Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2019-111
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 21 May 2019

Geoscientific Model Development Discussions
Open Access

Development of "Physical Parametrizations with PYthon" (PPPY, version 1.1), and its usage to reduce the time-step dependency in the ICE microphysical scheme

Sébastien Riette[1]

[1]CNRM, Université de Toulouse, Météo-France, CNRS, Toulouse, France

Correspondence: Sébastien RIETTE (sebastien.riette@meteo.fr)

Abstract. To help develop and compare physical parametrizations such as those found in a numerical weather or climate model, a new tool was developed. The tool provides a framework to plug external parametrizations, run them in an offline mode, save the results and plot diagnostics. With the help of this tool, the origin of the time-step dependency of the microphysical scheme used in the Météo-France small scale operational numerical weather model was identified. The sources of dependency lied in some process formulations and in the algorithm used to allow the competition between the different processes. Some corrections have been introduced and their efficiency was checked with the tool. This usage illustrates how the tool can be used in 0D or 1D mode, with schemes coming from different models and with different time-advance methods to produce different kinds of plots..

1 Introduction

A weather or climate numerical model contains several parametrizations (e.g. turbulence, convection ...) that interact together and with the dynamical core. When a parametrization is being developed or debugged, these interactions can hide and/or amplify a tested modification. When the goal is to compare two parametrizations hosted by different models, theses interactions distort the comparison as the other model components can be very different (dynamical core, discretization and other parametrizations). To circumvent these effects, one can reduce the interactions by unplugging other parametrizations (ideal cases, aqua-planet experiments), by reducing the problem size (2D vertical simulations, single column model)...

The choice of the comparison strategy depends on the intended goal: a full 3D-model is able to represent all the interactions whereas simplified models represent only a subset of these interactions. However, even the single column model (one of the simplest configuration) is not always sufficient to separate the impact of the different parametrizations (see, for example, point 8 of Ghan et al. (2000) conclusion). Hence, a simpler framework could be useful; this can be toy models in which only one parametrization is plugged.

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2019-111
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 21 May 2019

An example of a toy model used to compare microphysical schemes is given by Shipway and Hill (2012) with the Kinematic Driver (KiD) model. This is also the approach taken here to develop the PPPY tool in which we can plug existing parametrizations from different models, deal with the simulations, compare the outputs and plot the results. The tool described here has some common points with the KiD model but is able to deal with any parametrization (not only microphysics), integrates the graphical part and is very flexible by the use of the python language to control the execution flow (for example running and comparing hundreds of different configurations is not an issue, extending the tool by incorporating diagnostics should be simple, . . .). The KiD model, for its part, allows advection and, hence, lies between the tool described here and a Single Column Model.

A difference was observed on 3D simulations with the Meso-NH model (Lac et al., 2018) when the time step was changed. Because the impact of the time step was the most important on rain accumulation and also when prognostic hail was activated, the microphysical scheme (Pinty and Jabouille, 1998, hereafter referenced as ICE) was suspected to be the first responsible. To assess this dependency, simulations in a 0D mode, using only the microphysical core processes, are performed using the PPPY tool. To test all the processes, an initial state involving all the hydrometeors was chosen. Initial mixing ratios were quite important ($10\ \mathrm{g\,kg^{-1}}$ for vapor, no hail and $1\ \mathrm{g\,kg^{-1}}$ for the other hydrometeors) and hail was activated (even if the illustrations here were made with simulations without hail to simplify the plots); the initial temperature was set to 270 K.

When several hydrometeors are mixed in a model cell without exchange with the exterior, the microphysical processes tend to an equilibrium state. And, this final state must not depend on the time step used. Moreover, when two (or more) simulations run with different time steps are compared, they should have the same results for common output times. In Fig. 1, we can see that the final state depends on the time step used (between 1 s and 60 s) for water vapor, rain and temperature. Furthermore, the chaotic appearance of this plot is a signature of the time-step dependency. Without this drawback, all curves of a same color normally would follow the same time evolution. For example, after 60 s of simulation (which corresponds to the order of magnitude of the time-step length used in the Météo-France small scale operational numerical weather model, AROME (Application of Research to Operations at Mesoscale, Seity et al., 2011), which share the same physical package with the Meso-NH model), a great uncertainty exists on the hydrometeors presence; depending on the time step used, rain, graupel and snow can exist (with very significant content) or not.

In the COnsortium for Small-scale MOdeling (COSMO) model, Barrett et al. (2019) also observed a time-step dependency on rain and hail accumulations that they traced back mainly to the interaction between the dynamics and the physics of the model, and, to a lesser extent, to some microphysical processes. The example shown in this paper demonstrates that a significant part of the time-step dependency can also be explained by the microphysical scheme itself.

The time-step dependency of the microphysical scheme is not specific to the ICE parametrization. The dependency is also observed (Fig. 2) with the Liquid Ice Multiple Aerosols (LIMA) scheme (Vié et al., 2016) (a quasi two-moment microphysical scheme in development in the Meso-NH and AROME models). And, such a dependency was also observed in the Integrated Forecasting System (IFS) model (Forbes, 2018), and were related to the formulation of the warm-rain processes. Moreover, some microphysical schemes of the Weather Research and Forecasting (WRF) Model (version 3.9.1.1) have been plugged in the tool and also exhibit time-step dependency, as shown in Fig 3 for the Eta (Ferrier) scheme (Rogers et al., 2001, panel a), the

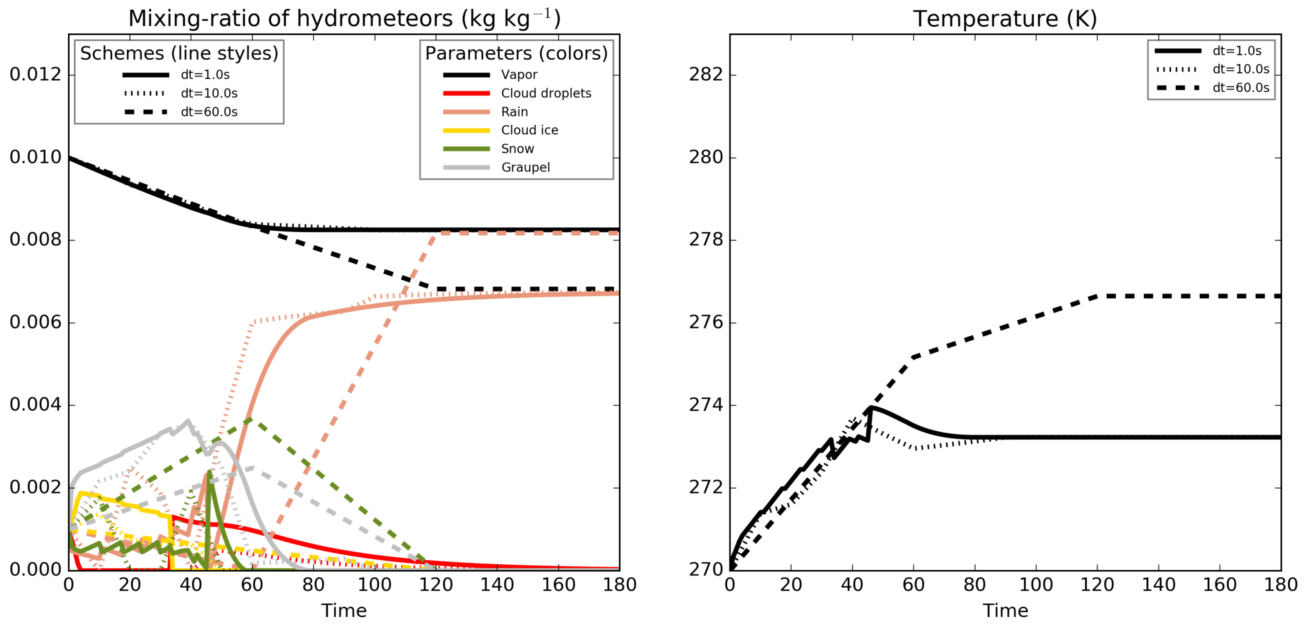

Figure 1. Time evolution of the mixing ratio of the different hydrometeors (in kg/kg, left panel) and of the temperature (in K, right panel). The simulations were performed using time steps between 1 s and 60 s.

Milbrandt–Yau Double Moment scheme (Milbrandt and Yau, 2005a, b, panel b), the Morrison 2–moment scheme (Morrison et al., 2009, panel c), the Hebrew University of Jerusalem Spectral Bin Microphysical (HUJI SBM) scheme (Khain et al., 2004, panel d), the Thompson scheme (Thompson et al., 2008, panel e) and the WRF Single–moment 6–class (WSM6) scheme (Hong and Lim, 2006, panel f). These example also show how the PPPY tool can be useful to exhibit some behaviors of a scheme (time-step dependency, oscillations, water conservation, . . .) independently of the other model components.

Section 2 describes the technical choices and provides an overview of what can be done with the tool. Some examples of usage are given in Sect. 3 before the conclusion.

2 Functionalities and technical aspect

A documentation is available in the tool package. To complement this documentation, this section gives some details on how to add a parametrization and describes the functionalities related to the parametrizations and to the comparisons.

