The Kinematic Driver model (KiD)

Version 2.3.2625

B. J. Shipway & A. A. Hill

(last update: 16th May 2012)

Contents

1 Introduction

The original Kinematic Driver (KiD) model was designed as a basic wrapper to enable consistent testing of different microphysics schemes using a common advection component. It was not designed with an accurate representation of atmospheric dynamics in mind, but rather as a consistant and flexible framework for forcing microphysics schemes. That said, in situations where the vertical velocity is the principle advective forcing, e.g. wave clouds, frontal clouds, model may provide some useful insight. This original KiD model has been used effectively to investigate a range of microphysical processes (Shipway and Hill, 2012) and ice processes in wave clouds (Field et al, 2012).

At the GCSS microphysics meeting in Seattle 2010, it was decided that the capability of KiD model should be enhanced so that it can be a 1-D or 2-D kinematic framework. The initial reason for such a development was that it enables the simulation of published case studies of cumulus and stratocumulus, which can then be used for a microphysics intercomparison. In addition, a 2-D framework can utilise the 2-D wind fields from GCSS CRM intercomparisons hence be a useful tool when isolating role of microphysics in CRM intercomparisons.

This document describes the Kinematic Driver version 2.3(KiD v2.3), including details of new developments which mean it can run as the original 1-D KiD or as a 2-D kinematic framework.

2 Model variables and grid

In the vertical, the model grid of the KiD v2.3 is the same as that used in the original KiD model. The grid is defined in terms of "full" model levels and "half" levels, with height `z` and `z_half` held fixed on full and half levels respectively. While prognostic variables are held on full model levels, the vertical velocity (`w` and `w_half`) and density (`rho` and `rho_half`) are held on both sets of levels such that the grid can be used as Lorenz-type or Charney-Phillips-type for the chosen advection scheme. This vertical grid .

When the KiD v2.3 model is run in 2-D, the horizontal model grid is defined using a staggered grid with x distance (`x`) and x half distance (`x_half`). In the horizontal the prognostic components and velocity components are held on `x_half`.

The prognostic variables are:

`theta(:,0:nx+1)`	- potential temperature (K)

`qv(:,0:nx+1)`	- water vapour mixing ratio (kg/kg)
`hydrometeors(:, 0:nx+1,j)`	- jth hydrometeor category
`aerosols(:,0:nx+1,j)`	- jth aerosol category.

All prognostic variables are declared with dimension `nz` in the vertical, as in the original KiD model, and `0:nx+1` in the horizontal dimension. `nz` and `nx` are set in `parameters.f90`. The horizontal declaration includes haloes of `0` and `nx+1`, which are required for horizontal advection (performed with module `ultf2d_mod.f90`) and to permit periodic horizontal boundaries.

As in the original KiD model the pressure co-ordinate `exner` (exner pressure) is by default held fixed. Temperature (`TdegK`) and pressure (`pmb`) are updated diagnostically at the beginning of each timestep to facilitate their inclusion in microphysics subroutines.

All of the model variables are held in the module `column_variables.f90`.

2.1 The `species` derived type

In KiD v2.3, the hydrometeors are defined in a flexible derived type, `species`, in the module `class_species.f90`. The variable `hydrometeors(:,:,:)` has dimension `(nz,0:nx+1,nspecies)`, where `nz` and `nx` are parameters containing the number of vertical and horizontal levels, respectively, and `nspecies` a parameter set to be the number of hydrometeor species (see `parameters.f90`). Each element of the `hydrometeors(:,:,:)` array is of type `species` defined as:

```
type, public ::  species
   integer ::  h_id  (ID tag for species)
   integer ::  nmoments  (number of moments for species)
   integer ::  nbins  (number of bins for each moment)
   real(wp), pointer ::  moments(:,:) (the moments)
end type species.
```

Thus there is a flexibility for microphysics schemes that use differing number of moments, or between bulk schemes (`nbins=1`) and bin schemes.
In addition to the hydrometeors, arrays of type `species` are defined in the model for `aerosols`, which can be interfaced to the microphysics routines. By default there is one aerosol species (`naerosols=1`) having a single moment, single bin representation, but these can easily be extended through the namelists.

