The first polynomial is used when the mass of the collector m is smaller than a critical mass m_c separating cloud drops and rain drops, and set to the mass of a 40 μm radius droplet (m_c is known as the auto-conversion threshold). The second one is used when the collector's mass is greater than m_c. The constants k_c and k_r are given the values $9.44e9$ m3/kg2/s and 5.78 m3/kg/s respectively.

Integrating mass distributions using the above kernels leads to the exact expressions for the rates of auto-conversion (c+c $\rightarrow$ r), accretion (r+c $\rightarrow$ r) and selfcollection (c+c $\rightarrow$ c or r+r $\rightarrow$ r):

$$\left.\frac{\partial q_r}{\partial t}\right|_{auto} = \frac{k_c}{20 m_c}\frac{\left(\nu_c+2\right)\left(\nu_c+4\right)}{\left(\nu_c+1\right)\left(\nu_c+1\right)} q_c^2 \overline{m}_c\left[1 + \frac{\Phi_a\left(\tau\right)}{\left(1-\tau\right)^2}\right], \tag{74}$$

$$\left.\frac{\partial N_r}{\partial t}\right|_{auto} = \frac{1}{m_c}\left.\frac{\partial q_r}{\partial t}\right|_{auto} \tag{75}$$

$$\left.\frac{\partial q_c}{\partial t}\right|_{auto} = -\left.\frac{\partial q_r}{\partial t}\right|_{auto} \tag{76}$$

$$\left.\frac{\partial N_c}{\partial t}\right|_{auto} = -2\left.\frac{\partial N_r}{\partial t}\right|_{auto} \tag{77}$$

$$\left.\frac{\partial q_r}{\partial t}\right|_{acc} = k_r q_c q_r \Phi_{ac}\left(\tau\right) \tag{78}$$

$$\left.\frac{\partial N_r}{\partial t}\right|_{acc} = 0 \tag{79}$$

$$\left.\frac{\partial q_c}{\partial t}\right|_{acc} = -\left.\frac{\partial q_r}{\partial t}\right|_{acc} \tag{80}$$

$$\left.\frac{\partial N_c}{\partial t}\right|_{acc} = -\frac{1}{\overline{m}_c}\left.\frac{\partial q_r}{\partial t}\right|_{acc} \tag{81}$$

$$\left.\frac{\partial q_r}{\partial t}\right|_{sc} = 0 \tag{82}$$

$$\left.\frac{\partial N_r}{\partial t}\right|_{sc} = -k_r N_r q_r \Phi_b\left(\tau\right) \tag{83}$$

$$\left.\frac{\partial q_c}{\partial t}\right|_{sc} = 0 \tag{84}$$

$$\left.\frac{\partial N_c}{\partial t}\right|_{sc} = -k_c \frac{(\nu_c+2)}{(\nu_c+1)} q_c^2 + 2\frac{1}{m_c}\left.\frac{\partial q_r}{\partial t}\right|_{auto} \tag{85}$$

where $\nu_c = 1$ corresponds to the mass exponent in the mass distribution for cloud droplets. 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, Φ_{ac} and Φ_b are autoconversion, accretion and break-up rates defined by Seifert and Beheng [31].

For cold microphysics (involving ice particles), collisions are treated using the gravitational kernel:

$$K\left(D, D^*\right) = \rho E^* \frac{\pi}{4}\left(D + D^*\right)^2 \left|V_p - V_p^*\right|. \tag{86}$$

E^* is an efficiency coefficient, which can be defined as the product of a collision efficiency and a coalescence or sticking efficiency. While the traditional gravitational kernel is a function of the particles' size, a relationship between m and D must be provided to be able to integrate equations 70 and 71. In addition, the precipitation velocities V_p must also be computed as functions of m. Both D and V_p are usually defined as power law functions of the mass:

$$D = a_m m^{b_m} \tag{87}$$

$$V_p = a_v m^{b_v} \tag{88}$$

which simplifies greatly the computation of collision integrals. Default values for parameters a_m, b_m, a_v and b_v are provided in table 1. Note that alternative ice crystal types are available in MIMICA including dendrites, columns, plates or rosettes, and a different set of parameter values must be used for each different habit. Only one type of crystal can be used at a time, and this can be selected via the *ice_habit* parameter in *cm.nml* (*ice_habit* is a string of characters of length 3, equal to DEN, COL, PLA, BUL for the four crystal types mentioned previously).

In the case where a given particle k (with k standing for rain, ice, graupel, snow or hail) collides with cloud droplets, the problem can be further simplified by assuming that cloud droplets do not precipitate ($V_p = 0$) and $D_{drop} \ll D_k$. Special collision processes such as partial riming (conversion of ice crystals to graupel when they collide with small cloud droplets) or ice multiplication by the Hallett-Mossop process are also described in more details in [31]. All interactions between hydrometeors thus parameterized in MIMICA are listed in table **??**.

Following Pruppacher and Klett [26], the mass tendency of a given droplet of diameter D due to evaporation/condensation is given by:

$$\left.\frac{\partial m}{\partial t}\right|_{e/c} = 2\pi D\left(T, p\right) G F_v \delta, \tag{89}$$

with $\delta = q_v/q_s - 1$ is the relative supersaturation and:

$$G = \left[\frac{R_w T}{e_s D_v} + \frac{L_{lv}}{K_T T}\left(\frac{L_{lv}}{R_w T} - 1\right)\right]^{-1} \tag{90}$$

e_s being 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 ($\approx 3 \times \times 10^{-5}$ m^2/s), K_T the heat conductivity ($\approx 2.5 \times 10^{-2}$ Jm/s/K). The ventilation factor F_v is defined as a function of the Best number $X = Sc^{1/3} Re^{1/2}$, with $F_v = a_v + b_v X\left(m\right)^{\gamma}$ and $Re = \rho_0 V_p D/\mu$. The Schmidt

number Sc is assumed to be constant and equal to 0.7, while V_p is the drop terminal fall speed and μ is the dynamic viscosity of air ($\approx 1.5 \times 10^{-5}$ kg/m/s). All parameters appearing in F_v are assumed to be independent of hydrometeor type and $a_v = 0.78$, $b_v = 0.308$ and $\gamma = 1$. It is possible to switch on/off ventilation by setting *lvent* to *true/false* in *cm.nml* (i.e $F_v = 1$ if *lvent* = *.false.*).

To obtain condensation/evaporation sources for the whole drop population, we then have to integer relation 89 over the assumed mass distribution. The final total mass tendency takes the form:

$$\left.\frac{\partial q}{\partial t}\right|_{e/c} = 2\pi N G\left(T,p\right)\left\langle D\right\rangle F_v^* \frac{s}{q_s} \tag{91}$$

where we defined an averaged ventilation factor: $F_v^* = a_v^* + b_v^* X \left(\overline{m}\right)^{1/2}$, with $\overline{m} = q/N$, and the modified coefficients a_v^* and b_v^* are given in [31]. Evaporation also decreases the number concentration of the considered drop population, and the corresponding rate of change is expressed as:

$$\left.\frac{\partial N}{\partial t}\right|_{e/s} = \frac{N}{q}\min\left(\left.\frac{\partial q}{\partial t}\right|_{e/s}, 0\right). \tag{92}$$

This reduces to assuming that the mean particle mass is conserved during evaporation. Similar relations are found for sublimation/deposition processes applied to ice crystals, graupel and snow flakes. Note that evaporation of cloud droplets is treated as in [23]: N_c is held constant during droplet evaporation, but they all evaporate instantaneously when q_c drops below a tiny threshold value.

Note that condensation/evaporation of cloud droplets can alternatively be paramterized using saturation adjustment (*setenv SAT_ADJ TRUE* in *start*). Essentially, saturation adjustment assumes that condensation/evaporation over cloud droplets is an infinitely fast process. In other words, in a cloudy grid box, all the vapor in excess of saturation condenses extremely fast onto cloud particles until saturation is reached, and cloud particles will evaporate extremely fast in subsaturated conditions until saturation is reached. In practice, saturation adjustment is imposed diagnostically at the end of each time step following the procedure described below (see Grabowski scheme).

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

$$\left.\frac{\partial m}{\partial t}\right|_{melt} = \frac{2\pi}{L_{il}}\left[K_T\left(T - T_0\right)F_h + \frac{D_v}{L_{lv}}\left(\frac{p_v}{T} - \frac{p_s}{T_0}\right)F_v\right]D. \tag{93}$$

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:

$$\left.\frac{\partial Q}{\partial t}\right|_{melt} = \frac{2\pi}{L_{il}} N\left[K_T\left(T - T_0\right)Le_v + \frac{D_v}{L_{lv}}\left(\frac{p_v}{T} - \frac{p_s}{T_0}\right)\right]F_v^*\left\langle D\right\rangle, \tag{94}$$

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

Precipitation is handled by adding a precipitation term integrated over the hydrometeors' mass distribution:

$$\mathcal{Q}_{prec}^{\Psi} = -\frac{\partial \rho_0 \int_0^{+\infty} V_p\left(m\right)\Psi dm}{\partial z}, \tag{95}$$

to the governing equations for q and N (here $\Psi = \{q, N\}$). With the definition of V_p as a function of m given previously, the precipitation integrals appearing above are relatively easy to estimate directly. Once the velocity integrals are known, the $\mathcal{Q}^{\Psi}_{prec}$ terms are estimated implicitly which requires the solution of a tridiagonal system in each model column.

Finally, it should be noted that all hydrometeor mass and number concentrations are appropriately limited at the end of each time-step to avoid negative concentrations. In practice, limitation is applied whenever q and N drop below appropriately prescribed minimum values. The procedure is conservative in the sense that if a mass concentration becomes negative (even slightly), the mass of water added to reach 0 kg/m^3 is subtracted to the other hydrometeor concentrations.

hydrometeor	a_m	b_m	a_v	b_v	ν	μ	A_0 (1-moment only)
cloud (c)	0.124	1/3	-	-	1/3	1	-
rain (r)	0.124	1/3	159	0.266	-2/3	1/3	5e5
ice (i, default)	0.217	0.302	317	0.363	-1/3	1/3	-
graupel (g)	8.156	0.526	27.7	0.216	-2/3	1/3	4.5e6
snow (s)	0.168	0.323	40	0.230	1/3	1/2	1.8e8

Table 1: t Default microphysical parameters for the SB scheme and all kinds of hydrometeors. Units are SI (kg-m-s). Note that ν and μ values reported here are different from the original SB scheme for the parameterized mass distributions to be more consistent with typical size distributions. In particular, ν values were adjusted so that the main precipitating particles follow exponential size distributions.

Single-moment SB scheme: A single-moment version of the SB scheme described above has been implemented in MIMICA to provide a detailed description of all liquid and ice microphysics at a lower computational cost. This model can be selected by setting *SEIFERT* to *TRUE* in *start*, and adding the option *moments = 1* in *cm.nml* (default it 2 for the standard two-moment SB scheme).

The single-moment SB scheme follows exactly the standard two-moment version for the parameterization of all microphysical processes. The difference between one and two moments lies in the computation of the preexponential factor A_0 appearing in the definition of the generalized gamma distribution (equation 63), and of hydrometeor number concentrations N. For precipitating hydrometeors (i.e. rain, graupel and snow), while N is a prognostic variable in the two-moment scheme and A_0 is diagnosed, A_0 is now fixed in the single-moment version and N is diagnosed from q and A_0. Combining equations 66 and67, and letting N being the unknown, we find:

$$\lambda = \left[A_0 \frac{\Gamma\left(\frac{\nu+2}{\mu}\right)}{\mu q} \right]^{\frac{\mu}{\nu+2}} \tag{96}$$

and:

$$N = A_0 \frac{\Gamma\left(\frac{\nu+1}{\mu}\right)}{\mu} \lambda^{-\frac{\nu+1}{\mu}} \tag{97}$$

Note that even in the single-moment version of the SB scheme, cloud droplet and ice crystal number concentrations are obtained from the selected activation and nucleation schemes.

Transport equations for N_c and N_i are not solved in the single-moment scheme, and the activation and nucleation parameterizations are used diagnostically instead of equation 97.

With λ and N diagnosed from A_0 and q, all microphysical processes can be parameterized exactly as done for the standard two-moment scheme, without any further modification.

Single-moment Grabowski scheme: In the single-moment scheme from Grabowski [13] (default model is *SEIFERT* is *FALSE* in *start*), hydrometeors are divided into only two classes: precipitating and non-precipitating. The distinction between liquid and ice particles then solely depends on the modelled temperature. The parameterization therefore requires the addition of only two prognostic microphysical variables: q_c and q_r, the mass mixing ratios of non-precipitating and precipitating particles respectively. The equations governing the evolution of q_c and q_r are similar to equation 69.

Here, only 4 microphysical processes need to be parameterized:

1. auto-conversion: conversion from non-precipitating cloud particles to precipitating particles,
2. accretion: the collection of non-precipitating particles by precipitating particles,
3. phase changes: condensation and evaporation, or deposition and sublimation),
4. precipitation: only concerns precipitating particles.

Precipitating particles are assumed to follow a Marshall-Palmer (exponential) size distribution:

$$f(D) = N_0 \exp(-\lambda D), \tag{98}$$

where N_0 is fixed to $10^7\ m^{-4}$, and λ is diagnosed from q_r:

$$\lambda_r = \left[\frac{a_m N_0 \Gamma(b+1)}{q_r}\right]^{\frac{1}{b_m+1}}, \tag{99}$$

where b_m and a_m are the two parameters of the mass-size relationship, $m = a_m D^{b_m}$ (see table 2). The number concentration of precipitating particles can then be simply obtained using: $N = N_0/\lambda$.

To parameterize auto-conversion (and other processes), the partitioned concentrations of liquid and ice are obtained based on a simple linear function of temperature: $q_l = f_l q_c$ and $q_i = (1-q)\, q_c$, with:

$$f_l = \frac{T - T_i}{T_0 - T_i}, \tag{100}$$

with $T_0 = 273.15\ K$, and $T_i = 253.15\ K$ (the temperature below which the liquid and ice fractions are 0 and 100% respectively). Auto-conversion rates calculated separately for liquid and ice following [13] are then combined as follows:

$$\left.\frac{\partial q_r}{\partial t}\right|_{auto} = f_l Q_{auto,l} + (1 - f_l)\, Q_{auto,i}, \tag{101}$$

Accretion is parameterized following a very similar method, again distinguishing between liquid and ice particles using $f_l(T)$. Accretion rates of non-precipitating particles by precipitating particles is then expressed as:

$$Q_{acc} = \frac{\pi}{4} \alpha E N \overline{D}_r^2 V_{p,r}\left(\overline{D}_r\right) q_c, \tag{102}$$

with E the accretion efficiency and α the ratio of the particle's surface area and that of a perfect sphere of same size (see table 2). $\overline{D}$ is the diameter of a particle with mean mass $\overline{m} = q/N$, i.e. $\overline{D} = (q/a_m N)^{(1/b_m)}$, and $N = N_0/\lambda$. The precipitation velocity is expressed as a power of the particles' size: $V_p = a_v D^{b_v}$. Again, once the accretion rates have been calculated for both liquid and ice particles separately, the total accretion rate is calculated using an equation similar to 101.

Condensation/evaporation (or deposition/sublimation) of precipitating particles is parameterized using an expression similar to equation 89 used in the SB scheme. Again, phase change rates calculated separately for ice and liquid particles are combined using the liquid fraction f_l.