2.1 Technical aspects

Python was chosen because it allows to make plots (through the matplotlib module for this tool), deal with common file formats and interact with compiled code (this feature makes possible to use the original source code of the parametrization). The tool

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2019-111
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 21 May 2019

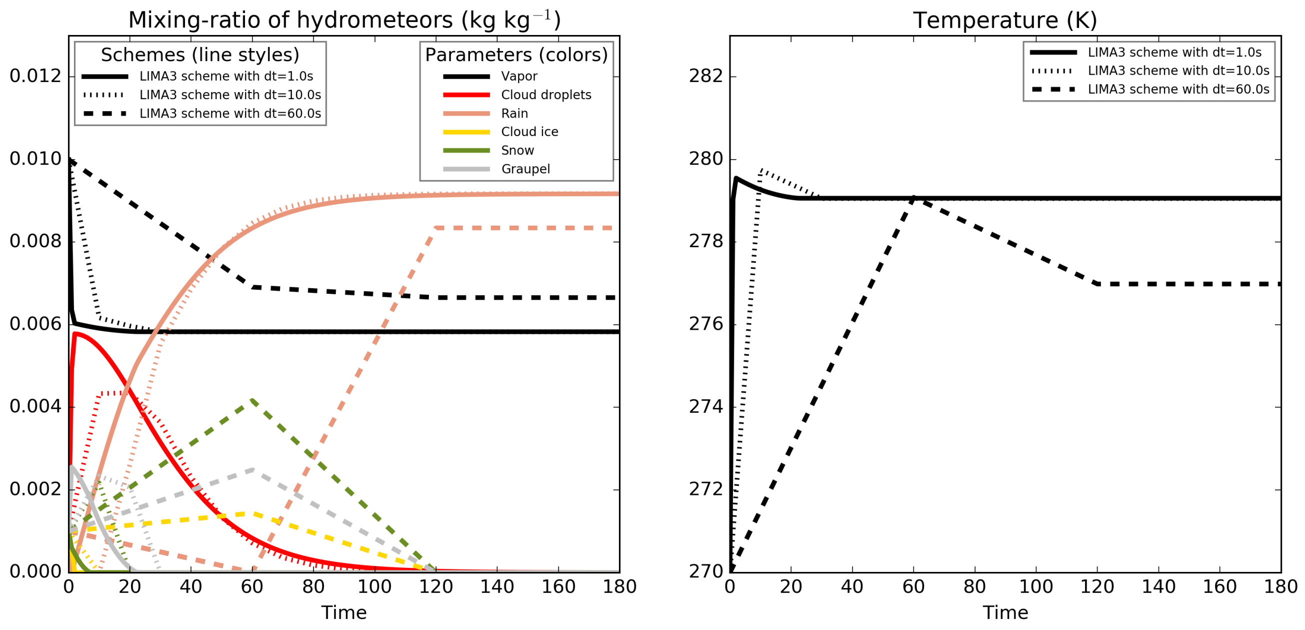

Figure 2. Same as Fig. 1 but using the LIMA scheme.

consists in a python package is written to drive the simulations (initialization, calling of the Fortran routines, saving of the results, plotting results).

Even with simulations limited to only one point, the computational time can be large when very small time steps are used; to avoid recomputing simulations already run, the script saves the results of the different simulations in HDF5 files (using the h5py module which is the required).

The general design of the tool is as follow (see Fig. 4): each parametrization code (built from FORTRAN source code for example) is associated to a python object (made from the provided PPPY class), and then, these different PPPY python objects are used by a PPPYComp object which performs the comparison. These different objects are described in the following subsections.

2.1.1 The low level part of the parametrization: source code and compilation

The most tricky part comes from the interfacing between python and the parametrization. This part is quite technical but is important as the main difficulty in using this tool with a new parametrization lies in this interfacing task.

If the parametrization was written using python (like the box-Lagrangian scheme used in Sect. 3.2) the interfacing would be straightforward but numerical weather and climate models often use Fortran and a tool is needed to do the interfacing. There are several ways to accomplish this task; this paper is not the place to do a review of the different ways but here are listed the two that were used at some stages of the development process. This two ways are still usable even if the examples provided with the tool use only the second one. The f2py utility can be used; it helps to build a shared library suitable to be imported and used

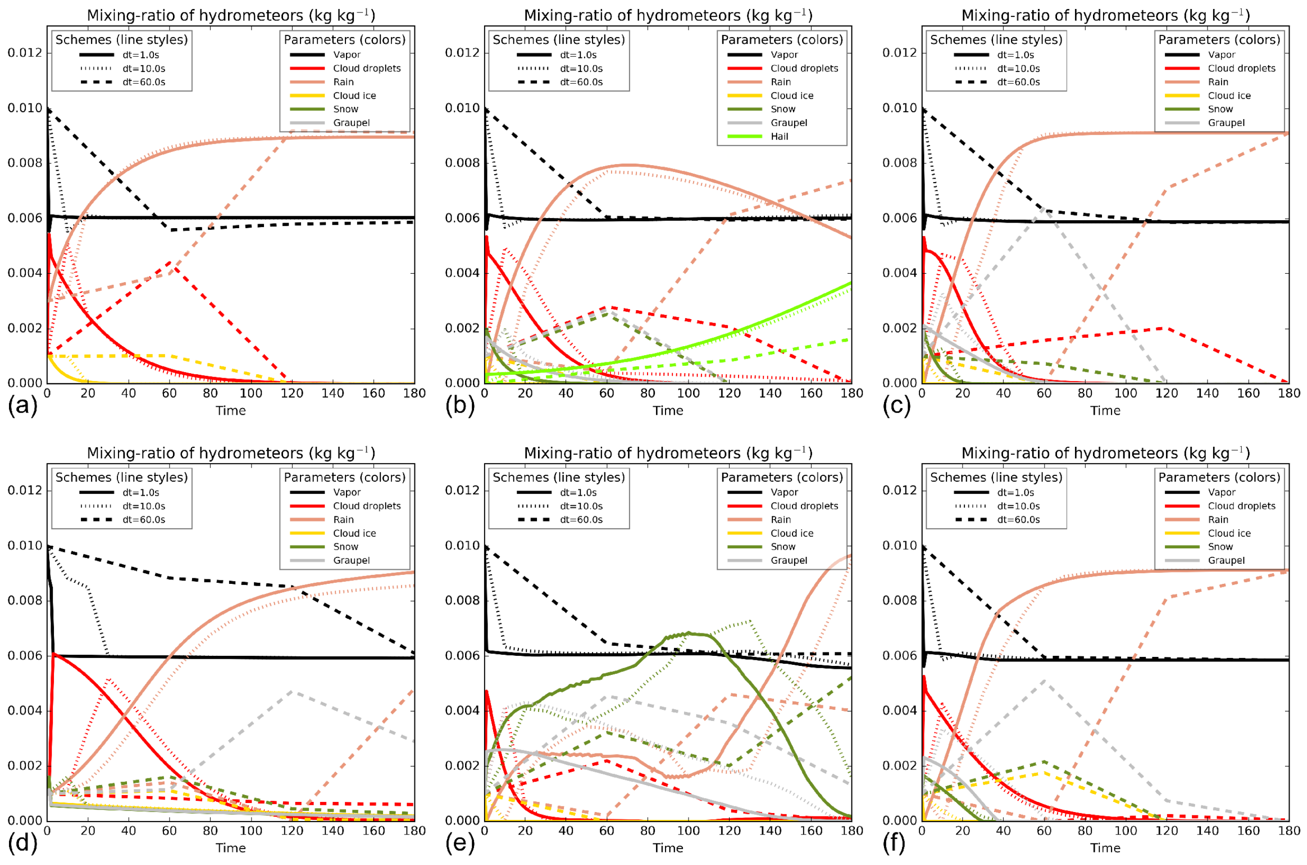

Figure 3. Same as left panel of Fig. 1 but using some schemes of the WRF model (see text).

from python but it led to problems maybe specific to our environment or source code (dependency to the exact python version, argument order not preserved...). The second one is home made; it aims at using the ctypes module (normally reserved for C codes) directly over a shared library built from the Fortran source code. For each Fortran subroutine (or function) to use, the interfacing consists in writing a python function with the same name as the Fortran subroutine and that returns the signature of it (that is the interface of the subroutine written in a specific way). A decorator (a python object that modifies the behavior of a function) is provided to convert this signature function into a function that actually calls the Fortran subroutine (or function) and returns the result. For the potential C-written parametrizations, the interfacing must directly use the ctypes module.

Because the compilation can be a complex process (that can involve scripts that modify, on the fly, source codes), it could be difficult to isolate and compile, outside of the box, the source code needed for a given parametrization. To reduce the difficulty, the different provided examples follow this procedure:

– Modification of the model compilation script and/or Makefile file to include the option to build a position-independent code, suitable for dynamic linking ("-fPIC" option),

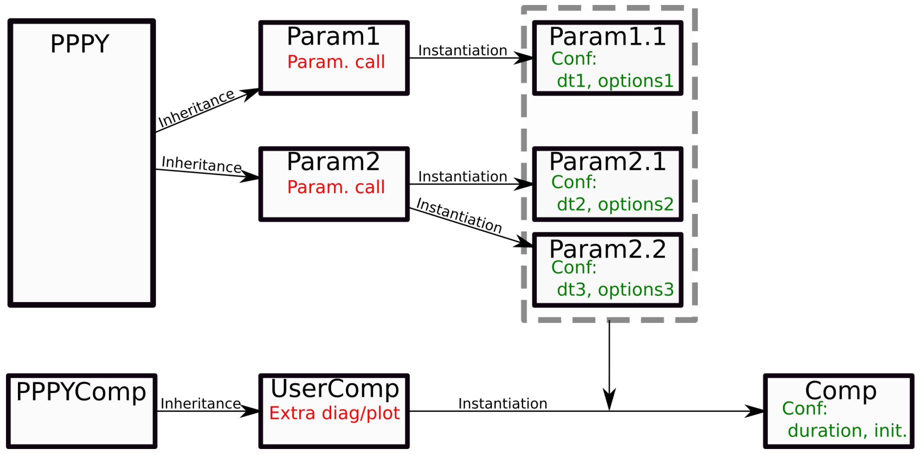

Figure 4. Tool organisation diagram. The 6 top boxes represent the parametrization objects while the 3 lower boxes are the comparison objects.