3 Advection

The default advection scheme is the Total Variance Diminishing(TVD) scheme of Leonard et al.(1993) known as ULTIMATE and is called through the `advection_interface.f90` routines. Other advection schemes can be simply interfaced through here.

As with the original KiD model, when KiD v2.3 is operated in a 1-D mode, there will be no mass conservation for a divergent vertical velocity field (i.e. if $\partial \rho w/\partial z \neq 0$). In order to mitigate this problem, a divergence term can be added such that horizontally homogeneous condition are assumed for scalars and a linear gradient of the horizontal velocity is assumed and determined from mass continuity (in 2 dimensions). Thus

$$\frac{\partial u}{\partial x} = -\frac{1}{\rho}\frac{\partial \rho w}{\partial z}, \tag{1}$$

and

$$\frac{d\chi}{dt}_{div} = -\chi\frac{\partial u}{\partial x}. \tag{2}$$

If `l_diverge` is set to true (see section 7 on namelists) then this extra tendency term is calculated in `divergence.f90` otherwise if `l_diverge_advection` is set to true, this term is implicit to the advection scheme. If both `l_diverge` and `l_diverge_advection` are false, then no effort is made to correct for this divergence.

If the KiD v2.3 model is run in 2-D then both `l_diverge_advection` and `l_diverge` should be set to false.

4 Interfacing a new microphysics subroutine

In order to interface a new microphysics scheme a switch value (`imphys_<name>`) must first be placed in the `switches.f90` module, and a `case` statement calling a bespoke microphysics interface routine added to the `mphys_interface.f90` module. An example of setting up an interface module and subroutine is given in `mphys_thompson07.f90`. For this scheme the interfacing is simply a case of importing the column variables and parameters (`Use parameters`, `Use column_variables`), passing the relevant variables through to your microphysics subroutines and returning the microphysics tendencies to the dχ`_mphys` arrays (where χ represent the prognostic variables). This method for including a scheme is the same as the original KiD model.

With the KiD v2.3 model, care needs to be taken when passing the variables to the microphysics scheme. In the example interface module (`mphys_thompson07.f90`), the prognostic variables from `column_variables` are passed as columns to `module_mp_thompson07.f90` and the loop over `nx` is executed in the interface. If, however, the bespoke microphysics scheme loops over `nx` and `nz` in its own modules, then the 2-D arrays need to be passed and the interface should not include a loop over `nx`.

As with the original KiD model, in the KiD v2.3 model, the advection, divergence, microphysics and imposed forcing are all calculated in parallel, with fields being updated at the end of the timestep in the `stepfields.f90` module. If your microphysics scheme requires fields updated after the advection step, these can be obtained by using the dχ`_adv` and dχ`_div` arrays, but only the microphysical component of the tendency should be passed back to dχ`_mphys`. For an example of how to interface a bulk microphysics scheme, see `mphys_thompson07.f90` or `mphys_morr_two_moment.f90`.

5 Adding user information

The module `header_data.f90` contains information which is written out with the diagnostic files (including defining the filenames of the data files). In order to facilitate processing of any data, please update this code where appropriate (i.e. `username, institution, mphys_id`) if you have any comments which are useful to those reading in the data, then please add these to the `comments` variable.

6 Diagnostics

The diagnostic data is saved on each diagnostic timestep (`DG_DT`, set in the `&CONTROL` namelist) using the `save_dg` subroutine. In general, data is saved into 0D, 1D or 2D instantaneous fields with the type of diagnostic being dependent on whether KiD v2.3 is used in 1-D or 2-D mode. This section explains the differences and how to output the data.