Phase changes for non-precipitating particles are parameterized using the saturation adjustment approach which relies on the assumption that in-cloud water vapor relaxes instantaneously to saturation. Let's denote q_c^n and q_v^n the cloud water and vapor concentrations at time t^n, before adjustment, and q_c^{n+1} and q_v^{n+1} the same quantities after adjustment. Knowing that $q_t = q_v + q_c$, the total water content, is conserved, we can write:

$$q_c^{n+1} = q_t^n - q_s\left(p, T^n\right), \tag{103}$$

were q_s is the saturation vapor mixing ratio calculated as a function of T^n and p only. When the cloud water and vapor concentrations after adjustment are known, an updated temperature is obtained using $T^{n+1} = T^n + \frac{L_{lv}}{c_p}\left(q_c^{n+1} - q_c^n\right)$. Because the saturation vapor mixing ratio depends explicitly on the temperature, the adjustment procedure must be repeated until q_c, T and q_s converge. In practice, the problem is solved using Newton-Raphson iterations. The final temperature, cloud water and vapor mixing ratios after adjustment are then given by the latest solution of the iterative procedure.

Finally, the mass weighted precipitation velocity used for the gravitational settling of q_r is given by:

$$\overline{V_p} = a_v \frac{\Gamma\left(b_m + b_v + 1\right)}{\Gamma\left(b_m + 1\right)} \lambda^{-b_v}. \tag{104}$$

hydrometeor	a_m	b_m	a_v	b_v	E	α
precip. liquid	523.6	3	130	0.5	0.8	1
precip. ice	0.025	2	4	0.25	0.2	0.3

Table 2: Default microphysical parameters for the Grabowski scheme and all kinds of hydrometeors. Units are SI (kg-m-s).

Time integration: In contrast to all other physical tendencies (including turbulent diffusion, nudging, radiation and others), microphysical tendencies are computed and integrated outside of the main time loop. The adopted split-explicit strategy gives better control on the stability and physical significance of the numerical solution.

In short, denoting $\mathcal{M}_\Phi$ the collection of all microphysical tendencies applied to a quantity Φ, and $\mathcal{T}_\Phi$ all other tendencies (including advection), the solution Φ^{n+1} at time t^{n+1} is obtained in two steps:

$$\Phi^* = \Phi^n + \Delta t \mathcal{M}_\Phi\left(\Phi^n\right) \tag{105}$$

$$\Phi^{n+1} = \Phi^* + \Delta t \mathcal{T}_\Phi\left(\Phi^*\right), \tag{106}$$

where we have employed the finite-volume approach to discretize the second, advective step.

The simple two-step splitting procedure described above is theoretically only first-order accurate. Second-order accuracy can be achieved easily using the so-called three-step Strang splitting [19]:

$$\Phi^* = \Phi^n + \frac{1}{2}\Delta t \mathcal{M}_\Phi\left(\Phi^n\right) \tag{107}$$

$$\Phi^{**} = \Phi^* + \Delta t \mathcal{T}_\Phi\left(\Phi^*\right) \tag{108}$$

$$\Phi^{n+1} = \Phi^{**} + \frac{1}{2}\Delta t \mathcal{M}_\Phi\left(\Phi^{**}\right). \tag{109}$$

With this method, microphysical tendencies must be computed twice per time-step. This obviously implies a certain computational overhead. Strang splitting can be selected by setting *split_mic* to *.true.* in *cm.nml*.

3.4.5 Droplet activation

It is sometimes enough to assume a fixed cloud droplet number concentration which doesn't require any specific parameterization for activation. This is done by setting *lndrop = 0* in *cm.nml* and specify the desired number concentration as *xn_ccn0*. In most situations however, activation needs to be modeled explicitly. This can be achieved using either a detailed representation of the aerosol population and aerosol physics (*setenv AEROSOL TRUE* in *start*, as described in section **??**), or a simplified parameterization as proposed by [15, 16] (*setenv AEROSOL FALSE* in *start*).

The latter relies on a log-normally distributed CCN population with mean size *xnc0_d* and standard deviation *xnc0_s* in *cm.nml*. The CCN concentration is assumed to be homogeneous across the domain and equal to *xn_ccn0*. The number of activated droplets within a given grid box with temperature T and water vapor mixing ratio q_v is then computed as:

$$N_{act} = \frac{1}{2} N_{CCN}\left[1 - erf\left(X\right)\right], \tag{110}$$

with:

$$X = \frac{\log\left(s_0/s\right)}{1.5\sqrt{2}\log\sigma}. \tag{111}$$

In the above, erf is the error function, N_{CCN} is the local CCN concentration, σ is the CCN size-distribution standard deviation, s is the local saturation ratio $q_v/q_s - 1$, s_0 is a critical saturation ratio connected to hygroscopicity (specified as *xnc0_k* in *cm.nml*). A detailed expression for s_0 is given by [16]. N_{act} only represents the number of particles that can potentially be activated within a time step. The rate at which cloud droplets are activated is thus given by:

$$\left.\frac{\partial N_c}{\partial t}\right|_{act} = \frac{\max(N_{act} - N_c, 0)}{\Delta t}. \tag{112}$$

The mass of activated water vapor can be computed diagnostically during the saturation adjustment step (if *SAT_ADJ* is *TRUE* in *start*) in which case no further calculation is required. When condensation/evaporation is treated explicitly, the subsequent growth of newly activated particles must be taken into account to determine the total mass of vapor condensed. This is done using small time-step iterations as explained in section 3.5.3.

3.4.6 Ice nucleation

Three ice nucleation parameterizations are currently implemented in MIMICA. Each of these can be selected by setting the appropriate value of *lfreeze* in *cm.nml* (from 0, to disable ice nucleation, to 3). Moreover, it is possible to only activate ice nucleation after a given time, prescribed in *cm.nml* through *ice_delay* (in seconds), in order to allow the model to spin-up without ice. All three available schemes are described below, from the simplest to the more complex one.

Simple relaxation model (*lfreeze = 1*): Ice nucleation is here calculated as a simple relaxation of the ice number concentration towards a fixed background IN concentration [6]:

$$\left.\frac{\partial N_i}{\partial t}\right|_{nuc} = \frac{N_i - N_{IN,0}}{\Delta t}, \tag{113}$$

where $N_{IN,0}$ is set in *cm.nml* via *xn_in0*. Ice nucleation is only active under case dependent conditions. By default, nucleation is only allowed when $q_c > 10^{-3}\ g.kg^{-1}$ and supersaturation over ice exceeds 5%. All newly nucleated ice crystals are assumed to have the same mass, $m_i = 4.2 \times 10^{-15}\ kg$.

Exponential model (*lfreeze = 2*): Ice nucleation is here parameterized as in equation 113. The difference is that IN_0 is not held constant anymore but depends on temperature [3]:

$$N_{IN,0} = 117 \exp\left[0.125\left(273.15 - \max\left(T, 233\right)\right)\right]. \tag{114}$$

Nucleation is only allowed when $q_c > 10^{-7}$ kg/kg and $T < 267.15$ K K, or if supersaturation with respect to ice exceeds 2% and $T < 248.15$ K. The nucleation rate for the ice mass concentration is again calculated using a fixed minimum ice crystal mass of $m_i = 4.2 \times 10^{-15}\ kg$.

In addition, rain freezing is allowed at this level. However, for simplicity, it is assumed that all rain drops freeze instantaneously below $T = 269$ K.

Diehl and Wurzler parameterization (*lfreeze = 3*) : This scheme parameterizes immersion freezing as a function of temperature, aerosol nucleation efficiency, and cloud droplets volume. According to [4], the fraction of ice nucleating particles (INPs) forming ice at a given time is:

$$N_{IN,k} = f_k\left[1 - \exp\left(-J_k \delta_k \Delta t\right)\right], \tag{115}$$

with Δt being the time-step, $\delta = 1/q_c dq_c/dt - a_k dT/dt$ is the rate at which INPs become able to nucleate ice (increases when new INPs are advected with liquid droplets or activated, and when the temperature decreases), and:

$$J_k = V_c B_k \exp\left[-a_k\left(T - T_0 - \Delta T_f\right)\right]. \tag{116}$$

In the above relationship, V_c is the mean volume of a cloud droplet, a quantity straightforward to estimate from q_c and N_c, $T_0 = 273.15\ K$, and ΔT_f is the freezing point depression (see [4]). Three types of INPs are considered here: dust, black carbon and bio-particles. Each type is characterized by the couple a_k-B_k (defined explicitly for each INP type), as well as the fraction f_k representing the fraction of INP k in the aerosol population (defined as a fraction of *xn_ccn0*). Once $N_{IN,k}$ is known for each particle type, the total nucleation rate is computed as:

$$\left.\frac{\partial N_i}{\partial t}\right|_{nuc} = \frac{N_i - \sum_{k=1}^{3} N_{IN,}}{\Delta t}. \tag{117}$$

Homogeneous freezing (only applied to cloud droplets, not rain) is calculated independently of the heterogeneous nucleation model, and is added to the total freezing rate. Homogeneous freezing is obtained from nucleation theory, and only computed below 240 K.

3.4.7 Damping, nudging and large scale tendencies

So-called sponge layers are generally imposed at the top of the numerical domain to damp gravity waves reflecting against the top boundary [5]. Damping is achieved by adding a source term of the form:

$$N_\Phi(x,y,z,t) = -C_\Phi(z)\,\frac{\Phi - \Phi_0(z)}{\tau}. \tag{118}$$

to the relevant governing equation (Φ is either the velocity, potential temperature or total water mixing ratio). Φ_0 is chosen to be the initial Φ profile. The damping strength is characterized by the time scale τ which can be prescribed in *cm.nml* via *tdamp*. The altitude dependent coefficient C_Φ allows a smooth transition between the base and the top of the sponge. Damping is only applied above altitude *zdamp* as prescribed in *cm.nml*). In general, reasonable results are obtained when the sponge layer occupies the upper third to upper fourth of the numerical domain. Note that the same model can be used to implement horizontal sponge layers along lateral boundaries. In this case, the thickness of these layers can be set with *dxdamp* and *dydamp* in *cm.nml*.

In some cases, the simulated wind and scalar fields are nudged to a given reference state (nudging is applied only if *with_nudg = .true.* in *cm.nml*). Nudging is based on the same principle as the damping layer described above except that C_Φ is now a user defined, case dependent function of altitude, and τ is now set via *tnudg* in *cm.nml*.

Large-scale scalar horizontal advection can be considered in a similar fashion. It is generally modeled as a constant, case dependent source added to the scalar tendencies (only for potential temperature and total water mixing ratio). These sources can possibly be time dependent.

Large-scale upwelling/downwelling is modelled assuming a prescribed background vertical velocity profile w_0. The background velocity can be prescribed in different ways: directly, with a constant value *w_up* in *cm.nml*, using a constant horizontal wind divergence *ddiv* in *cm.nml*, in which case $w_0(z) = -D_{div}z$, or by adding a specific w_0 column in the initial sounding file. Transport in the vertical direction by w_0 is then computed in advective form and is added as a source term to the relevant equation. By default, only the potential temperature field is affected by large-scale vertical transport. Other scalars (total water content and passive tracers) can also be transported by setting *lssub = .true.* in *cm.nml*.

3.4.8 Tracers

MIMICA offers the possibility to define and transport additional tracers for specific applications. While the basic set of prognostic variables includes the velocity vector, potential temperature, the total water mass concentration and microphysical variables (mass and number concentrations of selected hydrometeors), the *NSCAL* (with *NSCAL* set in *start*) additional scalars will obey similar basic governing equations (includes by default advection and turbulent diffusion), with predefined or user-defined initial conditions, and the possibility to add specially defined sources, sinks and even surface fluxes.

Predefined sets of additional tracers can be selected using parameter *sca_set* in *cm.nml*.

1. setting *sca_set = 1*, three tracers are defined to diagnose lateral entrainment/detrainment in cumulus clouds. The three scalars are defined as so-called purity tracers and are equal to

0 or 1 depending on the location (in general 1 inside cloudy areas or cloudy updrafts, and 0 everywhere else). Using this option, the selected scalars are transported using the usual governing equations, but their values are reset and recomputed at the end of each time step from the resolved flow field. The difference between the transported and diagnosed scalar values is used to calculate entrainment/detrainment.

2. setting *sca_set = 2*, a single tracer is added whose initial profile mimics that of ozone in the tropics. No chemistry is currently implemented, but this pseudo-ozone tracer is allowed to interact with radiation and specific nudging in the lower troposphere can be selected to model ozone sinks in this part of the atmosphere.

It is possible for anyone interested to implement new tracers or new sets of tracers in a very simple way. All the transport and mixing is handled by default in MIMICA, and the only thing that needs to be done concerns defining initial conditions and possible sources and sinks. This is all done in a single routine (*scalar_source.f90*), and the tracers already defined as part of the scalar sets described above can be used as templates.

3.5 Aerosols

A bulk aerosol model is implemented in MIMICA that allows the simulation of detailed cloud-aerosol processes using a very flexible framework. The module becomes active when *AEROSOL* is set to *TRUE* in *start*, and requires specific input parameters in *cm.nml*.

3.5.1 Bulk aerosol representation

In MIMICA, the aerosol population is composed of several modes (*nmode0* being the number of modes specified in *cm.nml*), each defined by a series of parameters characterizing its composition and distribution in size.

Each aerosol mode is composed of a mixture of 4 compounds present in different fractions (to be defined in *cm.nml* with *aero%init%present* and *aero%init%frac*): sulfates, black carbons, sea salt and organics. The hygroscopicity (as defined by [25]), density and molar weight of each of these elementary compounds are given values of $\{0.53, 0.01, 1.12, 0.06\}$, $\{1840, 600, 2180, 1560\}$ and $\{98, 12, 58.4, 104\}$ (for sulfates, black carbons, sea salt, and organics). The mixed properties of each aerosol mode are then calculated using volume weighted averages of the properties of each element constituting individual particles. For example, mixed hygroscopicity is calculated following [25]:

$$\kappa = \sum_{k}^{4} \epsilon_k \kappa_k, \tag{119}$$

with κ the mixed aerosol hygroscopicity, κ_k the kth element hygroscopicity, and $\epsilon_k = V_{d,k}/V_d$, with V_d the volume of the dry particle and $V_{d,k}$ the volume occupied by component k alone. Ignoring all physical and chemical processes affecting the composition of the dry aerosols, κ is assumed to remain constant for all aerosol modes during the course of a simulation, and is only determined by the initial composition assigned to each mode. The latter assumption simplifies greatly the problem of characterizing the properties of the whole aerosol population

Concerning the size representation, each mode k is assigned a log-normal distribution of the form [32]:

$$f_k(r) = \frac{dn_k}{dr} = \frac{n_k}{\sqrt{2\pi} r \log \sigma_k} \exp\left[-\frac{(\log r - \log r_{g,k})^2}{2 \log^2 \sigma_k} \right], \tag{120}$$

with σ_k the geometric standard deviation, and $r_{g,k}$ the geometric mean radius of aerosol mode k. The distribution is initialized by giving the standard deviation via *aeroi%size%sigma* and the initial geometric radius via *aeroi%size%rmean* in *cm.nml*. While σ_k is held constant during the simulation, the geometric mean radius $r_{g,k}$ is allowed to vary. It can be deduced for each mode from the knowledge of the number and mass concentrations (n_k and x_k) which are directly related to the 0th and 3rd moments of the aerosol size distribution.

By definition of the log-normal distribution, we can write:

$$n_k = \int_0^{+\infty} f_k(r)\, dr, \tag{121}$$

and:

$$x_k = \frac{4}{3}\pi\rho_k \int_0^{+\infty} r^3 f_k(r)\, dr, \tag{122}$$

with ρ_k being the density of aerosol mode k. By definition of the moment generating function of a log-normal distribution, x_k can then be expressed as:

$$x_k = \frac{4}{3}\pi\rho_k r_{g,k}^3 n_k \exp\left(\frac{9}{2}\log^2\sigma_k\right). \tag{123}$$

Solving transport equations for both n_k and x_k, the geometric mean radius $r_{g,k}$ can thus be diagnosed using equation 123. Similarly, while the initial mode number concentration is prescribed in *cm.nml* via *aeroi%n0*, the initial mass concentration is recomputed from the knowledge of the initial mode geometric size and standard deviation.