– Normal compilation of the model,

– Use of the different object codes and/or static libraries built during the normal compilation step to build a shared library with the different entry points needed to use the parametrization.

Because the shared object will be opened using the ctypes module (that normally handles C shared library), the Fortran subroutines and functions that will be used from the python script must not make use of some Fortran specific argument properties (e.g. assumed shape arrays, optional, assumed length character strings) that not exist in an identical way in C. If it is not the case, a wrapper must be written in Fortran that receives arguments declared with fixed lengths and calls the original subroutine.

At this stage the remaining difficulty is to identify the different routines that must be called to perform the parametrization initialization and execution.

2.1.2 The high level part of the parametrization: the PPPY python object

Once the compilation part is completed, a python object must be created to manipulate the compiled library. An abstract class (PPPY) is provided for this purpose and must be used, by inheritance, to build a class specific to the parametrization to use. The abstract class already contains everything needed to perform the time advance and the saving of results and have placeholders for the requested standardized methods that must be implemented. In these methods, new variables specific to the current scheme can be defined, conversions can be performed (for example to change the temperature variable from potential temperature to true temperature), modifications in memory representation can be done (all variables are 64 bit in the python script but can be converted into 32 bit for instance) and this is the place to make use of the shared library and run the actual parametrization code. All options concerning this parametrization can be made available through the initialization method.

Then, this class is instantiated (to get the Param1 and Param2 classes of Fig. 4) by providing some options (e.g. time step, options specific to the parametrization).

The tool was developed and tested to compare microphysical schemes but the code was made very general so it is possible to plug other parametrizations. As a consequence, the list of the physical quantities to monitor and compare is not hard written. Therefore, before trying to compare several parametrizations of a same process, one have to define the list of the variables to monitor with their units. For instance, temperature can be represented as potential temperature or true temperature in Celsius or Kelvin. The same quantities must be monitored by all the schemes even if, internally, each scheme uses its own set of variables. Conversions to and from these quantities are done around the parametrization call.

2.1.3 Comparison python object

The parametrizations, which are instances created from the PPPY class (as stated above), are intended to be used by a python object to perform a comparison. This comparison object (instantiated from the PPPYComp class, to get the Comp object of Fig. 4) is characterized by the list of parametrizations to use, the simulation length and the initial state of the simulations. The object is responsible to run the different parametrizations (isolated from each other), to compute diagnostics (that can be added by creating a custom comparison class by inheritance) and to plot the results (plots methods can also be added).

2.2 Tool functionalities

The tool allows to compare several parametrizations. The different parametrizations can differ by the underlying code or can differentiate themselves by the choice of the parameters controlling the scheme. The tool also enables the comparison between two identical parametrizations using different time steps or different time-advance methods. Two time-advance methods exist:

"step-by-step" like a true simulation, the output is computed from the output of the previous time step (Fig. 5).

"one-step" the output (at all output times) is computed by a direct integration from the initial state (Fig. 6).

The tool was developed in such a way as to allow the comparison of any parametrizations, and not only microphysical schemes. The set of variables followed by the tool is not limited to predefined ones; the user can add any variable of any dimensions. Moreover the tool is able to use schemes from different models (interfacing with AROME, Meso-NH and WRF has been done).

Currently two plot kinds are available but other can be written by extending the tool. The already existing plot methods can draw results for 0D and 1D simulations (problem size can be reduced -slicing- for plotting; this allows to run the simulations once on a variety of initial conditions and plot the results for only one point or profile). The y-axis is used to represent the variable intensity of the 0D simulations or the different points of the 1D simulations. The two plot kinds differ in the x-axis which is used to represent the time (with different plots superimposed for the different schemes, like in the examples of Sect. 3.1) or the different schemes (with different plots superimposed for the different output times, like in some examples of Sect. 3.2). In this context, the different schemes can be different parametrizations and/or a same parametrization using different

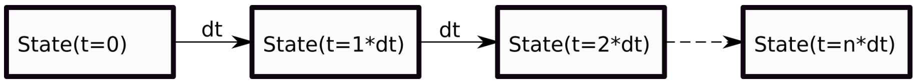

Figure 5. Time advance for a step-by-step simulation.

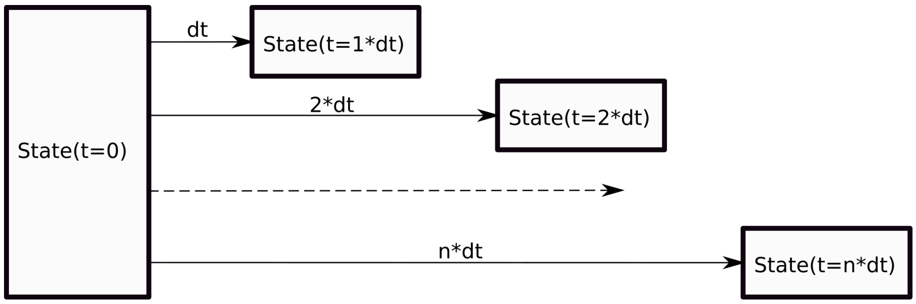

Figure 6. Time advance for a one-step simulation.

options (constants, configuration options or time step choice); this allows to perform sensitivity tests to one parameter. The figures that illustrate the examples shown in this paper have been directly produced by the tool.

3 Application to microphysical parametrizations

The ICE microphysical scheme is divided in three parts: a statistical adjustment to the saturation (to balance cloud water and ice with the vapor, according to the temperature), the core microphysical processes (collections, riming, vapor deposition, evaporation, . . .) and the sedimentation. Each of these three parts can contain sources of time-step dependency.

The adjustment to saturation modifies the temperature and hence modifies also the saturation point and then the cloud content. This feedback could be a source of time-step dependency but it was checked (not shown) that the saturation adjustment used reaches an equilibrium very quickly; the impact of a second iteration can hardly be seen. However, the cloud ice fraction (which is the ice content divided by the total -ice and liquid- content) is function of the temperature (for temperature above 0 °C, the cloud is liquid, for temperature under -20 °C, the cloud is icy and the cloud ice fraction is linearly interpolated between these two points) and, then, a consumption of one of these two species in the core microphysical processes implies a consumption of the other specie during the following saturation adjustment (to keep the cloud ice fraction consistent with the temperature). This mechanism leads to a time-step dependency.

In this section, two examples of the tool usage are shown. The first one deals with the time-step dependency due to the processes of the microphysical scheme (without adjustment and without sedimentation); and the second one is a comparison of several sedimentation schemes.

3.1 Time-step dependency in the microphysical scheme

The final goal was to suppress the time-step dependency on the simulations shown in Fig. 1. To achieve this result, a new set of simulations was performed using much smaller time steps (between 0.001 s and 1 s). The idea was to look for a convergence between the simulations when the time step decreases. In Fig. 7, the final values are much more consistent but some oscillations are still visible on the time evolutions and values between 40 s and 60 s of integration are still very uncertain. It was necessary to use very small time steps to obtain numerical oscillations smaller than the physical variations.

The purpose of the paper is not to give an extended review of all the modifications that were needed to suppress the time-step dependency. I will just list rapidly the most important ones:

- heat budgets must be computed when the feedback on temperature can stop the process. For example, when the temperature is positive, a specie cannot melt more than the quantity that implies a change in the temperature sign. The lack of heat budgets explain a large part of the time-step dependency observed with the small time steps;
- in the original version of ICE, the snow content rimed by cloud droplets (to produce graupel) was computed as a statistical adjustment: the process gave the mass of snow to convert into graupel, then this mass was divided by the time step. And, the mass of transformed snow did not take into account the quantity of cloud water involved. These two characteristics were at the origin of a time-step dependency. The process was modified using the Murakami (1990) approach based on the comparison between the effective cloud droplets collection and the mass of water needed to transform low density snow into high density graupel;
- the graupel growth mode choice was updated. It is now more continuous and, hence, less time-step dependent. The graupel growth mode has an impact on the collection efficiency of icy species (snow and cloud ice) with the graupel and can lead to significant differences in the collection rates;
- the water shedding (cloud droplets becoming rain drops when collected and not frozen by the graupel) was activated only with negative temperature whereas the process must be active as long as the graupel exists (when the graupel has melt into rain, rain drops actually collect cloud droplets; the process must be continuous during the graupel melting);
- several modifications have been carried out on the processes involving the hail category as a prognostic field: the processes dealing with hail are now completely based on those dealing with the graupel category (to insure consistency even if this did not produce time-step dependency) and the conversion rate of graupel into hail is now computed from the wet growth rate of graupel and not from the total content of graupel (this was the main reason of the time-step dependency on the hail category).