6.1 0-D diagnostics

0D diagnostic refers to scalar timeseries, such as liquid water path or surface precipitation. When KiD v2.3 is simulating a 1-D case to save the diagnostic for surface precipitation, for example, the following is executed

```
call save_dg(ppt, 'ppt', i_dgtime, 'mm/hr', dim='time')
```

where `ppt` is the generic precipitation variable. `i_dgtime` is the current diagnostic timestep and is made available along with the `save_dg` subroutine through the `diagnostics.f90` module (i.e. by adding

```
Use diagnostics, only:  save_dg, i_dgtime
```

to the top of your microphysics subroutine/module).

When KiD v2.3 is used to simulate a 2-D case it is assumed that the 0-D diagnostic is averaged over `nx`. Thus, the horizontally averaged scalar timeseries needs to be calculated in the microphysics or interface code, or it needs to be specified in the call for diagnostic, e.g.

```
call save_dg(ppt/nx, 'ppt', i_dgtime, 'mm/hr', dim='time')
```

6.2 1-D diagnostics

When KiD v2.3 is used in 1-D mode 1D diagnostics refer to height profile timeseries of instantaneous fields. For example, outputting a profile of vapour mixing ratio, `qv`, on all levels at each diagnostic timestep is done with the following code is executed in `diagnostics.f90`

```
field(:)=qv(:,nx)
call save_dg(field,'vapour',i_dgtime, units='kg/kg',dim='z')
```

The above line will save a diagnostic for the entire `field`. The diagnostic can also be called within the loop `nz`, so that there is a diagnostic call for each level, `k`, e.g.

```
call save_dg(k,qv(k),'vapour',i_dgtime, units='kg/kg',dim='z')
```

In KiD v2.3 it is assumed that when `nx = 1`, i.e. 1-D, 1-D diagnostics are always a timeseries of height profile because `save_diagnostics_1d.f90` is called from `main.f90`. It is also assumed that diagnostic has dimension `nz`. Note that height profile timeseries are NOT output when KiD v2.3is running a 2D case, i.e. (`nx > 1`).

When the KiD v2.3 model is run in 2-D mode, 1D diagnostics refer to timeseries of scalar diagnostics, e.g. surface precipitation or liquid water path, for all horizontal columns. For example,

```
field_2d(:)=ppt(1:nx)
call save_dg(field,'surface ppt 2D',i_dgtime, units='mm/hr',dim='x')
```

This code will output the surface precipitation for each column in 2-D domain.

In KiD v2.3 it is assumed that when `nx > 1`, i.e. 2-D, 1-D diagnostics are always a timeseries of scalar diagnostic for all horizontal columns because `save_diagnostics_2d.f90` is called from `main.f90`. It is also assumed that diagnostic has dimension `1:nx`, i.e. the haloes are not included.

6.3 2-D diagnostics

When KiD v2.3 is used in 2D mode, 2D instantaneous fields, which are a timeseries of the 2D field, are output. For example, to output a 2D field of vapour mixing ratio at each diagnostic timestep, the following is used

```
call save_dg(qv(1:nz,1:nx), 'vapor', i_dgtime, 'kg/kg', dim='z,x')
```

If required, the 2D instantaneous fields can be post processed to derive the mean height profile timeseries.

6.4 General description of `save_dg`

In all example diagnostics above interface `save_dg` is used to save the diagnostic. For all the `save_dg`, the `units, dim` and `longname` arguments are optional, but it is recommended that `units` and `longname` are used where possible. The `dim` argument need only be used to define if a column variable is carried on 'z' levels or 'z_half' levels ('z' levels for 1D or 'z,x' levels are assumed if it is omitted).

6.5 Adding Diagnostics

It is a relatively simple procedure to add in diagnostics and it is anticipated that users will add in relevant diagnostics to their microphysics routines. To add 0D, 1D or 2D timeseries, as described above, a user can follow the examples presented already, remembering to make available `i_dgtime` and `save_dg` by adding

```
Use diagnostics, only:  save_dg, i_dgtime
```

to the top of your microphysics subroutine/module.