Without further details, the equations for n_k and x_k solved in MIMICA read:

$$\frac{\rho n_k}{\partial t} + \nabla\cdot(\rho\mathbf{u}n_k) = \mathcal{A}_{n,k} + \mathcal{R}_{n,k} + \mathcal{I}_{n,k} \tag{124}$$

$$\frac{\rho x_k}{\partial t} + \nabla\cdot(\rho\mathbf{u}x_k) = \mathcal{A}_{x,k} + \mathcal{R}_{x,k} + \mathcal{I}_{x,k}. \tag{125}$$

In order to keep track of the total aerosol population, an equation for the mass of activated particles, a_k, is also solved for each mode k:

$$\frac{\rho a_k}{\partial t} + \nabla\cdot(\rho\mathbf{u}a_k) = -\mathcal{A}_{x,k} - \mathcal{R}_{x,k} - \mathcal{I}_{x,k} + \mathcal{P}_{x,k}. \tag{126}$$

where $\mathcal{A}_{n,k}$ (resp. $\mathcal{A}_{x,k}$), $\mathcal{R}_{n,k}$ (resp. $\mathcal{R}_{x,k}$), and $\mathcal{I}_{n,k}$ (resp. $\mathcal{I}_{x,k}$) represent aerosol number (resp. mass) concentration tendencies due to activation, regeneration (the former acting as a sink and the latter as a source of aerosol particles) and impaction scavenging. $\mathcal{P}_k$ represents precipitation scavenging of activated aerosol particles.

Four different levels of complexity are available in MIMICA, controlled by *laero* in *cm.nml*. *laero = -1* maintains the number and mass aerosol concentrations constant everywhere, *laero = 0* introduces advection and removal by activation, *laero = 1* adds precipitation scavenging and regeneration, while with *laero = 2* the all physical processes including impaction scavenging are treated.

3.5.2 Activation

A commonly used approximation of the equilibrium saturation s over a solution droplet of radius r that provides reasonably good results over a wide range of κ values is given by [32]:

$$s(r) = \frac{A}{r} - \frac{\kappa r_d^3}{r^3} \tag{127}$$

with $A = \frac{2\sigma_{s/a} M_w}{\rho_w RT}$, R the universal gas constant, $\sigma_{s/a}$ the surface tension at the solution/air interface, M_w and ρ_w the molecular weight and density of water, and T the air temperature. Using the above equation, the critical saturation ratio and critical wet particle size can be found easily from the condition $ds/dr = 0$. This yields:

$$s_c = \sqrt{\frac{4A^3}{27\kappa r_d^3}} \text{ and } r_c = \sqrt{\frac{3\kappa r_d^3}{A}}. \tag{128}$$

Now, due to the simple relationship between r_d and the critical saturation s_c, the activation spectrum can be calculated [16, 7]:

$$\frac{dn_k}{ds} = \frac{n_k}{\sqrt{2\pi} s \log \sigma_k^{3/2}} \exp\left[-\frac{\left(\log s - \log s_{c,k}\right)^2}{2\log^2 \sigma_k^{3/2}}\right], \tag{129}$$

Further assuming that the particles that activate under a certain model supersaturation s are those for which $s_c < s$, we can estimate the number of activated particles as:

$$n_{act,k} = \frac{n_k}{2}\left[1 - erf\left(\frac{\log s - \log s_{c,k}}{\sqrt{2}\log \sigma_k^{3/2}}\right)\right], \tag{130}$$

where $n_{act,k}$ is the number of activating particles in mode k. We then can simply define:

$$\mathcal{A}_{n,k} = -\frac{n_{act,k}}{\Delta t}. \tag{131}$$

Note that if kinetic growth of activated droplets is explicitly accounted for, and equation for s is solved with a small time-step $\Delta\tau$ and is therefore estimated explicitly. s is otherwise chosen as the maximum between the modeled saturation at the beginning of the time-step and at the end (s at time t^{n+1} is then guessed by integrating equation 136 explicitly).

To determine the mass of activated particles, we simply note that the aerosol mass distribution can be deduced from equation 120 and reads:

$$\frac{dn_k}{dm} = \frac{n_k}{\sqrt{2\pi} m_k \log \sigma_{m,k}} \exp\left[-\frac{\left(\log m - \log m_{g,k}\right)^2}{2\log^2 \sigma_{m,k}}\right], \tag{132}$$

with $m_{g,k} = 4/3\pi\rho_k r_{g,k}^3$ the mass of a particle of size $r_{g,k}$ (all particles are assumed to be spherical) and $\sigma_{m,k} = \sigma_k^3$. The total mass of particles larger than $r_{act,k}$ can thus be obtained following:

$$x_{act,k} = \int_{m_{act,k}}^{+\infty} m \frac{dn_k}{dm} dm \tag{133}$$

$$= \frac{n_k}{2} m_{g,k} \exp\left(\frac{1}{2}\log^2 \sigma_{m,k}\right)\left[1 - erf\left(\frac{\log m_{act,k} - \log m_{g,k} - \log^2 \sigma_{m,k}}{\sqrt{2}\log \sigma_{m,k}}\right)\right]. \tag{134}$$

It then follows that:

$$\mathcal{A}_{x,k} = -\frac{x_{act,k}}{\Delta t}. \tag{135}$$

Note that when *laero = -1*, the potential number of activated particles is calculated as described above, but since the overall aerosol concentration is maintained constant, the number of activated droplets is determined based on a method similar to equation 112 (see simple CCN activation method, section 3.4.5).

3.5.3 Kinetic growth

In the more general case (that is for *SAT_ADJ FALSE* in *start*), newly activated particles will immediately grow within updrafts where supersaturated conditions prevail. Growth by vapor condensation is, in this case, limited by a balance between the increase of supersaturation through adiabatic cooling and its decrease through condensation. This balance can be explicitly accounted for using small time-step integration when *lkin* is *.true.* in *cm.nml*. By default (i.e. in the absence of explicit kinetic growth), the mass of newly activated droplets is set to a constant value of 4.19×10^{-15} kg (corresponding to a droplet of radius 1 μm).

Kinetic growth of activated particles can be described by the following equations [24, 7]:

$$\frac{ds}{dt} = (s+1)\left(\alpha - \gamma \frac{dq}{dt}\right), \tag{136}$$

with dq/dt the total condensation rate over all activated particles, and:

$$\alpha = \frac{L_v}{RvT^2}\beta - \frac{g}{R_a T} w, \tag{137}$$

$$\gamma = \frac{1}{q_v} + \frac{L_v^2}{c_p R_v T^2}. \tag{138}$$

In the above, w is the updraft velocity and $\beta \approx -w\partial T/\partial z + Q_R$ is the sum of adiabatic and radiative cooling contributions. As s continues to increase over a time step ($ds/dt > 0$), further particles can activate and subsequently grow. To treat this continuous activation/growth process, a split-explicit method is employed where the problem is integrated using a small time-step ($\Delta\tau$) at the moment fixed to 0.5 s.

First, activation is calculated at the beginning of the time-step (at time t^n) giving a first estimate of $n_{act,k}$ corresponding to a supersaturation ratio s^n. Equation 136 is then solved to estimate s at time $t^n + \Delta\tau$, with:

$$\frac{dq}{dt} = 4\pi G\,(s - s_{eq}) \sum_{k}^{N_{mode}} \overline{r}_{w,k} n_{act,k}, \tag{139}$$

with s_{eq} the equilibrium saturation over a particle of size r_w (given by equation 127), G is given by equation 90 (the dependence of diffusivity on particle size is not considered here) and $\overline{r}_{w,k}$ is the mean wet radius of particles just activated. Just like we did with the mean mass of activated particles, it can be shown that the particles wet size distribution follows:

$$\frac{dn_k}{dr_w} = \frac{n_k}{\sqrt{2\pi} r_{w,k} \log \sigma_{r,k}} \exp\left[-\frac{\left(\log r_{w,k} - \log r_{wg,k}\right)^2}{2\log^2 \sigma_{r,k}}\right], \tag{140}$$

with $r_{w,k}$ the critical wet radius for a particle of size $r_{g,k}$ and $\sigma_{r,k} = \sigma_k^{3/2}$. The total mass of particles larger than $r_{act,k}$ can thus be obtained following:

$$\overline{r}_{w,k} = r_{wg,k} \exp\left(\frac{1}{2}\log^2 \sigma_{r,k}\right) \frac{\left[1 - erf\left(\frac{\log r_{w,k} - \log r_{wg,k} - \log^2 \sigma_{r,k}}{\sqrt{2}\log \sigma_{r,k}}\right)\right]}{\left[1 - erf\left(\frac{\log r_{act,k} - \log r_{g,k}}{\sqrt{2}\log \sigma_k}\right)\right]}. \tag{141}$$

Similarly, the total amount of condensed water vapor as well as the temperature can be updated at time $t^n + \Delta\tau$.

If ds/dt is positive, supersaturation increases, and the number of particles that activate when going from $s = s^n$ to $s = s^n + \Delta\tau ds/dt$ is recomputed. $n_{act,k}$, $x_{act,k}$ can thus be updated as well as the mass of condensed water and, finally, the supersaturation. It should be noted that in the algorithm implemented, the distinction is made between the smaller particles that form at each small time-steps and those that were formed previously but continue to grow by condensation.

The procedure is repeated over subsequent small time-steps until time t^{n+1}. Again, New particles activate and already formed particles grow as long as $ds/dt > 0$, but these existing particles will also evaporate if $ds/dt < 0$.

3.5.4 Regeneration

In practice, regeneration $\mathcal{R}_{x,k}$ for each aerosol mode is evaluated following:

$$\mathcal{R}_{x,k} = -a_k \frac{\sum_{j=1}^{N_{hydro}} \frac{dN_j}{dt}\Big|_{evap/sublim}}{\sum_{j=1}^{N_{hydro}} N_j}, \tag{142}$$

with N_j the number concentration of hydrometeors of type j, and $dN_j/dt|_{evap/sublim}$ the evaporation/sublimation rates of hydrometeor type j as computed by the selected microphysics scheme (negative by definition).

The number of regenerated aerosols is calculated by assuming that the mean mass of regenerated particles is equal to the mean mass of the original aerosol population:

$$\overline{m}_k = \frac{4}{3}\pi\rho_k r_{g,k}^3 \exp\left(\frac{9}{2}\log^2\sigma_k\right), \tag{143}$$

that is:

$$\mathcal{R}_{n,k} = \frac{\mathcal{R}_{x,k}}{\overline{m}_k}. \tag{144}$$

3.5.5 Precipitation scavenging

Precipitation scavenging of activated particles is estimated as follows:

$$\mathcal{P}_{x,k} = a_k \frac{\sum_{j=1}^{N_{hydro}} \frac{\partial \rho N_j V_{p,j}}{\partial z}}{\sum_{j=1}^{N_{hydro}} N_j}, \tag{145}$$

where $V_{p,j}$ is the terminal fall speed of hydrometeors in category j, and the summation is performed over all hydrometeor categories.

3.5.6 Impaction scavenging

As they fall, hydrometeors collect aerosol particles. The rate at which aerosol particles in mode k are collected by all hydrometeors can be expressed as:

$$\mathcal{I}_{n,k} = -n_k \sum_{j=1}^{N_{hydro}} I_{j,k}, \tag{146}$$

with $I_{j,k}$ the reduced collection rate between hydromteteor j and particle k:

$$I_{j,k} = \frac{\pi}{4}\overline{E}_{j,k}\overline{D_j^2 V_{p,j}} N_j, \tag{147}$$

with $\overline{D_j^2 V_{p,j}}$ the mean product of the hydrometeor size squared and precipitation velocity (which stems from the integration over the hydrometeor size distribution), and $\overline{E}_{j,k}$ the mean impaction efficiency between the hydrometeor and aerosol particles. Integrating impaction scavenging alone over a time-step, we can rewrite the impaction rates implicitly as [1]:

$$\mathcal{I}_{n,k} = -\frac{1 - \exp\left(-\Delta t \sum_{j=1}^{N_{hydro}} I_{j,k}\right)}{\Delta t}. \tag{148}$$

A similar expression is obtained for $\mathcal{I}_{x,k}$.

Impaction efficiencies are parameterized following for example [1] and include contributions from Brownian motions, interception and inertial impaction. For simplicity, the average efficiencies $\overline{E}_{j,k}$ are calculated using mean hydrometeor and aerosol properties (mean mass, size and precipitation velocities).

3.6 Lagrangian particle tracking

MIMICA offers the possibility to seed the numerical domain with multiple particles, and subsequently track their trajectories over the course of the simulation using a lagrangian framework. Lagrangian particle tracking is turned on by setting *LAGRANGE TRUE* in *start*, while *NPART* indicates the number of particles to intitialize.

At the beginning, the position of each particle is chosen randomly within a prescribed part of the domain (to be defined with *lag_init* in *cm.nml*). Particles are then allowed to follow the simulated flow by solving the following lagrangian equation:

$$\frac{d\mathbf{x}_k}{dt} = \mathbf{u}_k + \mathbf{u}'_k, \tag{149}$$

where $\mathbf{x}_k$ indicates the position in cartesian coordinates of particle k, $\mathbf{u}_k$ is the flow velocity interpolated at location $\mathbf{x}_k$ and $\mathbf{u}'_k$ is a turbulent contribution. This latter is calculated as follows:

$$\mathbf{u}'_k = \mathbf{r}_k \sqrt{\frac{2}{3} e'_k} \tag{150}$$

with e'_k being the turbulent kinetic energy interpolated at parcel k's location and $\mathbf{r}_k$ is a vector of 3 random numbers normally distributed between 0 and 1.

While the basic principles and equations are relatively simple, one of the difficulties of the tracking algorithm resides in the many interpolations needed to estimate flow properties at the parcels locations. Interpolations are required to estimate particle velocities appearing in equation 149, but also to estimate basic flow properties (in particular thermodynamic properties) for diagnostic and output purposes. For example, the algorithm gives access to the values of temperature, moisture, buoyancy, relative humidity, as well as several tendencies at the location of each seeded particle. In the present algorithm, no calculation is performed to estimate particle properties. These are always interpolated from the eulerian model solution.

Two interpolation procedures are currently available and accessible via *lag_ord* in *cm.nml*. The first method relies on the application of a three-dimensional Gaussian filter centered at the location of each parcel. The filter has a radius of $\approx 0.8\Delta x$. The second method relies on simple linear interpolations between neighbouring grid points values.

It should be noted finally that the algorithm is fully parallelized. Initially, each particle is assigned to the sub-domain to which it belongs. Solving equation 149 for all particles can thus be done in parallel. Later, as particles cross sub-domain boundaries, they are also moved from one processor to another.

4 MIMICA inputs

4.1 *start*

start is a csh script piloting the compilation of MIMICA. Compilation is simply accomplished by typing

```
./start
```

in your local working directory. *start* also contains a series of configuration options, controlling, for example, the selection of various physical modules, the size of the domain or MPI decomposition. Each option can be activated (resp. deactivated) by uncommenting *setenv XXXXX TRUE* (resp. *setenv XXXXX FALSE*), with *XXXXX* being the desired option keyword. Note that MIMICA must be recompiled entirely after changing anything in *start* by first cleaning all the directory (`make clean`). A *compile.log* file is created locally in your working directory each time *start* is executed. This file contains a summary of all the configuration options used to create the last MIMICA executable.