These modifications try to suppress the time-step dependency present inside each of the microphysical processes. Another source of dependency lies in the interactions between these processes. One process must take into account that a given specie can be consumed or produced, in the same time, by another process. The modifications listed above were sufficient to suppress

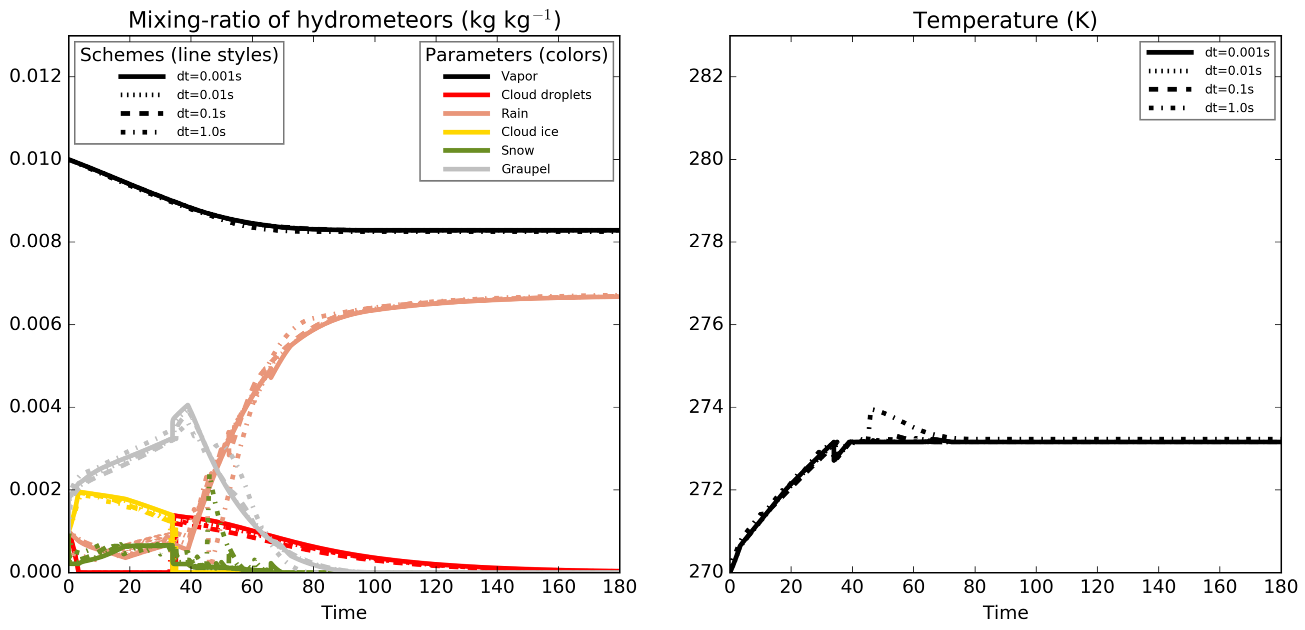

Figure 7. Same as Fig. 1 but using smaller time steps between 0.001 s and 1 s.

or, at least, limit the time-step dependency until time steps around 10 s (not shown). To use time steps greater, some kind of splitting was needed to reduce the effective time step used in the microphysical scheme to allow interactions between processes.

The modified scheme allows two splitting methods: a classical time-splitting method that uses a fixed sub-time-step and a "mixing-ratio-splitting" method that computes, at each iteration, the sub-time-step to use in order to not have a single mixing ratio change exceeding a given threshold. The second method has the advantage to adapt the number of iterations to the intensity of the microphysical processes. When little happens, only one iteration is performed; on the contrary, when the exchanges are intense, several iterations are performed.

With all these modifications, including a mixing-ratio-splitting, the different simulations shown in Fig. 7 and Fig. 1 are now perfectly indistinguishable in Fig. 9 and Fig. 8 (the curves are superimposed at every common output times).

To produce these figures, the threshold used in the mixing-ratio-splitting (1.E-5 g kg^{-1}) was small and could induce a substantial additional cost on a real simulation. This value was chosen to illustrate the scheme behavior but a value of 5.E-5 g kg^{-1} seems to be more acceptable (empirical value obtained in comparing cost and time-step dependency in a 2D simulation) for operations.

The 0D tool was very useful to identify and correct the processes involved in the time-step dependency of the ICE microphysical scheme. It would have been certainly possible to achieve the same result with another method or tool but this method was convenient (0D simulations are very rapid and a unique tool performs the simulations, compares the outputs and plots the comparison) and allowed to completely isolate the processes of interest from the other parts of the model (dynamic with transport, other physical parametrizations, sedimentation and adjustment).

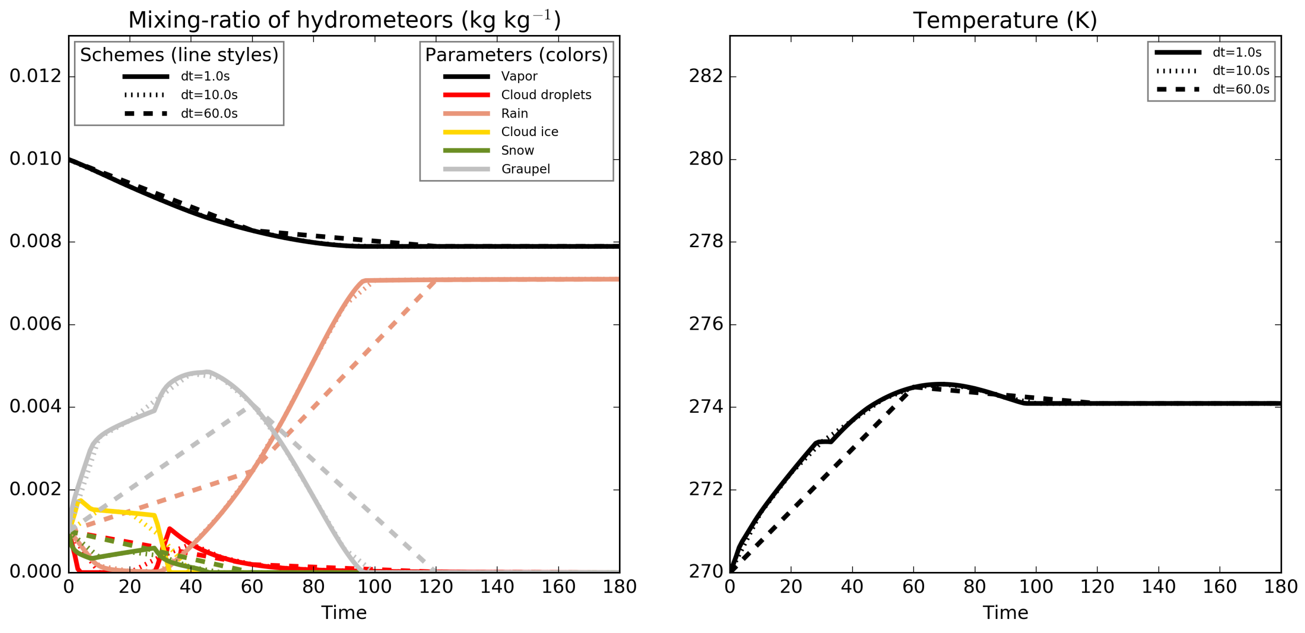

Figure 8. Same as Fig 1 but using the new version of the microphysical scheme

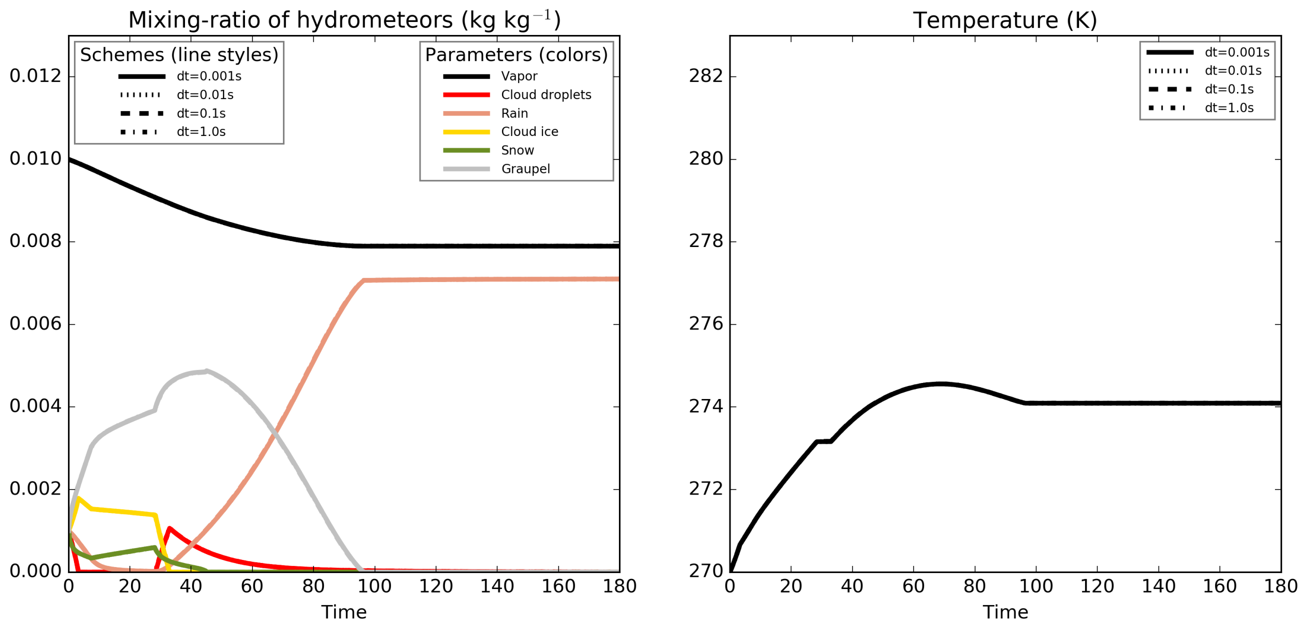

Figure 9. Same as Fig. 7 but using the new version of the microphysical scheme (all four curves are superimposed)

3.2 Sedimentation schemes

A similar time-step sensitivity test is done regarding the sedimentation scheme used in the model (all other parametrizations, including the microphysical processes, were turned off). Two schemes are available: the operational one (Bouteloup et al., 2010) (BSB2010 hereafter) which is a statistical scheme and the other one is an Eulerian scheme. In order to remain stable, the Eulerian scheme uses a time-splitting technique with an upstream differencing scheme. The internal time step[1] used for this computation is determined from the Courant-Friedrichs-Lewy (CFL) stability criterion based on a maximum fall velocity of 40 $\mathrm{m\,s^{-1}}$ if hail is allowed or 10 $\mathrm{m\,s^{-1}}$ otherwise (then, the same internal time step is used for all the species). A part of the difficulty for these schemes to resolve accurately the sedimentation process comes from the hypothesis that the terminal fall speed is directly linked to the mean content (more the content is important, more the fall is rapid).