A description of all options currently available in *start* is provided below.

MIMICA, *INCDIR*, *DATADIR*, *NETCDF* It is important here to replace whatever is defined by default by the paths to your own MIMICA directory, INCLUDE directory (can be replaced by $MIMICA/INCLUDE), DATA directory (can be replaced by $MIMICA/DATA), and finally the path to the netCDF repository you want to compile MIMICA with.

CMPLER **(default: commented)** Setting *setenv CMPLER INTEL* enables compilation with the intel ifort compiler. The default compiler (if *CMPLER* is left commented) is the gfortran compiler.

DEBUG **(default:** *setenv DEBUG FALSE***)** Setting *setenv DEBUG TRUE* turns on all recommended debug options. The MIMICA executable thus created is usually larger than an optimized one, and its execution is about 2 to 10 times slower. It is however strongly recommended to use the DEBUG configuration and run MIMICA with debug options when problems possibly related to coding errors is encountered at run time.

PROF **(default:** *setenv PROF FALSE***)** Setting *setenv PROF TRUE* turns on profiling (with gprof) options. The MIMICA executable thus created can be run to estimate the time spent in each routine at run time. This option should be used for optimization purposes.

SPMD **(default:** *setenv SPMD FALSE***)** Setting *setenv SPMD TRUE* allows parallelization with MPI. MIMICA is then compiled using the appropriate MPI wrapper, and must be executed using commands like *mpirun* or *mpiexec*.

M3D **(default:** *setenv M3D FALSE***)** Setting *setenv M3D TRUE* will compile MIMICA for a three-dimensional simulation.

DECOMP_2D **(default:** *setenv DECOMP_2D FALSE***)** This option is only relevant when *SPMD* and *M3D* are both set to *TRUE* (three-dimensional parallel simulation compiled with MPI). Setting *setenv DECOMP_2D TRUE* enables a two-dimensional decomposition of the three-dimensional domain. When set to *FALSE* (1D decomposition, default), subdomains are created by splitting the numerical domain NP times along the x direction (NP is here the total

number of processors). However, when *setenv DECOMP_2D TRUE*, the numerical is split *NPX* times along x, and *NPY* times along y. If the numerical domain is big enough, a two-dimensional decomposition is generally more computationally efficient than a one-dimensional decomposition.

NPX **and** *NPY* *NPX* and *NPY* correspond to the number of subdivisions of the numerical domain along dimensions x and y respectively. If *setenv DECOMP_2D TRUE*, the total number of subdomains (also corresponding to the number of processors to be used) is given by $NP = NPX \times NPY$. Otherwise, in a 1D decomposition case, $NP = NPX$, and *NPX* is redefined as the product of *NPX* and *NPY* (although it is recommended to set *NPY = 1* in this situation).

NPART **and** *NSCAL* **(default: 0)** *NPART* is the number of lagrangian parcels initialized in the numerical domain (if *LAGRANGE TRUE*). Similarly, *NSCAL* is the number of additional tracers to be transported (predefined sets of tracers can be selected in *cm.nml*, but users can also code their own additional tracers).

FINE **(default:** *setenv FINE FALSE***)** If *setenv FINE TRUE*, a stretched grid will be used in the vertical dimension. The type of grid stretching used depends on the *casename* in *cm.nml*.

NESTING **(default:** *setenv NESTING FALSE***)** Setting *setenv NESTING TRUE* allows nested simulations (very very experimental).

CHANNEL **(default:** *setenv CHANNEL FALSE***)** Enables channel like configurations with free slip boundary conditions along Y axis instead of periodicity (only in 3D). Typically used to simulate the tropical atmosphere centered on the equator.

PARALLEL_OUT **(default:** *setenv PARALLEL_OUT FALSE***)** Setting *setenv PARALLEL_OUT TRUE* will create parallel outputs: there will be one netCDF file per processor or subdomain (i.e. *NP* full output files). This option can be interesting if outputting the full 3D domain in very large simulations is absolutely necessary (the amount of data that can be written at once in netCDF files is indeed limited). In general, it is however not recommended to output such large datasets altogether.

MEANDATA **(default:** *setenv MEANDATA FALSE***)** With *setenv MEANDATA TRUE*, temporally averaged data are output instead of instantaneous ones. The averaging frequency is defined in *cm.nml* with *iav*.

RADIA **(default:** *setenv RADIA FALSE***)** Setting *setenv RADIA TRUE* allows the use of the interactive radiative transfer model. If *FALSE*, radiation is either completely turned off, or a simplified representation of cloud related long-wave cooling is used (case dependent option).

CHEM **(default:** *setenv CHEM FALSE***)** This option allows MIMICA's chemistry module. The module has not been revised in a while and it is currently not recommended to enable it.

AEROSOL **(default:** *setenv AEROSOL FALSE***)** Setting *setenv AEROSOL TRUE* enables MIMICA's aerosol module. If set to *FALSE*, a single bulk CCN population will be used for droplet activation. Options for the aerosol parameterization (number of aerosol modes, initial compositions, concentrations and size distributions) must be set in *cm.nml* under *cm_aero*, while parameter *laero* controls the model's level of complexity (i.e. whether regeneration, scavenging, etc must be included).

TKE **(default:** *setenv TKE FALSE***)** With *setenv TKE TRUE*, turbulent mixing is parameterized using the 1.5th order scheme with prognostic turbulent kinetic energy. If *FALSE* (default), turbulent viscosity is calculated using the standard Smagorinsky-Lilly model.

LAGRANGE **(default:** *setenv LAGRANGE FALSE***)** Setting *setenv LAGRANGE TRUE* enables lagrangian parcel tracking. The number of tracked parcels is set by *NPART*, and their initial location is either prescribed in an external file (*parcels.dat*, which must be present in the simulation's directory) or drawn randomly inside the numerical domain (if no parcel file is present). Options for the tracking algorithm must be set in *cm.nml* under *cm_lag*. Setting *LAGRANGE TRUE* also allows all parcel related outputs.

ISENTROPIC **(default:** *setenv ISENTROPIC TRUE***)** The isentropic solver (*setenv ISENTROPIC TRUE*) uses potential temperature as the default conserved energy quantity (as described in this document). When *ISENTROPIC* is set to TRUE, potential temperature is replaced with Moist Static Energy, a quantity directly related to enthalpy and therefore to the first law of thermodynamics.

ANELASTIC **(default:** *setenv ANELASTIC TRUE***)** The anelastic solver (*setenv ANELASTIC TRUE*) is used to efficiently filter out acoustic waves. In this situation, numerical stability can be achieved at larger time-steps. In this solver, the density is assumed to be time independent and to vary only with altitude. The continuity equation is therefore used as a mere constraint on the velocity field. The alternative, *setenv ANELASTIC FALSE*, solves the fully compressible set of equations, including an explicit continuity equation. The solver relies on a time-split approach where all terms connected to acoustic perturbations are integrated using a smaller time-step.

CONSERVATIVE **(default:** *setenv CONSERVATIVE TRUE***)** With *setenv CONSERVATIVE TRUE*, all equations are cast and solved in conservative form. As its name says, the option allows to conserve mass and energy to machine precision. When the option is set to *FALSE*, all equations are solved in advective form.

ADV_SPLIT **(default:** *setenv ADV_SPLIT FALSE***)** When the option is set to *TRUE*, advection in all 3 cartesian directions (X, Y, Z) is performed one after the other. For example, a variable Ψ is first advected along X to produce an intermediate solution Ψ^1, Ψ^1 is then advected along Y to give Ψ^2, and Ψ^2 is finally advected along Z. When *FALSE* (default) advection is performed simultaneously along each direction.

ADV_CROW **(default:** *setenv ADV_CROW FALSE***)** When *TRUE*, the alternative 4th or 3rd order Crowley scheme is used for momentum advection. This is a high-order extension of the classical Lax-Wendroff scheme.

RK **(default:** *setenv RK FALSE***)** When *TRUE*, the 2nd order Runge-Kutta time integration scheme is employed.

SAT_ADJ **(default:** *setenv SAT_ADJ TRUE***)** Saturation adjustment is used to calculate cloud droplet evaporation/condensation when set to *TRUE*. Phase changes involving cloud droplets are treated explicitly otherwise, using, by default, a semi-analytical method.

SEIFERT **(default:** *setenv SEIFERT TRUE***)** The option enables the two-moment bulk microphysics scheme as described by Seifert and Beheng. By default, the scheme includes 5 types of hydrometeors for which both the mass and number concentrations are transported. The level of complexity of the scheme can be modified through various *cm.nml* parameters. When *FALSE*, the simple one-moment scheme from Grabowski is used.

NUC_CNT **(default:** *setenv NUC_CNT FALSE***)** Turns on ice nucleation based on classical nucleation theory and a detailed description of the ice nuclei population.

4.2 *cm.nml*

cm.nml is a namelist file containing the most important options and parameters required to run MIMICA. The namelist is organized in 10 "sections":

cm_run contains general options related to time integration and the simulation period

cm_init contains options related model initialization

cm_grid contains options related to the grid and numerical domain

cm_out contains options related to model outputs

cm_num contains options related to the model's numerical schemes

cm_phys contains options related to physical models

cm_bc contains options related to boundary and surface conditions

cm_micro contains options specific to the microphysics schemes

cm_lag contains options specific to lagrangian particle tracking

cm_aero contains options specific to lagrangian particle tracking

Default values for all the parameters contained in *cm.nml* are defined in *INCLUDE/default.h*. If an option is not present in your local *cm.nml* file, the default value is applied automatically. MIMICA does not need to be recompiled when an option is modified in *cm.nml*.

4.2.1 *cm_run*

dt0 **(default:** *dt0 = 1***)** *dt0* is the initial/default model time step in seconds. Unless specified otherwise (see *ldtfix*), *dt0* is used to compute the first time step but is adjusted automatically later on based on the CFL stability criterion.

ldtfix **(default:** *ldtfix = 0***)** If *ldtfix* is set to 0 (default), the time step is adjusted automatically during the course of the simulation to satisfy the CFL stability criterion. If *ldtfix* is set to 1, the time step is held constant and equal to *dt0* during the entire simulation.

limit_ts **(default:** *limit_ts = 20 limit_ts* is the ration between the default time step *dt0* and the minimum time step allowed during the simulation: after adjustment, the time step cannot become smaller than $dt0/limit\_ts$. If it does, the simulation stops with the following error message: "*ERROR: Time-step too small*".

tstart **(default:** *tstart = 0*) *tstart* is the time at initialization.

tstop **(default:** *tstop = 1*) *tstop* is the simulation end time in seconds. For example, if *tstart = 0* and *tstop = 86400*, the simulated period will be exactly 1 day.

ntau **(default:** *ntau = 30*) Number of small time-steps in split-explicit method employed by fully compressible solver (only relevant with *ANELASTIC FALSE*).

ldebug **(default:** *ldebug = .false.*) When true, information on code execution are dumped at run time in *cm.prt*. Typically informs the user when the code enters and leaves specific routines. Useful to know where the code crashes in case of problems.

new_run **(default:** *new_run = .true.*) Simulation starts at time 0 when true. Simulation restarts from state dumped in *restart.dat* when false. Needed to restart from or continue a previous run.

nest_run **(default:** *nest_run = .false.*) Experimental. Allows grid nesting when true: data stored in nest file are used as initial and boundary conditions.

4.2.2 *cm_init*

casename **(default:** *casename = 'none'*) Allows the specification of predefined model configurations. When an existing case name is given, corresponding initial conditions are read from a .h file, and potential case specific model settings are automatically activated (including specific grid layouts, radiative cooling approximations, sources and sinks...).

psurf **(default:** *psurf = 101600.*) Surface pressure (in Pa). This is considered a constant.

dpt **(default:** *dpt = 1.e-4*) Magnitude of initial potential temperature perturbations applied up to *kpert*. Perturbations are drawn from a normal distribution. Often required to speed up model spin-up.

dqv **(default:** *dqv = 0.*) Magnitude of initial water vapor mixing ratio perturbations applied up to *kpert*. Perturbations are drawn from a normal distribution.

dw **(default:** *dw = 0.*) Magnitude of initial vertical velocity perturbations applied up to *kpert*. Perturbations are drawn from a normal distribution.

sca_set **(default:** *sca_set = 0*) Selects a predefined set of passive tracers. The option automatically turns on specific sources, sinks and possible reinitialization procedures corresponding to the selected tracers. Current possibilities include:

- 0: No predefined tracer.
- 1: Three tracers are initialized to diagnose convective clouds (inside cloud cores, in the environment and inside cloud shells).
- 2: A single tracer representative of atmospheric ozone.

kpert **(default:** *kpert = 20***)** Vertical level up to which random perturbations are applied (all perturbations are set to 0 above).

j_day **(default:** *j_day = 67***)** Julian day at the start of the simulation (integer).

ctr_lat **(default:** *ctr_lat = 0.***)** Latitude in degrees. Corresponds roughly to the position at the center of the domain.

t_local **(default:** *t_local = 0.***)** Local time (UST, in hours) at initialisation.

file_init **(default:** *file_init = 'initial.dat'***)** Name of the file containing initial soundings (stored in *INCLUDE*).

file_rest **(default:** *file_rest = 'restart.dat'***)** Name of restart file that will be read if *new_run* is false, and written every *ires* time steps. The file is by default located in *OUTPUT*.

4.2.3 *cm_grid*

dx **(default:** *dx = 50.***)** Cell size in X direction (in m).

dy **(default:** *dy = 50.***)** Cell size in Y direction (in m).

dz **(default:** *dz = 20.***)** Cell size in Z direction (in m). This is not a constant, rather a reference value. The true vertical cell size may depend on altitude and is generally case dependent.

gridfile **(default:** *gridfile = 'none'***)** Name of a file containing the vertical levels used to generate the numerical grid. The file must contain a single column of sorted altitudes corresponding to the altitudes at cell centers. The file must be located in your local working directory.

xc_nest, yc_nest **(default:** *xc_nest = yc_nest = 0***)** Experimental. Defines the position in X and Y of the center of the nested grid.

lx_nest, ly_nest **(default:** *lx_nest = ly_nest = 0***)** Experimental. Length of nested mesh along X and Y.

z1, z2 **(default:** *z1 = 500., z2 = 900***)** Particular altitudes used for grid refinement. Depending on the type of refinement selected (with one or two layers of refinement), the two parameters may not be used.

ztop **(default:** *ztop = 1750.***)** Altitude at the top of the domain. Used by certain grid refinement methods.

sratio **(default:** *sratio = 1.***)** Factor used by certain grid refinement methods to define the size of the smallest vertical grid cells as $z_{min} = sratio \times dz$. Must vary between 0 and 1.

zdamp **(default:** *zdamp = 1500.***)** Altitude where the damping layer starts.

dx_damp, dy_damp **(default:** *dx_damp = dy_damp = 0.***)** Define the thickness of damping layers imposed along the lateral boundaries in X and Y. May be used to simulate pseudo-open domains.

tdamp **(default:** *tdamp = 300.***)** Relaxation time scale imposed in the sponge layer.

tnudg **(default:** *tnudg = 21600.***)** Nudging time scale used to nudge prognostic quantities across the domain. This is a reference value since nudging is often corrected by an altitude dependent factor.

4.2.4 *cm_out*

no_out **(default:** *no_out = .false.***)** When true, the full 3D (if *M3D* is *TRUE* in *start*) or 2D data is not output. It is possible to output

iax **(default:** *iax = 120***)** If *no_out = .false.*, output frequency of full 3D or 2D data in seconds. If *no_out = .true.* the frequency applies to the slices output.

its **(default:** *its = 200***)** Output frequency of time-series data (file T_S) in seconds.

ipro **(default:** *ipro = 300***)** Output frequency of one-dimensional profiles (files *profiles.nc*) in seconds.

ires **(default:** *ires = 300***)** Output frequency of restart file (*restart.dat*) in seconds.

inest **(default:** *inest = 0***)** Output frequency in seconds of nesting file used to restart a nested run (file *nesting.dat*).

iav **(default:** *iav = 120***)** Reset frequency of temporal averages (when *MEANDATA* is *TRUE* in *start*).

minmax **(default:** *minmax = .false.***)** When true, min and max values of relevant prognostic quantities are output at the end of each time-step in *cm.prt*.

zbl, ztrop **(default:** *zbl = ztrop = 0***)** Height of boundary layer and tropopause respectively. Used to compute certain diagnostics.

kout **(default:** *kout = 0***)** When larger than 0, writes full 3D (if *M3D* is *TRUE* in *start*) or 2D data only below vertical level *kout*.

nslicex, nslicey, nslicez **(default:** *nslicex = nslicey = nslicez = 0***)** Number of slices output along directions X, Y and Z. A maximum of 15 slices can be created along each direction. Slice positions are specified as follows.

slicex_io, slicey_io, slicez_io **(default:** *slicex_io = slicey_io = slicez_io = 0.***)** One-dimensional arrays of length 15. Allows the specification of the position of output slices along dimensions X, Y, Z. Must be specified as a list of real numbers separated by comas.

all_res **(default:** *all_res = .false.***)** When true, all restart files are stored separately with an extension corresponding to their respective output times. In the default case, *restart.dat* is overwritten each time model data is dumped for restart.

ts_out **(default:** *ts_out = 'stratus'***)** Type of time series file dumped. Three choices possible: 'stratus', 'cumulus', 'deep'. Each possibility corresponds to a predefined set of scalars output in *T_S* relevant to the selected cloud type.

out_surf **(default:** *out_surf = .false.***)** Enables the calculation and output of surface related quantities (including surface fluxes, surface rain rates) as well as other two-dimensional diagnostics (including LWP, IWP and other column integrated quantities). *nslicez* must be larger than 0. All surface variables will then be dumped to all Z slices output.

out_hov **(default:** *out_hov = .false.***)** Enables the creation of a hovmöller type of output.

out_yav **(default:** *out_yav = .false.***)** Enables the creation of two-dimensional slices in the X-Z plane, averaged along the Y dimension. Useful in long channel configurations.

spec_diag **(default:** *spec_diag = .false.***)** Enables the calculation of advanced diagnostics including turbulence, cloud and thermodynamic diagnostics. This must be turned on to output quantities such as resolved and subgrid scale vertical fluxes, column integrated quantities (such as LWP) or hydrometeor sizes.

file_output **(default:** *file_output = 'mimica.nc'***)** Name of main netCDF output (specified with .nc extension).

4.2.5 *cm_num*

scal_adv **(default:** *scal_adv = 'muscl'***)** Selects the scalar advection scheme. Possible choices currently implemented include: 'muscl', 'quick', 'lw' and 'ppm'.

limit **(default:** *limit = .true.***)** Enables flux limitation for scalar advection when true.

lim_tol **(default:** *lim_tol = 1.e-7***)** Relative tolerance for the application of flux limiters: for limitation to be applied, the ratio between the difference of two successive grid point scalar values and the local scalar value must be larger than *lim_tol*.

imp_buoy **(default:** *imp_buoy = .true.***)** Buoyancy is treated implicitly in the vertical momentum equation when true. May help with under-resolved gravity waves.

split_mic **(default:** *split_mic = 0***)** Three possible choices: when 0, microphysics tendencies are aggregated and added to the total scalar tendencies, when 1, microphysics tendencies are calculated after solving for advection and mixing, when 2, a Strang splitting strategy is applied (microphysics tendencies are integrated over half a time step, then advection over a full time-step, and finally microphysics again over anothe half time-step).

mom_ord **(default:** *mom_ord = 3***)** Order of momentum advection scheme. The stable, upwind biased 3rd order finite-difference scheme is the default.

diff_ord **(default:** *diff_ord = 2***)** Order of second order spatial derivatives: 2 or 3.

p_mcons **(default:** *p_mcons = .false.***)** The pressure correction calculated by the anelastic solver is always determined within a constant. When true, *p_mcons* determines the constant of integration such that the total energy in the domain is conserved (the constant is otherwise calculated such that pressure perturbations average to 0 everywhere).

nusbp **(default:** *nsubp = 1***)** Number of times the pressure correction is calculated when *ANELASTIC* is *TRUE* in *start*. A higher number of iterations leads to a more accurate velocity field (momentum divergence tends to 0) but is computationally more expensive.

cfl_min, cfl_max **(default:** *cfl_min = 0.4, cfl_max = 0.5***)** When *ldtfix = 0*, the time-step is dynamically updated over the course of a simulation such that CFL always remains within these prescribed bounds.

adiv **(default:** *adiv = 0.01***)** Divergence damping coefficient. Helps damping acoustic waves in the fully compressible solver and gravity waves in the anelastic solver.

4.2.6 *cm_bc*

bcl(1-6) **(default:** *bcl(1-6) = ('per','per','per','per','nnf','nnf')***)** Type of boundary conditions applied along X (left and right), Y (left and right) and Z (surface and top). 'per' indicates periodic conditions, 'nnf' indicates a no normal flow condition (symmetry) and 'ope' indicates open boundaries (experimental).

isurf **(default:** *isurf = 0***)** Type of surface conditions:

- 0: fixed surface fluxes.
- 1: fixed surface temperature and moisture, with similarity theory.
- 2: fixed surface temperature and moisture, with fixed drag coefficients.
- 3: like 2 but follows Grabowski's parameterization.
- 4: fluxes calculated based on surface properties computed to result on average in prescribed fluxes.

shf0, lhf0 **(default:** *shf0 = lhf0 = 0.***)** Values of fixed sensible and latent heat fluxes (with *isurf = 0* or 4). These are reference values as fluxes may be defined as time or space dependent functions.

scf **(default:** *scf = 0.***)** Fixed surface scalar flux.

momsf **(default:** *momsf = 0.***)** Momentum surface fluxes are calculated if *momsf = 1.* These fluxes are set to 0 otherwise.

zrough **(default:** *zrough = 1.e-2***)** Surface roughness length (constant across the domain).

ust **(default:** *ust = 0.***)** if *ust* is different from 0., its value is used to specify the surface velocity scale $u^\star$. It is otherwise calculated using similarity theory.

min_w **(default:** *min_w = 0.1***)** Minimum velocity at the surface. This guarantees that surface fluxes are non zero even when horizontal winds are 0 (for example after initialization).

sst, ssm **(default:** *sst = 295., ssm = 1.***)** Fixed surface temperature and relative humidity. Used to calculate surface fluxes with *isurf = 1* for example.

c_dm, c_ds **(default:** *c_dm = 1.2e-3, c_ds = 1.1e-3***)** Fixed drag coefficients (for momentum and scalars respectively) used to calculate surface fluxes with *isurf = 2* and *3*.

alb, emi **(default:** *alb = 0.07, emi = 0.984***)** Surface albedo and emissivity used by radiative transfer model.

4.2.7 *cm_phys*

with_mom **(default:** *with_mom = .true.***)** Enables solving the full momentum equations. Velocity is held constant everywhere otherwise.

with_scal **(default:** *with_scal = .true.***)** Enables solving the scalar equations (including potential temperature and total water mixing ratio). Velocity is held constant everywhere otherwise.

with_dif **(default:** *with_dif = .true.***)** Enables turbulent diffusion and surface fluxes. No parameterized turbulent mixing otherwise (also disables surface fluxes).

with_buoy **(default:** *with_buoy = .true.***)** Enables the computation of buoyancy. The flow is not buoyant otherwise.

with_mic **(default:** *with_mic = .true.***)** Enables the computation of microphysics tendencies. When false, hydrometeor quantities are still defined and initialized, but no microphysics is calculated.

with_adv **(default:** *with_adv = .true.***)** Enables advection of all prognostic quantities. No transport implemented otherwise.

with_rad **(default:** *with_rad = .true.***)** Enables radiation.

with_cor **(default:** *with_cor = .false.***)** Enables Coriolis forces.

with_lssub **(default:** *with_lssub = .false.***)** When true, large scale subsidence/upwelling is also computed for the total water mixing ratio and passive tracers. Only temperature is affected otherwise.

with_nudg **(default:** *with_nudg = .false.***)** Enables nudging of prognostic quantities across the domain (nudging strength is generally case dependent).

with_lssrc **(default:** *with_lssrc = .false.***)** Enables additional, case dependent sources and sinks (generally applied to passive tracers only).

with_lsadv **(default:** *with_lsadv = .false.***)** Enables large scale scalar advection. Large-scale tendencies are computed as altitude and case dependent sources and sinks added to the prognostic scalar equations.

diff_2d **(default:** *diff_2d = .false.***)** When true, turbulent diffusivity is calculated independently in the horizontal and vertical directions. Turbulent mixing is therefore different in the horizontal and vertical.

iradx **(default:** *iradx = 30.***)** Frequency (in seconds) at which the radiative transfer model is called. Radiative tendencies are held constant between two radiation steps.

rad_sw **(default:** *rad_sw = .true.***)** Enables the calculation of short-wave solar radiation. Only long-wave radiation is included otherwise.

rad_o3 **(default:** *rad_o3 = 0***)** This option controls the way ozone interacts with radiation. Three options possible: when 0, the default background ozone sounding is used, when 1, a used defined constant ozone sounding is used, when 2, ozone is fully interactive and treated as a passive tracer advected around. An ozone tracer can be included easily choosing *sca_set = 2* and *NSCAL = 1* (in *start*).

diff **(default:** *diff = 0.*) When different from 0, defines a constant turbulent diffusion coefficient, held constant in time and space.

pran **(default:** *pran = 0.4*) Turbulent Prandtl number, defined as the ratio between the turbulent viscosity and turbulent diffusivity (applied to momentum and scalars respectively). *pran* can be defined as a negative number in which case its absolute value is used but turbulent diffusion is only applied up to a predefined level (see below).

zdec **(default:** *zdec = 2000.*) Defines the altitude up to which turbulent mixed is applied (only if *pran* $<$ *0.*). Turbulent diffusion quickly tend to 0 above.

u0shift, v0shift **(default:** *u0shift = v0shift = 0.*) Geostrophic winds: used both to calculate Coriolis effects and lagrangian runs (the numerical domain is advected with velocities *u0shift*, *v0shift*).

Ddiv **(default:** *Ddiv = 0.*) Horizontal wind divergence (in seconds) used to calculate large-scale subsidence.

w_up **(default:** *w_up = 0.*) Large scale upwelling velocity used to imposed a background vertical velocity.

csp_cp **(default:** *csp_cp = 1*) When equal to 1, constant c_p and c_v values are used for air. When equal to 0, air c_p and c_v depend on the amount of vapor and condensed vapor.

4.2.8 *cm_mic*

micro_dif **(default:** *micro_dif = .true.*) When true, microphysical and aerosol variables (number and mass concentrations) are subject to turbulent diffusion. Turbulent mixing is otherwise ignored for these quantities.

lmicro **(default:** *lmicro = 0*) Microphysics level of complexity.

- 0: no microphysics.
- 1: warm phase microphysics only.
- 2: like 1 plus pristine ice crystals.
- 3: like 2 plus graupel and snow.
- 4: like 3 plus hail.

laero **(default:** *laero = -1*) Aerosol description level of complexity.

- -1: activation only, but aerosols are not removed. The number of aerosols is overall kept constant.
- 0: advection and aerosols are removed during activation.

- 1: like 0 plus wet scavenging and regeneration.
- 2: like 1 plus impaction scavenging.

lndrop **(default:** *lndrop = 0***)** When equal to 0, the cloud droplet number concentration is held constant and equal to *xn_ccn0*. No aerosol activation is thus needed.

lfreeze **(default:** *lfreeze = 0***)** Selection of freezing parameterization:

- 0: No freezing at all.
- 1: simple relaxation towards a constant ice crystal number concentration.
- 2: temperature dependent freezing following Cooper.
- 3: empirical parameterization following Diehl and Wurzler.

lrime **(default:** *lrime = .true.***)** Allows riming. No graupel can be formed when false.

lvent **(default:** *lvent = .true.***)** Enables ventilation effects during evaporation/sublimation.

dtcon **(default:** *dtcon = 0.5***)** Small time-step used to solve condensation/evaporation explicitly. A stable semi-analytic method is used otherwise when $dtcon \leq 0$.

auto_k **(default:** *auto_k = .false.***)** When true, auto-conversion is computed following the simple Kessler parameterization.

xauto **(default:** *xauto = 2.6e-10***)** Seifert-Beheng auto-conversion threshold. By default, this corresponds to the mass of a cloud droplet of radius 40 $/mu$m.

qauto **(default:** *qauto = 1.e-4***)** Kessler auto-conversion threshold (used with *auto_k = .true.*) expressed in kg/m^{-3}.

moments **(default:** *moments = 2***)** Takes values 1 or 2. Indicates the number of moments used in Seifert-Beheng microphysics.

ice_delay **(default:** *ice_delay = 0.***)** Delay (in seconds) between beginning of the simulation and first ice initiation.

ice_habit **(default:** *ice_habit = 'DEF'***)** A string of characters of length 3 indicating the default type of ice crystals considered by the model. Many choices possible including 'DEN' (dendrites), 'PLA' (hexagonal plates), 'BUL' (bullet rosettes), 'COL' (columnar crystals)... Each type corresponds to a predefined set of microphysical parameters, notably used to calculate mass-size and mass-fall speed relationships.

qthres **(default:** *qthres = 1.e-5***)** Threshold condensed water content used to identify cloud regions. For diagnostic purposes only.

xn_ccn0 **(default:** *xn_ccn0 = 50.e6*) Homogeneous CCN number concentration used for droplet activation (*lndrop = 1*) or homogeneous cloud droplet number concentration (*lndrop = 0*).

xn_in0 **(default:** *xn_in0 = 1000.*) Homogeneous IN number concentration used to define the background ice crystal concentration when *lfreeze = 1.*

xnc0_d **(default:** *xnc0_d = 1.e-7*) Geometrical diameter of CCN particles. Used for activation when *AEROSOL* is *FALSE* in *start*.

xnc0_s **(default:** *xnc0_s = 1.8*) Standard deviation of CCN size distribution (assumed to be lognormal). Used for activation when *AEROSOL* is *FALSE* in *start*.

xnc0_k **(default:** *xnc0_k = 0.7*) CCN hygroscopicity for activation when *AEROSOL* is *FALSE* in *start*.

4.2.9 *cm_aero*

nmode0 **(default:** *nmode0 = 1*) Number of aerosol modes.

reg_mode **(default:** *reg_mode = .false.*) An additional mode is defined that includes only regenerated particles. The initial number of particles in this mode is 0.

lkin **(default:** *lkin = .false.*) Enables the kinetic growth of newly activated cloud droplets using small time-steps of 0.2 s. The size of activated droplets is otherwise set to 1 μm by default.

aero_sfc_source **(default:** *aero_sfc_source = .false.*) Enables surface sources of aerosols.

aeroi(k)%nelem **(default:** *aeroi(k)%nelem = 0*) Number of elements constituting mode k. Varies between 1 and 4.

aeroi(k)%init%present **(default:** *aeroi(k)%init%present = (/.false., .false., .true., .false./)*) Indicates the presence or absence of predefined elements constituting aerosol mode k. In order, the array indicates the presence of sulfates, black carbons, sea salt and organics.

aeroi(k)%init%frac **(default:** *aeroi(k)%init%frac = (/0., 0., 1., 0./)*) Indicates the volume fraction of each element constituting aerosol mode k.

aeroi(k)%init%n0 **(default:** *aeroi(k)%init%n0 = 65e6*) Initial aerosol number concentration in mode k (units are kg m^{-3}).

aeroi(k)%size%rmean **(default:** *aeroi(k)%size%rmean = 0.0465*) Geometric mean radius of aerosols in mode k (in microns).

aeroi(k)%size%sigma **(default:** *aeroi(k)%size%sigma = 1.5*) Standard deviation of kth aerosol size distribution.