A vertical profile was initialized with a rain mixing-ratio of 0.1 $\mathrm{g\,kg^{-1}}$ in one cell at 1400 m above ground level; grid levels were 10 m thick. In order to build a reference solution independent from the time step, a box-Lagrangian scheme (based on Kato, 1995) is used with the one-step time-advance method (see Sect. 2.2 for a description of the time-advance method). Because the model schemes use a particle size distribution, the first reference simulation we can build is by dividing the total content into bins and apply the sedimentation on each bin (as for Milbrandt and McTaggart-Cowan, 2010). The reference time evolution is then shown in the upper panel of Fig. 10 (this reference is computed using only 25 bins to allow the reader to identify the trajectories of each bin; when more bins are used the time evolution gets smoother). The bigger drops reach the ground after around 200 s whereas the smaller ones haven't reach the ground after 1500 s. After 400 s of simulation (dashed vertical line on the plot), one third of the rain is expected to be on ground and the remaining part spread in the column. However, the model schemes compute the mass-weighted bulk terminal fall velocity and apply this velocity to the entire content. With this hypothesis, the awaited time evolution is given by the lower panel of Fig. 10. In this case, all the rain content is expected to follow the same trajectory and to be near the ground after 400 s of simulation. The first reference simulation has a more physical behavior and is able to reproduce the size-sorting effect but this result cannot be reached by the one-moment schemes used in this study. Then, the bulk simulation is taken as the reference simulation to compare the model schemes over 400 s long simulations. According to the initial mixing-ratio, to the parameters used in the sedimentation scheme and to the hypothesis of a mass-weighted bulk terminal fall velocity, the rain is expected to fall with a 3.3 $\mathrm{m\,s^{-1}}$ speed, leading to a fall of 1320 m during a 400 s period. It should be noted that with a grid spacing of 10 m, the unit value for the CFL number is reached with a time step of around 3 s.

The top panel of Fig. 11 shows the resulting profile for different time steps (for time steps between 0.1 s to 60 s in steps of 0.1 s) using the BSB2010 scheme (this is a not a time evolution, all profiles are the result of a 400 s long integration). The time-step dependency is obvious and it must be noted that for time steps longer than 30 s a part of the water has reached the ground. Longer is the time step, more this part is important. It reaches around 11% of the total water for the 60 s time step.

[1]the term "internal time step" is reserved, in this paper, for the time step used internally in the scheme to perform the time splitting. It is different from the (external) time step used for the scheme integration

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2019-111
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 21 May 2019

For time steps between 0.1 s and 3 s, longer is the time step, more the bottom of the precipitation envelope gets closer to the ground. This behavior is certainly common to all the schemes that do not diagnose or predict the vertical heterogeneity. Indeed, with a small time step, only a tiny fraction of the grid content is able to fall in the lower grid level during one integration step. At the next time step, it is hypothesized that the remaining water is uniformly distributed on the vertical in the cell; this implies 1) a computed fall speed slightly smaller because the mean content is weaker, and 2) a longer time needed to make fall the grid content because the part of this content which get closer to the bottom of the grid had been redistributed on the whole cell (upward diffusion). Smaller is the CFL number, more the fall speed is reduced. A small CFL number must also lead to a diffusion downward because of the same redistribution mechanism applied to the weak content that just crossed the level boundary and is distributed over the whole grid at the next time step. But, because the content is weak, the fall speed is small and these cells are quite easily caught up by the surrounding cell with a greater content (thus falling with a a greater speed).

With a 3 s time step, the algorithm of the scheme induces a fall without diffusion which leads to the correct result (assuming that the fall speed is uniform for all the drops which is an hypothesis of the studied schemes because the mass-weighted terminal fall velocity is affected to all the drops). The P1 (defined in BSB2010 as the proportion of "rain which leaves the layer during the time step") and P2 ("Among all the drops that enter the layer under consideration (from above) during the time step, P2 is the proportion of those which also leave the layer (by the bottom) during the time step.") terms of BSB2010 evaluate to, respectively, one and zero; the rain crosses exactly one grid cell by time step.

With slightly larger time steps, P1 is still one and P2 remains small. With P2 small, diffusion downward remains small, the peak content is still on the top level of the precipitation envelope. During the simulation, the level with the peak content falls of one level, catches up the content which has fallen quicker (during the preceding time step) and so, the diffusion stays weak during the rain fall. And, by consequences, P1 remains equal to one for the highest level with rain. But because the fall of the peak content occurs at a rate of one grid level by time step, the fall speed gets smaller when the time step increases because less iterations are needed to reach the 400 s integration duration.

With much larger time steps, P1 is still one at the beginning and P2 is larger. With a greater value of P2, at the first time step, a significant rain content crosses several levels. This induces that the top level of the precipitation envelope may not have a sufficient content, at next time step, to insure a CFL number larger to one. This had two consequences: the top level of the envelope falls slower and the rain contained in the top level does not catch up the rain already fallen lower. This leads to an upward diffusion and to a reduction of the mean velocity at all levels because rain content is diluted on the vertical.

The BSB2010 scheme behaves differently regarding the CFL number. The artifact seen in the figure illustrates the short-coming of the scheme but cannot be so important in a true model simulation. In a real case with advection, non-constant grid spacing and microphysical sources and sinks, it is not possible to keep the CFL number constant and equal to one during the entire fall.

For the Eulerian scheme (lower panel of Fig. 11), the time-splitting technique used with a very small internal time step of 0.25 s (value obtained considering a maximum fall speed of 40 $\mathrm{ms^{-1}}$) leads to do approximately the same computation whatever is the time step of the simulation, greater than this value. Therefore, there is no time-step dependency in the result. However, the very small internal time step induces, as for the BSB2010 scheme with small values of the CFL number, an upward diffusion and a reduced fall speed. Moreover, the Eulerian scheme has an increased numerical cost due to the very small internal time step used.

None of both schemes is correct; both diffuse the precipitation and have a too small fall speed. Moreover, the BSB2010 scheme exhibits a time-step dependency whereas the Eulerian scheme is costly. The common problems to both schemes are the result of two mechanisms. Firstly, for small CFL values the schemes are penalized by the fact that the water content is supposed to be, at each time step, uniformly spread on the cell; if the time step is not long enough to put all the water in the level under, at the next time step, the remaining content is artificially put higher. Secondly, because we consider one-moment microphysics, when a content is split across two cells (by a first iteration of the algorithm), at the next time step the mean content over each of both cells is weaker and the mean fall speed is reduced. With a two-moment scheme, the content which had fallen in the lower level would be associated to bigger drops and would keep a greater fall speed than with the one-moment scheme.

Because none of the scheme can reproduce the right fall speed, a compromise could be found by optimizing the Eulerian scheme in order to obtain a scheme without time-step dependency and with a reasonable cost. The idea is to not use a fixed value for the internal time step. In the optimized version, at each iteration in the time splitting, the maximum time of integration is computed from the maximum CFL number on each column and for each specie. The integration is done for one internal time step and then the following iteration is done the same way until the advance reaches the whole time step.

The result is shown in Fig. 12 for different values of maximum CFL number allowed. The top panel is obtained with a maximum CFL number of one. The time-step dependency is quite large because, for integer CFL numbers, the fall occurs without diffusion. For CFL numbers slightly superior to an integer value, the first internal iterations are done with a CFL number of one (without diffusion) and the last internal iteration produces a great content surrounding a cell with a weak content that could be easily caught up during the next time step even if mean speed was reduced. For CFL numbers slightly inferior to an integer value, the last internal iteration produces a weak content surrounding a great content which lead to a diffusion and a reduced fall speed.

A maximum CFL value of one cannot be obtained everywhere and every time in a true simulation and, to reduce the risk of instability, a maximum CFL number allowed can be set. The panel in the middle is obtained with a maximum CFL number value of 0.1. The resulting figure is almost identical to the one obtained in Fig. 11 for which the small internal time step induced a small CFL number. The lower panel is for a larger maximum CFL number (0.8). For small time steps inducing a CFL number inferior to this maximum value, a time-step dependency can be seen. For large time-step values, the computation leads to the same results whatever is the time step because, in this case, the internal time step used in the time-splitting algorithm is not limited by the external time step. In contrast with the original version or the modified version with a small value for the maximum CFL number, the simulations are less diffusive and are able to reproduce a peak value (in the bottom of the precipitation envelope). The resulting fall speed is still reduced (after the 400 s integration duration, all the water content should be near the ground according to the hypothesis done) but slightly better than those produced by the other versions of the Eulerian scheme.