4.2.10 *cm_lag*

ilag **(default:** *ilag = 1.***)** Lagrangian particle output frequency expressed in seconds.

aerosol_lag **(default:** *aerosol_lag = .false.***)** Seeds each lagrangian parcel with aerosol particles.

nlag **(default:** *nlag = 0***)** When *aerosol_lag* is true, indicates the number of aerosol particles per parcel. Particles are drawn from a log-normal size distributions.

mu_lag, sigma_lag **(default:** *mu_lag = sigma_lag = 0.***)** Aerosol size distribution characteristics (mean geometric radius and standard deviation) for parcel aerosols.

compos_lag **(default:** *compos_lag = 'NaCl'***)** Composition of aerosol particles within each parcel (only NaCl currently avaialble).

res_lag **(default:** *res_lag = .true.***)** When true, for each particle exiting the numerical domain, a new particle is randomly created within the domain.

lag_init **(default:** *lag_init = 1***)** Method used to initialize the lagrangian particles:

- 1: Particles initialized within the boundary layer (below altitude *zbl*).
- 2: Particles initialized between altitudes *zl1* and *zl2*.
- 3: Particles initialized along a single line (2D) or within a single plane (3D) located at altitude *zbl*.

zl1, zl2 **(default:** *zl1 = zl2 = 0.***)** Altitudes defining the layer where aerosol particles are initialized (with *lag_init = 2*).

lag_mix **(default:** *lag_mix = .false.***)** Enables the impact of turbulent mixing on lagrangian particle trajectories.

lag_ord **(default:** *lag_ord = 1***)** Method of data interpolation at parcels location . When 0, a gaussian filter is applied centered at parcels locations. When 1, interpolation is performed using quadratic Lagrange polynomials (2nd order interpolation).

4.3 *out.nml*

Just like *cm.nml*, *out.nml* is a namelist file used to enable/disable output variables. It is organized in 4 "sections":

ou_in contains switches for all output variables included in the full 2D/3D output file and profiles

ou2d_in contains switches for surface or vertically integrated quantities (eg. surface rain rates or LWP)

oud_in contains switches for the tendencies included in the full 2D/3D output file and profiles

pro_in contains options related to the different horizontal profiles to output

Default values for all the parameters contained in *out.nml* are also defined in *INCLUDE/default.h*. A complete description of all *out.nml* options follows.

4.3.1 *out_in*

Flags under *out_in* allow the user to select the output variables that will be written in the complete 2D-3D output files, 2D slices, and one-dimensional profiles (see a description of all existing output files in section 5).

out_u, out_v, out_w **(default:** *out_u = out_v = out_w = .true.***)** Outputs winds in all 3 directions (variable names: U, V, W). *out_v* is automatically false in 2D simulations.

out_p **(default:** *out_p = .true.***)** Outputs the hydrostatic pressure and the pressure perturbation calculated using either the anelastic or fully compressible solver (2 variables: P_tot, P_pert).

out_pt **(default:** *out_pt = .true.***)** Outputs potential temperature (variable name: PT).

out_ptv **(default:** *out_ptv = .false.***)** Outputs the virtual potential temperature as well as virtual potential temperature perturbations (2 variables: PTv, PTv_pert).

out_t **(default:** *out_t = .false.***)** Outputs the absolute temperature temperature (variable name: T).

out_mse **(default:** *out_mse = .false.***)** Outputs the moist static energy (variable name: MSE), here defined as:

$$MSE = c_p\,(T - T_0) + L_v q_v + (L_v - L_s) \sum_{k=\{i,g,s\}} q_k + gz, \tag{151}$$

with summation performed over all ice categories, and $T_0 = 273.15$ K.

out_rho **(default:** *out_rho = .false.***)** Outputs the density and perturbation density (2 variables: RHO, RHO_pert).

out_sca **(default:** *out_sca = .false.***)** Outputs all scalars defined with *NSCAL* in *start* (*NSCAL* variables: SCAX, with X the scalar number).

out_qv, out_qt, out_ql **(default:** *out_qv = out_qt = out_ql = .false.***)** Outputs the water vapor, total water and liquid water mass mixing ratios (units are kg/kg, variable names: Qv, Qt, Ql).

out_qc, out_qr, out_qi, out_qg, out_qs, out_qh **(default:** *out_qc = out_qr = out_qi = out_qg = out_qs = out_qh*
Outputs the mass number concentrations of each hydrometeor category (units are kg/kg, variable names: Qc, Qr, Qi, Qg, Qs, Qh). Of course, the appropriate level of complexity must be set (*lmicro* in *cm.nml*).

out_nc, out_nr, out_ni, out_ng, out_ns, out_nh **(default:** *out_nc = out_nr = out_ni = out_ng = out_ns = out_*
Same as above with hydrometeor number concentrations (variable names: Nc, Nr, Ni, Ng, Ns, Nh).

out_dc, out_dr, out_di, out_dg, out_ds, out_dh **(default:** *out_dc = out_dr = out_di = out_dg = out_ds = out_d*
Mean hydrometeor diameters (variable names: Dc, Dr, Di, Dg, Ds, Dh). *spec_diag* must be true in *cm.nml*.

out_z **(default:** *out_z = .false.***)** Outputs modeled reflectivity (variable name: Ze). *spec_diag* must be true in *cm.nml*.

out_prec **(default:** *out_prec = .false.***)** Outputs liquid (if *lmicro > 0*) and ice (if *lmicro > 1*) precipitation fluxes defined as: $p_k = \rho q_k V_{p,k}$ (variable names: PREC_RAIN, PREC_ICE)

out_ccn, out_in **(default:** *out_ccn = out_in = .false.***)** Outputs the CCN and IN concentrations (variable names: CCN, IN). When *NUC_CNT* is *TRUE* in *start*, *out_in* enables the output of all nuclei number and mass concentrations (12 variables).

out_sat **(default:** *out_sat = .false.***)** Outputs relative humidities with respect to liquid and ice (2 variables: RH, RHi).

out_mf **(default:** *out_mf = .false.***)** Outputs in-cloud mass flux and cloud fraction (2 variables: MFL, CC). *spec_diag* must be true in *cm.nml*.

out_k **(default:** *out_k = .false.***)** Outputs the turbulent viscosity (variable name: Ksgs).

out_tke **(default:** *out_tke = .false.***)** Outputs multiple quantities related to turbulence: resolved turbulent kinetic energy, parameterized turbulent kinetic energy, buoyancy frequency $\mathcal{N}^2$ and shear S^2 (4 variables: TKE, TKE_sgs, N2, S2). *spec_diag* must be true in *cm.nml*.

out_flut **(default:** *out_flut = .false.***)** Output resolved vertical fluxes ($\Psi' w'$ with primes being fluctuations around a horizontal average) of various quantities: u if *out_u*, v if *out_v*, potential temperature if *out_pt*, virtual potential temperature if *out_ptv*, total water mixing ratio if *out_qt*, buoyancy if *out_buoy*, and scalars if *out_sca* (up to 6+NSCAL variables: UW_flux, VW_flux, WPT_flux, WQT_flux, WPTv_flux, WB_flux, WSCA1_flux). *spec_diag* must be true in *cm.nml*.

out_fsgs **(default:** *out_fsgs = .false.***)** Outputs parameterized vertical subgrid scale fluxes of all variables listed above (except scalars and virtual potential temperature, up to 5 variables: UWsgs_flux, VWsgs_flux, WPTsgs_flux, WQTsgs_flux, WBsgs_flux). *spec_diag* must be true in *cm.nml*.

out_var **(default:** *out_var = .false.***)** Outputs variances of potential temperature (if *out_pt*) and water vapor mixing ratio (if *out_qv*, up to 2 variables: Pt_var, Qv_var). *spec_diag* must be true in *cm.nml*.

out_wmom **(default:** *out_wmom = .false.***)** Outputs vertical velocity moments: variance and skewness (2 variables: W_var, W_skew). *spec_diag* must be true in *cm.nml*.

out_div **(default:** *out_div = .false.***)** Outputs flow divergence (variable name: Divergence). *spec_diag* must be true in *cm.nml*.

out_vort **(default:** *out_vort = .false.***)** Outputs the vorticity vector (3 components: Vorticity_x, Vorticity_y, Vorticity_z). *spec_diag* must be true in *cm.nml*.

out_grad **(default:** *out_grad = .false.***)** Outputs gradients of prognostic quantities: u, v, w, θ and q_t (at most 15 variables: grad_U_x, grad_U_y, grad_U_z,..., grad_PT_x, grad_PT_y, grad_PT_z, ...). *spec_diag* must be true in *cm.nml*.

out_buoy **(default:** *out_buoy = .false.***)** Outputs buoyancy (variable name: Buoy).

out_beff **(default:** *out_beff = .false.***)** Outputs effective buoyancy (variable name: Beff). *spec_diag* must be true in *cm.nml*.

out_dp **(default:** *out_dp = .false.***)** Outputs diagnosed non-hydrostatic pressure perturbation contributions (3 variables: P_b, P_d, P_nh). *spec_diag* must be true in *cm.nml*.

out_dtnet **(default:** *out_dtnet = .false.***)** Outputs radiative cooling/heating rates (variable name: DT_NET).

out_frad **(default:** *out_frad = .false.***)** Outputs radiative fluxes: net fluxes, short-wave fluxes and long-wave fluxes (3 variables: FRAD_net, FRAD_sw, FRAD_lw).

out_aero **(default:** *out_aero = .false.***)** Outputs interstitial aerosol number and mass concentrations, as well as activated aerosol mass for each defined mode (at most $3\times nmode$ variables: N_AEROX, M_AEROX, MA_AEROX, with X the mode number). Only when *AEROSOL* is *TRUE* in *start*.

out_ent **(default:** *out_ent = .false.***)** Outputs cumulus entrainment diagnostics (17 variables).

4.3.2 *ou2d_in*

All flags under *ou2d_in* control the output of two-dimensional fields (for a 3D simulation). These outputs are only available in 2D horizontal slices, that is with *no_out = .true.* and *nslicez > 0*.

out_lwp **(default:** *out_lwp = .false.***)** Enables the output of two-dimensional, vertically integrated condensed water content. This includes both liquid water path (LWP) and ice water path (IWP).

out_wvp **(default:** *out_wvp = .false.***)** Enables the output of vertically integrated water vapor content (water vapor path). A distinction is made between water vapor integrated over the entire depth of the domain, only in the boundary layer (below *zbl*) and in the free-troposphere (above *zbl*).

out_ints **(default:** *out_ints = .false.***)** Outputs vertically integrated passive tracer values. Mostly used in conjunction with *sca_set = 2* which defines a unique tracer mimicking ozone concentration in the atmosphere.

out_cmse **(default:** *out_cmse = .false.***)** Same as *out_wvp* above, but with moist static energy (MSE) instead of water vapor mixing ratio.

out_cmfl **(default:** *out_cmfl = .false.***)** Output of vertically integrated vertical momentum ρw.

out_cape **(default:** *out_cape = .false.***)** Output of two-dimensional values of CAPE, CIN, LCL and LNB computed within each model column. These quantities are calculated using the algorithm from G. Bryan.

out_ctop **(default:** *out_ctop = .false.***)** Output of two-dimensional cloud top and cloud base altitudes.

out_base **(default:** *out_base = .false.***)** Outputs two-dimensional cloud base mass, virtual potential temperature and precipitation fluxes.

out_cp **(default:** *out_cp = .false.***)** Outputs two-dimensional, cold pool specific variables. This includes cold pool intensity (based on low-level, vertically integrated virtual potential temperature anomaly), horizontal cold pool intensity fluxes and, if enabled, cold pool intensity diagnostics.

out_rrate **(default:** *out_rrate = .false.***)** Enables the output of the instantaneous and accumulated surface precipitation rates.

out_srad **(default:** *out_srad = .false.***)** Enables the output of net surface longwave and shortwave radiative fluxes.

out_olr **(default:** *out_olr = .false.***)** Enables the output of two-dimensional outgoing longwave radiation and net top of the atmosphere radiation fields.

out_sfl **(default:** *out_sfl = .false.***)** Enables the output of the full surface heat fluxes (sensible and latent).

4.3.3 *oud_in*

All flags under *oud_in* control the output of tendencies for major prognostic quantities. Each flag enables up to 10 output variables (generally named '_tend_1' to '_tend_10'). A complete description of all 10 quantities dumped for each *oud_in* flag is given in the file *README.diag* found in the main MIMICA repository. In the following, we only give a short overview of what each of these flags do.

out_diagu, out_diagv, out_diagw **(default:** *out_diagu = out_diagv = out_diagw = .false.***)** Outputs tendencies contributing to the three components of the momentum equations (advection, turbulent mixing, pressure gradient, buoyancy, coriolis, nudging...).

out_diagp **(default:** *out_diagp = .false.***)** Outputs tendencies contributing to advancing perturbation pressure. Note that when using the anelastic solver, *out_diagp* does not necessarily contain pressure tendencies, but rather various steps needed to solve the poisson pressure equation.

out_diagt **(default:** *out_diagt = .false.***)** Outputs tendencies contributing to advancing potential temperature (advection, turbulent mixing, radiation, phase changes, nudging...).

out_diagtv **(default:** *out_diagtv = .false.***)** Outputs tendencies contributing to advancing the virtual potential temperature. These are not computed directly by the various physics parameterizations, but are diagnosed from the potential temperature, total water and microphysical tendencies.

out_diagq **(default:** *out_diagq = .false.***)** Outputs tendencies contributing to advancing the total water content (advection, turbulent mixing, precipitation, nudging...).

out_diagk **(default:** *out_diagk = .false.***)** Outputs kinetic energy tendencies (purely diagnostic). Computed directly from U, V and W tendencies. Components considered in kinetic energy depend on the selected velocity diagnostic flags (*out_diagu, out_diagv, out_diagw*).

out_diagl, out_diagr, out_diagi **(default:** *out_diagl = out_diagr = out_diagi = .false.***)** Outputs tendencies contributing to advancing the cloud water, rain and cloud ice mass mixing ratios (advection, turbulent mixing, precipitation, phase changes, other microphysical contributions...).

out_diags **(default:** *out_diags = .false.***)** Outputs tendencies contributing to advancing passive tracers (advection, turbulent mixing, possible extra sources and sinks...).

out_diaga **(default:** *out_diaga = .false.***)** Outputs tendencies contributing to advancing aerosol number concentrations for each mode defined (advection, turbulent mixing, activation, precipitation scavenging, regeneration...).

out_micro **(default:** *out_micro = .false.***)** Outputs the detail of all microphysical tendencies contributing to each selected hydrometeor category (number and mass concentrations) depending on the microphysics level. For example, micro outputs separate contributions from condensation/evaporation, melting, auto-conversion, accretion, riming, freezing, precipitation...

4.3.4 *pro_in*

Flags under *pro_in* enable the creation of time-dependent, one-dimensional vertical profiles (2D outputs in the Z-time space). All output variables (as selected with the *oud_in* and *out_in* flags) are here averaged horizontally. The various *pro_in* flags allow the selection of various "filters" to perform horizontal averaging over grid points satisfying particular conditions (in particular cloudy or non-cloudy).

pro_tot **(default:** *pro_tot = .true.***)** Enables the creation of one-dimensional vertical profiles where all output variables are averaged horizontally across the entire domain. This is the default type of 1D profile outputs.

pro_env **(default:** *pro_env = .false.***)** Enables 1D vertical profiles like *pro_tot*, but all output quantities are here averaged over non-cloudy grid points only (defined with a condensed water content smaller than q_{thres}, *qthres* in *cm.nml*).

pro_cl **(default:** *pro_cl = .false.***)** Again, like *pro_tot*, but all output quantities are now averaged over cloudy grid points only (defined with a condensed water content larger than q_{thres}, *qthres* in *cm.nml*).

pro_cor **(default:** *pro_cor = .false.***)** Again, like *pro_tot*, but all output quantities are now averaged over cloud cores only (defined with a condensed water content larger than q_{thres} AND a vertical velocity w exceeding 1 m/s).

4.4 Initial soundings and idealized profiles

MIMICA can be initialized by either including idealized profiles of potential temperature θ, total water mixing ratio q_t, and wind velocities, or by specifying initial 1D soundings functions of altitude. Both initialization methods depend on external files stored in the *INCLUDE* directory: the former have a *.h* extension and contain short pieces of code written in fortran, while the latter are simple text files with a standard formatted column structure. The use of one or the other method depends on two *cm.nml* flags: the name of the simulation/configuration can be specified with *casename*, while *file_init* is the name of the input sounding file to be used by the model (the specified file must be present in *INCLUDE*). The choice of one method or the other is essentially case dependent: idealized case studies tend to rely on standard initial profiles provided as simple functions of altitude which can easily be implemented in include *.h* files, while more realistic configurations will typically require the specification of a realistic atmospheric sounding.

A number of predefined test cases and configurations (see section **??**) are available with the MIMICA code. Appropriate initial conditions are available for each of these templates, each file being also found in the *INCLUDE* directory. However, anyone interested in configuring its own simulation needs to create a new initial file. While using *.h* files requires modifying the fortran code, the simplest way to design new initial conditions is to create a new sounding file whose general structure is described below.

1D soundings files generally have a 5 or 6-column structure, with a single line header starting with #:

```
#           H         T(C)           RH            U            V
```

Each item on the line is a keyword indicating which variables are to be read in the file:

H indicates the altitude of each sounding level, **in meters above the surface**

T(C) indicates the absolute temperature **in C** at each vertical level. The temperature can alternatively be specified **in K** by specifying the keyword `T(K)` instead. An other possibility is to specify the potential temperature at each vertical level using the keyword `PT`

RH indicates the relative humidity **in %** at each vertical level. The water vapor mixing ratio can alternatively be specified **in g/kg** using the keyword `Q` instead

QL (optional) indicates the liquid water mixing ratio **in kg/kg** at each vertical level

U indicates the horizontal U velocity **in m/s** at each vertical level

V indicates the horizontal V velocity **in m/s** at each vertical level.

Note that `QL` is optional, while the specification of temperature and moisture profiles can be done in several ways depending on the selected keyword and is mandatory.

In addition to these two types of initial condition files, there also exists a *sounding_rad.dat* file, located in the *DATA* directory, and required to run MIMICA with interactive radiation (with *setenv RADIA TRUE* in *start*). The role of *sounding_rad.dat* is to describe the thermodynamic state of the atmospheric column extending between the top of the model domain (typically located between 1.5 and 45 km depending on the simulated configuration) and the top of the atmosphere. This is absolutely required by the radiative transfer model to calculate the exact solar insulation at the top of the model domain. Because the exact structure of the full atmosphere is generally not known at the precise location of the simulation, several default soundings are available (*.lay* files in *DATA*), representative of a standard atmosphere at various locations and various seasons. For example, *kmlw.dat* is representative of a mid-latitude winter atmosphere, while *ktrop.lay* is representative of the tropical atmosphere. In order to use one of the available profiles, it is enough to rename it *sounding_rad.dat*. Care must however be taken as the transition between the thermodynamic state at the top of the model domain and that of the standard atmosphere must be smooth enough for the radiation code to work successfully. Modifying the standard *sounding_rad.dat* file may be necessary in some instances to guarantee a smooth transition.

4.5 Recommended configuration

In the following, recommended sets of *start* and *cm.nml* option flags are described. The proposed choices should be stable and yield reasonable results in most situations

4.5.1 Numerics

For a default numerical configuration, it is recommended to set the following flags to TRUE in *start*: *ANELASTIC, CONSERVATIVE, ISENTROPIC*. With these, the model will employ the anelastic approximation (an elliptic pressure equation is solved with the FFT solver to satisfy continuity), all scalar equations are written in conservative form (for exact energy conservation), and the potential temperature is used as a proxy for thermal energy. The model has been extensively tested in this configuration and it should not result in any unexpected behavior.

Besides, two different stable scalar configurations can be selected.

- The first option is fast but perhaps less reliable since it involves solving for the non-linear momentum equations using a simple forward-in-time scheme: *RK* must then be set to *FALSE* in *start*, and *scal_adv* should be set to *lw* in *cm.nml*. This set of parameters will enable the default finite-volume scheme for scalar integration.

- The second option is more computationally expensive, but it is also associated with a higher-order of accuracy and it seems to behave well in most situations: *RK* should here be set to *TRUE* in *start*, with *scal_adv = 'quick'* in *cm.nml*. This configuration enables the method of lines for scalar advection, with a 2nd order Runge-Kutta time-stepping scheme.

In addition to these parameters, setting *mom_ord = 3* and *nsubp = 2* seems to provide reasonable accuracy and stability in any case. Moreover, it is always recommended to use flux limiters for scalar advection which can be done by setting *limit = .true.* in *cm.nml*, with a tolerance *lim_tol = 1.e-4 - 1.e-7* (higher values are faster, lower values guarantee that no spurious oscillations can develop).

As a final recommendation, stability seems to be preserved in most situations when *cfl_max = 0.48* in *cm.nml* (setting *cfl_min = 0.43* is then a reasonable choice). It is possible that higher *cfl_max* values may also yield stable solutions, but in general, only experimenting with different CFL numbers can confirm this. Another feature that may effect stability is the sponge layer added at the top of the domain to damp propagating waves. As a rule of thumb, it is generally recommended that the sponge layer corresponds to the upper fourth-third of the numerical domain (this sets *zdamp* in *cm.nml*), while its strength should be case dependent (*tdamp* can be large - ∼4-6 h - in deep domains, but small - ∼10-30 min - in shallow domains).

4.5.2 Physics

The set of recommended physical parameters will obviously be highly case dependent. It is however always useful to define sets of recommended options for each selected physical parameterization. Only the most important parameterizations are covered below: radiation, turbulent diffusion and microphysics. It should be noted in passing that selecting and using a particular parameterization generally involves setting the appropriate flag to *TRUE* in *start*, and/or turning the appropriate flag to *.true.* in *cm.nml* (for example *with_mic = .true.* to enable microphysics, *with_rad = .true.* to enable radiation, or *with_nudg = .true.* to enable nudging).

There is only a small number of parameters controlling the radiative transfer model. The most important one is *iradx* in *cm.nml* which sets the frequency at which radiation is updated. Calling radiation more frequently can be very slow, but calling it too seldom may introduce large errors. Because of that, it is generally difficult to recommend a precise value of *iradx*: depending on the case, and the clouds characteristic time scales, values between 1 min to 30 min may be chosen. Besides *iradx*, it is possible to change the surface albedo, surface emissivity, and even the presence of short-wave radiation via *cm.nml* flags.

Among the two turbulent diffusion parameterizations implemented in MIMICA, the simple Smagorinsky model will generally provide reasonable results in most situations (*TKE FALSE* in *start*) and has been extensively tested. Using the 1st order TKE model can be advantageous when higher accuracy is required to parameterize boundary-layer turbulence. Once a proper parameter has been chosen, one can control certain details of the parameterization with particular *cm.nml* flags. For example, it is possible to let the scalar turbulent diffusion coefficients go to 0 above a certain altitude *zdec*. This will be in effect when *pran < 0* (recall here that *pran*, the turbulent Prandtl number, corresponds to the ratio between turbulent viscosity for momentum and turbulent diffusion for scalars, and is generally ∼0.4). With *micro_dif*, it is also possible to disable turbulent diffusion for microphysical

quantities. Diffusing these quantities may indeed not be absolutely necessary, but is generally computationally expensive. Note finally that all parameters controlling surface fluxes in *cm.nml* (directly connected to turbulent diffusion) are very much case dependent and are not discussed here.

Last but not least, MIMICA offers a lot of flexibility when it comes to parameterizing microphysics, and this is associated with a lot of different microphysical flags and options. The first choice to be made is between the two available microphysics scheme: Seifert and Beheng (*SEIFERT TRUE* in *start*) or the one-moment, two-classes Grabowski scheme (*SEIFERT FALSE*). It is further possible to choose between saturation adjustment or explicit condensation/evaporation of cloud droplets with the *SAT_ADJ* flag. In general, the Seifert Beheng scheme is recommended, except if speed and efficiency are required (for example for long-term convective equilibrium simulations). Similarly, saturation adjustment is useful mostly when long time-steps, relatively coarse resolution cloud resolving simulations are performed, while explicit condensation/evaporation is preferable when simulating high-resolution boundary-layer stratiform clouds.

Once the proper microphysics scheme has been selected, one can control the desired level of details with flags such as *lndrop* (to enable activation or leave the cloud droplet number concentration constant), *lmicro* (to select the hydrometeor classes to be included) or *laero* (to select how aerosols are parameterized) in *cm.nml*. Certain parameters affecting the microphysics scheme should only be modified in specific situations (debugging or sensitivity tests). This includes *lvent* (ventilation, should be set to *.true.*), *ldrizz* (enables precipitation, set to *.true.*), or *auto_k* (enables Kessler autoconversion, set to *0*). Besides, *dtcon* controls the treatment of explicit condensation/evaporation and although it is recommended to leave the parameter to *0.* (pseudo-analytic method), small values on the order of *0.5* s may be employed for an explicit integration. It should be noted finally that while Seifert Beheng recommend to set *xauto*, the auto-conversion mass threshold, to *2.6e-10* (corresponding to a size threshold of 80 μm), we found that a smaller value of *6.55e-11* (corresponding to a size threshold of 50 μm) is more appropriate for most MIMICA applications.

5 MIMICA outputs

MIMICA offers a wide range of output possibilities, from instantaneous three-dimensional data fields, to one-dimensional time-dependent profiles. Overall, if one carefully manages the kind of output files written along with the list of dumped variables (see the full list of options available in *out.nml*), it is possible to perform very detailed result analyses while keeping the occupied disk space to a minimum.

To briefly summarize, MIMICA outputs can be classified into 4 main categories (from "big" to "small"):

1. the full output (in 2D or 3D) of all variables selected under *out_in* (in *out.nml*) at specified times,

2. two-dimensional slices extracted from three-dimensional data (with *MODEL_3D TRUE*) at specified positions along the X, Y and Z directions (*slicex_io, slicey_io, slicez_io* in *cm.nml*),

3. vertical profiles of horizontally averaged variables (in 2D or 3D) as a function of time, with the possibility to apply filters to average over, for example, cloudy or non-cloudy grid points only,

4. time-series of global variables (domain averages or domain mins and maxes) output at a high frequency.

Most of these output files are created in the netCDF V4 format. The only exception is the ASCII formatted time-series file (this allows a quick inspection of a simulation's good proceedings). All output files are created and stored in a local *OUTPUT* directory.

5.1 2D-3D complete outputs

To output the complete 2D or 3D data, *no_out* must be set to *.false.* in *cm.nml*. The list of variables dumped can be selected in *out.nml* and outputs will be written every *iax* seconds (set in *cm.nml*).

In the general case, a single file is created containing the complete data at all output times. The file's name can be specified under *file_output* in *cm.nml*. It should be noted however that, for very large simulations, it is possible to create one file per processor (this option is enabled by setting *PARALLEL_OUT* to *TRUE* in *start*). In this case, each file is named from the specified file name followed by the extension '_XXX' where XXX is the process number. The main advantage of this option is that it doesn't require collecting all data on a single processor before the output. This then saves both CPU time (MPI communications can be slow) and memory.

5.2 2D Slices (3D simulations only)

Since outputting the complete data fields of a three-dimensional run can quickly become overwhelming, it can be advantageous and sufficient to only extract two-dimensional slices in the X-Y, X-Z or Y-Z planes at specified positions. To do so, *no_out* must be set to *.true.* in *cm.nml* and the number and position of each individual slice must be specified using *nslicex, nslicey, nslicez* and *slicex_io, slicey_io, slicez_io* respectively. No slice is created by default, *nslicex, nslicey, nslicez* being all set to 0, and a maximum of 15 slices along each direction can be created. The slice positions *slicex_io, slicey_io, slicez_io* are specified in meters as lists of up to 15 real numbers separated by comas. The output frequency is again set by *iax* in *cm.nml*.

Slice files are simply named *slice_x_XXXX.nc, slice_y_XXXX.nc, slice_z_XXXX.nc* for slices taken in X, Y and Z respectively, with 'XXXX' denoting the position (in meters) along the considered axis.

All slices will contain by default all variables selected under *out_in* in *out.nml*. In addition, Z slices can contain surface specific as well as vertically integrated outputs (depending on X and Y) such as sensible and latent heat fluxes (SHF and LHF), liquid water path (LWP), outgoing longwave radiation (OLR) or surface precipitation rates. These quantities will be output if a Z slice is created (at any given altitude, not necessarily at $z = 0$), and if *out_surf* is *.true.* in *cm.nml*. The dumped variables can then be selected in *out.nml* under *ou2d_in*.

Besides the local slices described above, the option *out_yav* in *cm.nml* enables the creation of a single slice in the X-Z plane where all output variables are averaged along the Y dimension. This output may be useful for "long channel" configurations where the domain breadth in Y is very small compared to the length in X.

5.3 1D profiles

Time-dependent, one-dimensional profiles are output if (at the very least) *pro_tot* is *.true.* in *out.nml*. This is the default option such that 1D profiles are created by default by MIMICA. 1D variables (the same as in the complete outputs and slices) are written in profile files every *ipro* seconds (in *cm.nml*).

1D profiles are calculated by averaging horizontally the output variables at each model level. By default (*pro_tot*) horizontally averages are performed over all grid points. However, it is possible to restrict the horizontal averaging to specific grid points in each plane using filters. The three most important filters can be selected with *pro_env, pro_cl, pro_cor* set to *.true.* in *out.nml*: the first filter generates horizontal averaged computed over non-cloudy grid points only, the second calculates averages over all cloudy grid points only (i.e. points with condensed water content exceeding q_{thres} in *cm.nml*), and the third calculates averages over cloud core grid points only (with condensed water content larger than q_{thres} AND vertical velocity larger than 1 m/s).

Profile files corresponding to the *pro_tot, pro_env, pro_cl, pro_cor* keywords are respectively named: *profiles_tot.nc, profiles_env.nc, profiles_cl.nc, profiles_cor.nc*.

Among the family of one-dimensional outputs, although this is not a vertical profile, MIMICA offers the possibility to create hovmöller plots. This can be done by selecting *out_hov = .true.* in *cm.nml*, and the corresponding output file is called *hovmoller.nc*. Hovmöller plots are generally useful to visualize the temporal evolution of one-dimensional surface properties (in particular surface precipitation rates or outgoing longwave radiation). In MIMICA, the hovmoller outputs are created by averaging the 2D surface variables selected under *ou2d_in* along the Y dimension (as in *slice_yav.nc*). The dimensions of the *hovmoller.nc* files are thus X-time.