In order to test further the impact of the algorithm on the sedimentation results, the box-Lagrangian scheme (used previously to build the reference results) is used in a simulation mode (using the step-by-step time-advance method). Top panel of Fig. 13

shows the resulting profiles after the 400 s long simulation using the bulk approach. The result is noisy, time-step dependent and there is still no rain on the ground. The noise could be certainly reduce by imposing a speed continuity between the different layers of the model (following the idea of Henry Juang and Hong (2010) for example) but this simulation demonstrates that among the different numerical schemes used in this paper (Eulerian, statistical and Lagrangian) none are able to reproduce the reference simulation. This is mainly due to the fact that after each time-step, if CFL is not one, the rain content is split over two grid cells and, then, the content is reduced at the next time step and this induces a diffusion on the vertical and a decreasing fall speed. We could try to reduce this feature by using an hybrid scheme (as used in Morrison, 2012) in conjunction with the box-Lagrangian scheme. At each time step, the content is divided in bins, then, each bin falls using the box-Lagrangian scheme. At the end of the time step, the total content at each model layer is computed and used by the following iteration. The simulations are done using 500 bins (lower panel of Fig. 13). This scheme allows to make fall the bigger drops quicker. While this approach reduces the noise, some time step dependency remains and the scheme is still not able to make fall the precipitation quick enough to reproduce the reference simulation (and this scheme does not fulfill the requirement of a mass-weighted bulk terminal fall velocity used to build the reference). Nevertheless, it should be noted that this result is quite similar to the result obtained with the new version of the Eulerian scheme (lower panel of Fig. 12). It seems that no algorithm choice can reproduce the reference simulation due to the diffusion appearing during the fall that leads to a reduced speed. It seems difficult to find a solution to improve the simulations without prognosing an other moment of the distribution (this other moment must be advected and not only diagnosed for the sedimentation process). Milbrandt and McTaggart-Cowan (2010) also came to the same conclusion and gave hints on the choice of the best moments to use and on the corrections that can be applied to a two-moment scheme to reduce the sedimentation error.

On the one hand, the algorithm choice could be important because this choice modifies the effective fall speed and the relative position of precipitation on ground with respect to the cloud which generated it, through the horizontal advection. But, on the other hand, none of the schemes tested in this study is able to reproduce the reference simulation. Moreover, with the mixing induced by the dynamic and the turbulence, and with the interaction with the other microphysical processes, this choice has little impact on the resulting simulation of a real 3D case (not shown). It is believed that no scheme could perform drastically better with the one-moment hypothesis.

In the future, this study could be to the two-moment sedimentation scheme used by LIMA.

4 Conclusions

In this paper, a new tool designed to allow the comparison of Physical Parametrizations with PYthon (PPPY) independently of all other model components was described technically and functionally. Its ability to use FORTRAN-compiled library from different models, as well as python based parametrizations has been shown and used through two examples. The tool has been successfully used (in a 0D mode) to identify the sources of the time-step dependency which was present in the microphysical scheme in use in the AROME and Meso-NH models. Some solutions have been proposed to correct the scheme and have been tested with the tool. Then, the sedimentation schemes have been plugged and compared (in a 1D mode) to a reference box-

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2019-111
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 21 May 2019

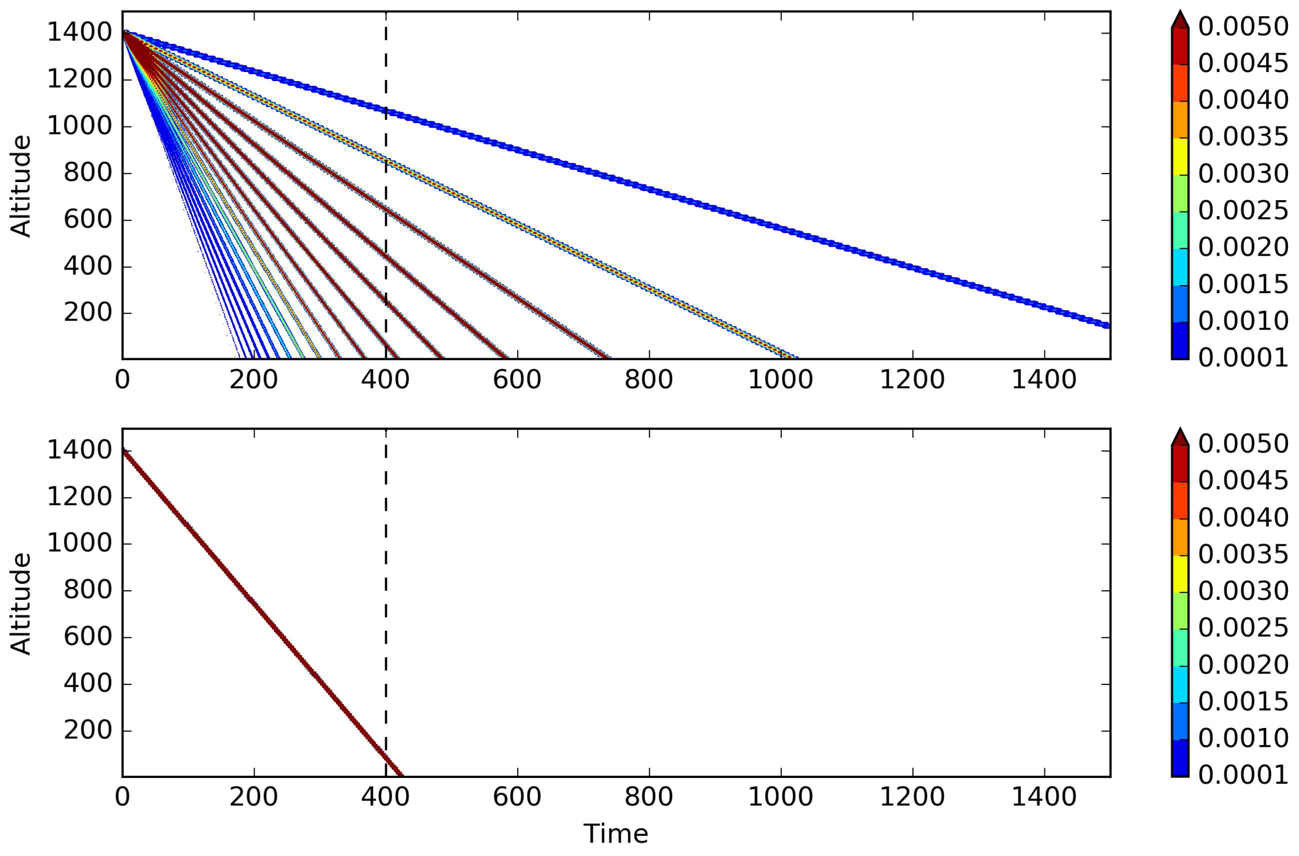

Figure 10. Time evolution of the vertical profile of rain mixing-ratio (the color scale represents the mixing-ratio in g kg^{-1}) for a bin box-Lagrangian scheme (upper panel) and a bulk box-Lagrangian scheme (lower panel).

Lagrangian scheme. These two examples have shown that it would be beneficial to use this kind of tool systematically when developing a parametrization in order to perform simple tests providing a first validation step (mass conservation, time-step dependency, absence of oscillations), before going through more complex validation stages (1D model, full simulations).

In addition to the ICE scheme, the LIMA microphysical scheme and some of the WRF microphysical schemes have been plugged into the tool. Then, it could now be used to compare microphysical schemes originally hosted by different models (AROME, Meso-NH and WRF). In order to compare them, it will be necessary to work on the initialization of the different schemes to insure that the results can actually be compared. Moreover, it will be necessary to define a suitable time step for each scheme to compare because of the time-step dependency present in the different schemes (a way to suppress it would be to select, for each scheme, the greater time step which allows the convergence towards the solution given by smaller time steps).

The tool is not limited to microphysical schemes and, in the future, it could also be used to compare other parametrizations like mass flux or turbulence schemes.