5.4 Time series

The formatted time-series files *T_S* are used to plot the evolution of important global variables as a function of time (generally with a high frequency, specified with *its* in *cm.nml*). These global quantities include in particular domain averages, domain minima and domain maxima.

Unlike the other output files, the list of variables dumped in *T_S* is fixed. There exists however three predefined sets of time-series variables which can be selected using *ts_out* in *cm.nml*:

'stratus': with 'stratus', global variables relevant to stratocumulus clouds are dumped, including for example the mean cloud top entrainment rate and the mean LWP and IWP.

'cumulus': with 'cumulus', global variables relevant to convective clouds are dumped, including mean cloud base mass flux, the maximum cloud top height and mean surface fluxes.

'deep': with 'deep', global variables relevant to deep tropical convection are dumped, including the mean tropopause altitude, or mean tropopause temperature.

In addition to these case dependent global outputs, certain variables will be present in *T_S* in any case. This includes for example the mean cloud base and cloud top altitudes, mean surface precipitation rates, maximum vertical velocity in the domain, and some radiation properties such as mean OLR and TOA. Of course, some of these outputs are only available if the appropriate physics package (in *cm.nml*) and output flag (in *out.nml*) are enabled.

5.5 Restart files

The last file that you will find in your local *OUTPUT* directory is a *restart.dat* file (although its name can be changed with *file_rest* in *cm.nml*). This non-formatted binary file is created (in fact, overwritten unless the flag *all_rest* is *.true.* in *cm.nml*) after each *ires* seconds of simulated time, and it contains all the 2D or 3D data necessary to restart MIMICA (which is done by setting *new_run* = *.false.* in *cm.nml*). This includes in particular the complete prognostic variable fields (potential temperature, wind speeds, total water content, prognostic microphysical quantities...) as well as the corresponding standard profiles (that is typically the initial state).

6 Templates and examples

6.1 Available templates

The *templates* repository present in your main MIMICA repository contains a series of predefined, ready-to-use test cases that MIMICA users can use for testing, debugging, or as examples to start brand new simulations. Each template sub-directory contains at least three files that can be readily used to compile and run the corresponding test case: the first file defines the initial state from which the simulation is started (either a .h or a sounding file), the second is a *start* file that can be used to produce a standard mimica executable for the case, and the third one is a *cm.nml* file containing default option values to run the case in a standard configuration (users are invited to test various options for individual templates).

Each template is intended to represent either a highly idealized and simplified test case focusing on a very specific aspect of the model, or a more comprehensive case previously used for research purposes. Idealized test cases (such as rising warm bubbles, cold density current, inertia gravity waves or elementary 1D and 2D advection scenarii) are typically designed for testing and debugging some of the core elements of the model, either numerical schemes or physical parameterizations. The second category of templates is obviously intended to serve as a basis for future research (but they may of course be used also to ensure the reproducibility of reference results).

The following gives an overview of all the 34 templates currently available in the current MIMICA version.

1D_ADV: Simple one-dimensional advection of a step function with fixed, homogeneous wind speed. Uses the Anelastic approximation by default with the QUICK advection scheme with a 100 m resolution, but can be used to test various other scalar advection schemes. All physical parameterizations are disabled.

1D_DIFF: One-dimensional diffusion of a step function. No advection is included and the diffusion coefficient is fixed (0.01). All physical parameterizations are disabled.

2D_ADV: Two-dimensional advection of a gaussian blob. The gaussian scalar field is advected along the diagonal using the QUICK scheme by default, and with fixed u and w velocities (u=w). All physical parameterizations are disabled.

2D_NL_ADV: Two-dimensional non-linear advection-diffusion problem (no scalar solved). Initial condition consists in a square shaped homogeneous U and W velocity perturbation.

2D_DIFF: Two-dimensional diffusion of a gaussian blob with a fixed diffusion coefficient (0.01). Again, advection and all other physical parameterizations are disabled.

ASCOS: Semi-idealized case based on ASCOS data. The available configuration includes all relevant physical parameterizations (in particular microphysics and radiation), and uses the anelastic solver with limited Lax-Wendroff scalar advection.

BUBBLE_ANELASTIC: Warm bubble rising in an homogeneous atmosphere with open boundaries. The case is driven by buoyancy only. The Anelastic assumption is used with flux-limited Lax-Wendroff advection for scalars. Resolution is approximately 10 m.

BUBBLE_REFINED: Same as BUBBLE_ANELASTIC but with non-constant grid spacing in the vertical.

BUBBLE_3D: Same as BUBBLE_ANELASTIC but in 3D, with a small amount of diffusion (although not necessary), and a 15 m resolution.

BUBBLE_COMPRESSIBLE: Same as BUBBLE_ANELASTIC but using the fully compressible solver.

BUBBLE_PRECIP: A rising warm bubble such as BUBBLE_ANELASTIC, but condensation of liquid water at saturation is here allowed.

BUBBLE_PRECIP_mse: Same as BUBBLE_PRECIP except that moist static energy is used instead of potential temperature as a prognostic measure of internal energy.

CONE: Two dimensional advection of a cone-shaped scalar field in a rotating flow (fixed velocity field). The case employs the anelastic solver and a MUSCL advection scheme.

DENSITY_CURRENT: An initially cold bubble sinks, hits the surface, and propagates laterally to develop a high density current. The case uses a 200 m resolution grid, the Anelastic approximation, QUICK advection, and a fixed diffusion coefficient.

DENSITY_CURRENT_MPI: Same as above but in parallel (4 processors).

DENSITY_CURRENT_comp: Same as DENSITY_CURRENT but using the fully compressible solver.

DRY_PBL: A typical test case where a dry planetary boundary layer is simulated at high resolution (60 m in the horizontal, 25 m in the vertical). The MUSCL advection scheme is employed, but no particular physical parameterization in used except for turbulent diffusion (Smagorinsky).

DYCOMS: The configuration is similar to the DYCOMS-II model intercomparison study. The case is representative of marine boundary layers capped with stratocumulus clouds.

HYDRO_ADJUST_1D: Hydrostatic adjustment with a fixed heat source in the middle of the domain. Pseudo 1D domain with open lateral boundaries and vertical periodicity. Non-conservative compressible equations are solved. Despite the fact that advection is enabled, the case is mostly used as a benchmark for testing the compressible (or anelastic) solver.

HYDRO_ADJUST_2D: A case similar (but not exactly) to HYDRO_ADJUST_2D but in two dimensions. The original configuration is compressible, periodic with Lax-Wendroff advection. Possibility to switch to anelastic.

HYDRO_ADJUST_2D_mse: Anelastic version of the above. Also uses moist static energy as the conserved energy variable.

IGW: Inertia gravity waves. 2D domain with fixed static stability and an initial temperature perturbation at the center. The MUSCL advection scheme is used with 1km horizontal resolution. No physical parameterization is used.

ISDAC: Based on ISDAC F31 intercomparison study: a mixed-phase Arctic stratus cloud case. The default configuration uses a 50 m resolution grid with MUSCL advection. Physical parameterizations include turbulent mixing, prescribed radiation, and the two-moment microphysics scheme (cloud water, rain and cloud ice only). Possibility to use various options for the modelling of ice phase initiation. Input files are here for a 2D simulation.

KMSc: An idealized stratocumulus case using a 2D kinematic framework (prescribed velocity field representative of a single updraft and downdraft, and homogeneous initial conditions). Uses the Seifert-Beheng microphysics with detailed activation. Possibility to use complex cloud-aerosol interactions (recommended to test the aerosol module).

MIDLATITUDE: An idealized midlatitude summertime convection case. The grid is 200x200 km2 with 200 m resolution. Parameterizations include the full Seifert-Beheng microphysics scheme, Smagorinsky turbulent diffusion, but no radiation. The time-dependent evolution of sensible and Latent heat fluxes is prescribed.

RCE: A radiative-convective equilibrium case (here in 2D). Anelastic approximation with conserved potential temperature. Horizontal resolution is set to 2 km, radiative cooling is fixed to -2 K/d everywhere, sea surface temperature is set to 300 K, and the simple Grabowski microphysics is employed by default. This is just a template for more elaborate RCE simulations.

RCEMIP: Setup is similar to the near-global, long-channel configuration defined for the RCEMIP intercomparison study. The domain is 6000 km long and 400 km wide with a 3 km horizontal resolution. The vertical grid is defined in the *new_grid.dat* file. SST is set to 300 K, and the simple Grabowski microphysics is employed. Radiation is interactive and uses a prescribed ozone profile.

RICO: Configuration based to the idealized RICO shallow cumulus case. Model resolution is 25 m along each direction, and QUICK advection is employed. Large-scale advection tendencies and nudging are prescribed. Physical parameterizations are used for Coriolis forces, turbulent diffusion and microphysics (Seifert and Beheng, warm phase only).

SINE: One-dimensional advection of a sine wave. Uses flux limited Lax-Wendroff advection by default and no diffusion. Does 20 revolutions.

SPIRAL: Advection of a scalar gaussian blob in a rotating flow (prescribed time dependent velocities). Anelastic equations are solved with limited Lax-Wendroff advection scheme, and split advection.

SPIRAL_HO: Same as above but with high-order numerics (PPM scalar advection and Runge-Kutta time integration).

SQUALL_LINE: A two-dimensional idealized squall line from the WRF test case suite. Uses the anelastic approximation and aturation adjustment with the full Seifert-Beheng microphysics (for autoconversion and precipitation). The case is two-dimensional with 1 km resolution in the horizontal and 250 m resolution in the vertical.

SQUALL_LINE_comp: Mostly similar to the SQUALL_LINE case above, but without saturation adjustment and using the fully compressible core.

TG_VORTEX: Taylor Green vortex: the flow field should be balanced by the pressure field calculated from the pressure solver at all time. Fully compressible case. No scalar is advected here, the focus is on representing the pressure-velocity balance.

TRANSITION: A shallow to deep convection transition case based on TRMM-LBA experiment. The case is 2D, with time varying surface fluxes, refined grid in the vertical, and several case specific I/O.

References

[1] S. Berthet, M. Leriche, J.-P. Pinty, J. Cuesta, and G. Pigeon. Scavenging of aerosol particles by rain in a cloud resolving model. *Atmospheric Research*, 96:325–336, 2010.

[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] P.J. DeMott, A.J. Prenni, X. Liu, S.M. Kreidenweis, M.D. Petters, C.H. Twohy, M.S. Richardson, T. Eidhammer, and D.C. Rogers. Predicting global atmospheric ice nuclei distributions and their impacts on climate. *Proceedings of the National Academy of Sciences*, 107:11217–11222, 2010.

[4] K. Diehl and S. Wurzler. Heterogeneous drop freezing in the immersion mode: Model calculations considering soluble and insoluble particles in the drops. *Journal of the Atmospheric Sciences*, 61:2063–2072, 2004.

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

[6] M. Ovchinnikov et al. Intercomparison of large-eddy simulations of arctic mixed-phase clouds: Importance of ice size distribution assumptions. *Journal of Advances in Modeling Earth Systems*, 6:223–248, 2014.

[7] S.J. Ghan et al. Droplet nucleation: Physically-based parameterizations and comparative evaluation. *Journal of Advances in Modeling Earth Systems*, 3:M10001, 2011.

[8] M. Frigo and S.G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[9] Q. Fu and K.-N. Liou. On the correlated k-distribution method for radiative-transfer in nonhomogeneous atmospheres. *Journal of the Atmospheric Sciences*, 49:2139–2156, 1992.

[10] Q. Fu, K.-N. Liou, M.C. Cribb, T.P. Charlock, and A. Grossman. Multiple scattering parameterization in thermal infrared radiative transfer. *Journal of Atmospheric Sciences*, 54:2799–2812, 1997.

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

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

[13] W.W. Grabowski. Toward cloud resolving modeling of large-scale tropical circulations: A simple cloud microphysics parameterisation. *Journal of the Atmospheric Sciences*, 55:3283–3298, 1998.

[14] Y. Gu, J. Farrara, K.-N. Liou, and C.R. Mechoso. Parameterization of cloud-radiation processes in the ucla general circulation model. *Journal of Climate*, 16:3357–3370, 2003.

[15] V.I. Khvorostyanov and J.A. Curry. A simple analytical model of aerosol properties with account for hygroscopic growth: 1. equilibrium size spectra and ccn activity spectra. *Journal of Geophysical Research*, 104:2163– 2174, 1999.

[16] V.I. Khvorostyanov and J.A. Curry. Aerosol size spectra and ccn activity spectra: Reconciling the lognormal, algebraic, and power laws. *Journal of Geophysical Research*, 111:D12202, 2006.

[17] J.B. Klemp, W.C. Skamarock, and J. Dudhia. Conservative split-explicit time integration methods for the compressible nonhydrostatic equations. *Monthly Weather Review*, 135:2897–2913, 2007.

[18] B.P. Leonard. The ultimate conservative difference scheme applied to unsteady one-dimensional advection. *Computer Methods in Applied Mechanics and Engineering*, 88:17–74, 1991.

[19] R.J. Leveque. *Finite volume methods for hyperbolic problems*. Cambridge University Press, 2002.

[20] F. B. Lipps and R. S. Helmer. A scale analysis of deep moist convection and some related numerical calculations. *Journal of the Atmospheric Sciences*, 39(10):2192–2210, 1982.

[21] C.-H. Moeng and J.C. Wyngaard. Evaluation of turbulent transport and dissipation closures in second-order modeling. *Journal of the Atmospheric Sciences*, 45:2311–2330, 1989.

[22] Y. Morinishi, T.S. Lund, O.V. Vasilyev, and P. Moin. Fully conservative high order finite difference schemes for incompressible flow. *Journal of Computational Physics*, 143:90–124, 1998.

[23] 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.

[24] A. Nenes and J.H. Seinfeld. Parameterization of cloud droplet formation in global climatemodels. *Journal of Geophysical Research*, 108:D14, 2003.

[25] M.D. Petters and S.M. Kreidenweis. A single parameter representation of hygroscopic growth and cloud condensation nucleus activity. *Atmospheric Chemistry and Physics*, 7:1961–1971, 2007.

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

[27] R.K. Rew and G.P. Davis. NetCDF: An interface for scientific data access. *IEEE Computer Graphics and Applications*, 10(4):76–82, 1990.

[28] M. Satoh. Conservative scheme for a compressible nonhydrostatic model with moist processes. *Monthly Weather Review*, 131:1033–1050, 2003.

[29] J. Savre, A. M. L. Ekman, and G. Svensson. Technical note: Introduction to mimica, a large-eddy simulation solver for cloudy planetary boundary layers. *J. Adv. Model. Earth Syst.*, 6(3):630–649, 2014.

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

[31] 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.

[32] J. Seinfeld and S. Pandis. *Atmospheric chemistry and physics: from air pollution to climate change*. John Wiley, New York, 1998.

[33] W.C. Skamarock and J.B. Klemp. The stability of time-split numerical methods for the hydrostatic and nonhydrostatic elastic equations. *Monthly Weather Review*, 120:2109–2127, 1992.

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

[35] 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.

[36] R.B. Stull. *An introduction to boundary layer meteorology*. Kluwer Academic Publishers, 1988.

[37] B. van Leer. Towards the ultimate conservative difference scheme, iv. a new approach to numerical convection. *Journal of Computational Physics*, 23:276–299, 1977.

[38] B. van Leer. Towards the ultimate conservative difference scheme, v. a second order sequel to godunov's method. *Journal of Computational Physics*, 32:101–136, 1979.