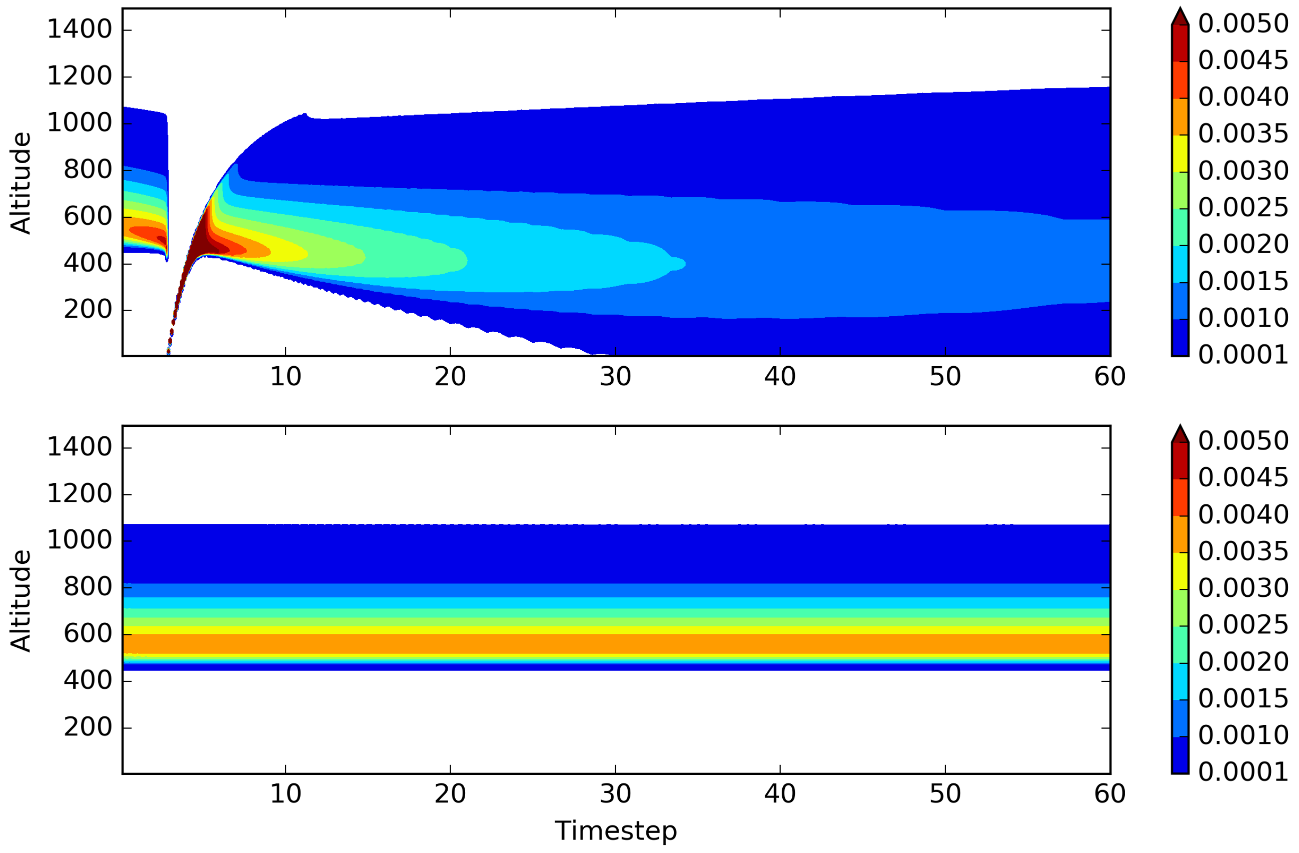

Figure 11. Vertical profile of the rain mixing-ratio (the color scale represents the mixing-ratio in $\mathrm{g\,kg^{-1}}$) after a 400 s long integration for different time steps (the time step varies between 0.1 s and 60 s with a 0.1 s step leading to 600 different simulations, abscissa) for the BSB2010 scheme (upper panel) and the Eulerian scheme as available in the operational source code (lower panel).

Code availability. PPPY is freely available under CeCILL-C license agreement (a French equivalent to the L-GPL license; http://www.cecill.info/licences/ C_V1-en.txt). PPPY v1.1 can be downloaded at https://opensource.umr-cnrm.fr/projects/pppy/files.

Competing interests. The author declares that he has no conflict of interest.

Acknowledgements. I which to thank B. Vié for having plugged the LIMA scheme into the tool, and the trainee team which work on the WRF plugging: A. Riandet, L. Richecoeur and M.-L. Roussel. I also thank C. Lac and B. Vié for their helpful comments on the manuscript.

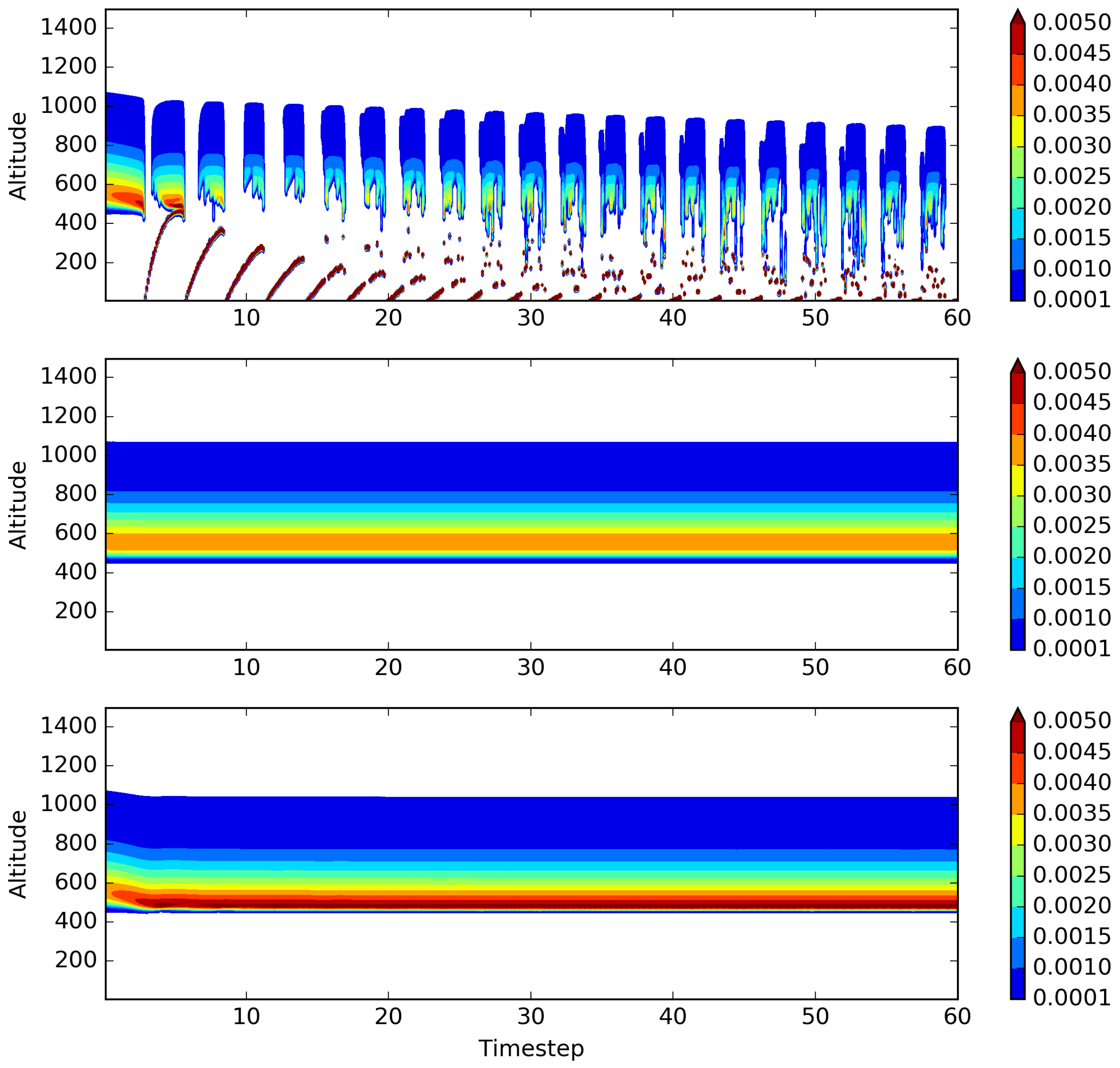

Figure 12. Same as Fig. 11 but for the modified version of the Eulerian scheme using different maximum values for the CFL number (1.0, 0.1 and 0.8 from top to bottom). The color scale represents the mixing-ratio in $\mathrm{g\,kg^{-1}}$.

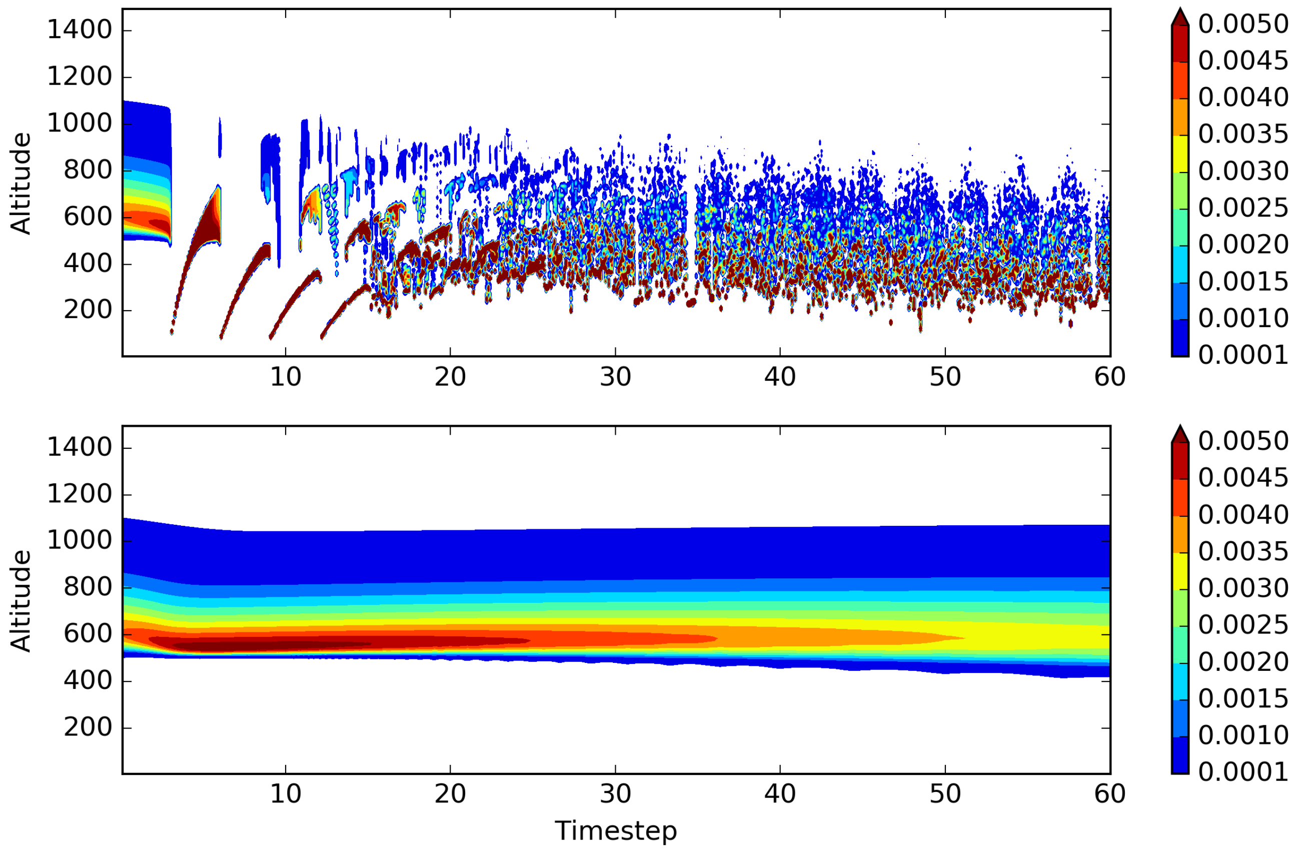

Figure 13. Same as Fig. 11 but for the box-Lagrangian scheme using the bulk approach (upper panel) and an hybrid approach (lower panel). The color scale represents the mixing-ratio in $\mathrm{g\,kg^{-1}}$.

References

Barrett, A. I., Wellmann, C., Seifert, A., Hoose, C., Vogel, B., and Kunz, M.: One Step at a Time: How Model Time Step Significantly Affects Convection-Permitting Simulations, Journal of Advances in Modeling Earth Systems, https://doi.org/10.1029/2018ms001418, 2019.

Bouteloup, Y., Seity, Y., and Bazile, E.: Description of the sedimentation scheme used operationally in all Météo-France NWP models, Tellus A, 63, 300–311, https://doi.org/10.1111/j.1600-0870.2010.00484.x, 2010.

Forbes, R.: Improved precipitation forecasts in IFS Cycle 45r1, ECMWF Newsletter, 156, 4, https://www.ecmwf.int/en/newsletter/156/news/improved-precipitation-forecasts-ifs-cycle-45r1, 2018.

Ghan, S., Randall, D., Xu, K.-M., Cederwall, R., Cripe, D., Hack, J., Iacobellis, S., Klein, S., Krueger, S., Lohmann, U., and et al.: A comparison of single column model simulations of summertime midlatitude continental convection, J Geophys Res, 105, 2091–2124, https://doi.org/10.1029/1999jd900971, 2000.

Henry Juang, H.-M. and Hong, S.-Y.: Forward Semi-Lagrangian Advection with Mass Conservation and Positive Definiteness for Falling Hydrometeors, Mon Weather Rev, 138, 1778–1791, https://doi.org/10.1175/2009mwr3109.1, 2010.

Hong, S.-Y. and Lim, J.-O. J.: The WRF single-moment 6-class microphysics scheme (WSM6), Journal of the Korean meteorological society, 42, 129–151, 2006.

Kato, T.: A Box-Lagrangian Rain-Drop Scheme, Journal of the Meteorological Society of Japan. Ser. II, 73, 241–245, https://www.jstage.jst.go.jp/article/jmsj1965/73/2/73_2_241/_article, 1995.

Khain, A., Pokrovsky, A., Pinsky, M., Seifert, A., and Phillips, V.: Simulation of Effects of Atmospheric Aerosols on Deep Turbulent Convective Clouds Using a Spectral Microphysics Mixed-Phase Cumulus Cloud Model. Part I: Model Description and Possible Applications, J. Atmos. Sci., 61, 2963–2982, https://doi.org/10.1175/jas-3350.1, 2004.

Lac, C., Chaboureau, J.-P., Masson, V., Pinty, J.-P., Tulet, P., Escobar, J., Leriche, M., Barthe, C., Aouizerats, B., Augros, C., Aumond, P., Auguste, F., Bechtold, P., Berthet, S., Bielli, S., Bosseur, F., Caumont, O., Cohard, J.-M., Colin, J., Couvreux, F., Cuxart, J., Delautier, G., Dauhut, T., Ducrocq, V., Filippi, J.-B., Gazen, D., Geoffroy, O., Gheusi, F., Honnert, R., Lafore, J.-P., Brossier, C. L., Libois, Q., Lunet, T., Mari, C., Maric, T., Mascart, P., Mogé, M., Molinié, G., Nuissier, O., Pantillon, F., Peyrillé, P., Pergaud, J., Perraud, E., Pianezze, J., Redelsperger, J.-L., Ricard, D., Richard, E., Riette, S., Rodier, Q., Schoetter, R., Seyfried, L., Stein, J., Suhre, K., Taufour, M., Thouron, O., Turner, S., Verrelle, A., Vié, B., Visentin, F., Vionnet, V., and Wautelet, P.: Overview of the Meso-NH model version 5.4 and its applications, Geosci Model Dev, 11, 1929–1969, https://doi.org/10.5194/gmd-11-1929-2018, 2018.

Milbrandt, J. A. and McTaggart-Cowan, R.: Sedimentation-Induced Errors in Bulk Microphysics Schemes, Journal of the Atmospheric Sciences, 67, 3931–3948, https://doi.org/10.1175/2010jas3541.1, 2010.

Milbrandt, J. A. and Yau, M. K.: A Multimoment Bulk Microphysics Parameterization. Part I: Analysis of the Role of the Spectral Shape Parameter, Journal of the Atmospheric Sciences, 62, 3051–3064, https://doi.org/10.1175/jas3534.1, 2005a.

Milbrandt, J. A. and Yau, M. K.: A Multimoment Bulk Microphysics Parameterization. Part II: A Proposed Three-Moment Closure and Scheme Description, Journal of the Atmospheric Sciences, 62, 3065–3081, https://doi.org/10.1175/jas3535.1, 2005b.

Morrison, H.: On the Numerical Treatment of Hydrometeor Sedimentation in Bulk and Hybrid Bulk–Bin Microphysics Schemes, Mon Weather Rev, 140, 1572–1588, https://doi.org/10.1175/mwr-d-11-00140.1, 2012.

Morrison, H., Thompson, G., and Tatarskii, V.: Impact of Cloud Microphysics on the Development of Trailing Stratiform Precipitation in a Simulated Squall Line: Comparison of One- and Two-Moment Schemes, Mon Weather Rev, 137, 991–1007, https://doi.org/10.1175/2008mwr2556.1, 2009.

Murakami, M.: Numerical Modeling of Dynamical and Microphysical Evolution of an Isolated Convective Cloud, Journal of the Meteorological Society of Japan. Ser. II, 68, 107–128, https://doi.org/10.2151/jmsj1965.68.2_107, 1990.

Pinty, J.-P. and Jabouille, P.: A mixed-phase cloud parameterization for use in mesoscale non-hydrostatic model: simulations of a squall line and of orographic precipitations, in: Proc. Conf. of Cloud Physics, Everett, WA, USA, Amer. Meteor. soc. Aug. 1999, pp. 217–220, http://www.gbv.de/dms/tib-ub-hannover/249955040.pdf, 1998.

Rogers, E., Black, T., Ferrier, B., Lin, Y., Parrish, D., and DiMego, G.: Changes to the NCEP Meso Eta Analysis and Forecast System: Increase in resolution, new cloud microphysics, modified precipitation assimilation, modified 3DVAR analysis, https://www.emc.ncep.noaa.gov/mmb/mmbpll/mesoimpl/eta12tpb/, 2001.

Seity, Y., Brousseau, P., Malardel, S., Hello, G., Bénard, P., Bouttier, F., Lac, C., and Masson, V.: The AROME-France Convective-Scale Operational Model, Mon Weather Rev, 139, 976–991, https://doi.org/10.1175/2010MWR3425.1, 2011.

Shipway, B. J. and Hill, A. A.: Diagnosis of systematic differences between multiple parametrizations of warm rain microphysics using a kinematic framework, Quarterly Journal of the Royal Meteorological Society, 138, 2196–2211, https://doi.org/10.1002/qj.1913, https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.1913, 2012.

Thompson, G., Field, P. R., Rasmussen, R. M., and Hall, W. D.: Explicit Forecasts of Winter Precipitation Using an Improved Bulk Microphysics Scheme. Part II: Implementation of a New Snow Parameterization, Mon Weather Rev, 136, 5095–5115, https://doi.org/10.1175/2008mwr2387.1, 2008.

Vié, B., Pinty, J.-P., Berthet, S., and Leriche, M.: LIMA (v1.0): A quasi two-moment microphysical scheme driven by a multimodal population of cloud condensation and ice freezing nuclei, Geosci Model Dev, 9, 567–586, https://doi.org/10.5194/gmd-9-567-2016, 2016.