When the KiD v2.3 is used in 1D mode, If you wish to save a point or range of points into the 1D data (e.g. if you calculate some values pointwise rather than carrying in an array), then simply prepend the grid index or start and end of the range of indices to the start of the subroutine arguments, e.g.

```
call save_dg(k,lambda,'lambda',i_dgtime, units='-',dim='z')
```

If no indices are given and `nx = 1`, then it will be assumed that this will be saved into a height profile timeseries.

If no indices are given and the field to save is a scalar value, then it will be assumed that this will be saved into a scalar timeseries, e.g

```
call save_dg(ppt,'ppt',i_dgtime,                          &
             units='mm/hr', longname='surface precipitation').
```

When the KiD v2.3 is used in 2D mode, i.e. `nx > 1`, the same syntax above is employed; however, it is assumed that if no index is provided the a 2D field is supplied and 2D timeseries is saved. If a 1D array is provided, then it is assumed that the array dimension is `nx`. As with 1D mode, if `nx > 1` and no indices are given in `save_dg` and the field to save is a scalar value, then it will be assumed that this will be saved into a scalar timeseries.

7 Namelists

The namelists are read in to control a number of aspects of the running of the code including the switches for choice of microphysics and advection schemes. In summary these are (default values given):

```
&MPHYS
     NUM_H_MOMENTS= 1,1,2,1,1,                 ! Number of moments for each
                                               ! hydrometeor species
     NUM_H_BINS= 1,1,1,1,1                     ! Number of bins for each
                                               ! hydrometeor species
     MOM_INIT= 0.0 , 1.0 , 0.0 ,               ! Initial value for each moment
                                               !(assumed the same for each species)
     H_NAMES='cloud' ,'rain' ,'ice'            ! Name of each species
     ,'snow' ,'graupel' ,                      !
     MOM_NAMES='mass' ,'number' ,'volume' ,    ! Name of each moment
                                               ! (assumed the same for each species)
     MOM_UNITS='kg/kg' ,'/kg' ,'m3/kg' ,       ! Units for each species
                                               ! (assumed the same for each species)
/
```

```
&CONTROL
      DT= 1.000000 ,                        ! Timestep
      DG_DT= 10.00000 ,                     ! Diagnostic time step
                                            ! (should be multiple of dt)
      MPHYS_SCHEME='lem2.4' ,               ! Microphysics scheme to use
      MPHYS_VAR= 0,                         ! Switch to choose variants of
                                            ! microphysics scheme. (E.g. using
                                            ! alternative parametrizations
                                            ! or constants)
/

&CASE
      INPUT_FILE='input.nc',                ! Input filename if required
      L_INPUT_FILE=.false.,                 ! Switch to indicate using
                                            ! an input file
      IFILETYPE= 998,                       ! Switch to choose format
                                            ! for input file
                                            ! (see switches.f90 )
      ICASE= 101,                           ! Switch to choose standard
                                            ! test case
                                            ! (see switches.f90 )
/

&SWITCH
      L_MPHYS=.true.,                       ! Use microphysics
      L_ADVECT=.true.,                      ! Use advection
      L_DIVERGE=.false.,                    ! Use divergence code
      L_DIVERGE_ADVECTION=.true.,           ! Add u field to advection
                                            ! such that continuity is satisfied
      L_PUPDATE=.false.,                    ! Update pressure field
      L_FIX_QV=.false.,                     ! Keep qv fixed
      L_NOMPHYS_QV=.false.,                 ! Don't update qv with
                                            ! microphysics increments
      L_NOADV_QV=.false.,                   ! Don't update qv with
                                            ! advection increments
      L_POSADV_QV=.false.,                  ! Only update qv with
                                            ! positive advection increments
      L_FIX_THETA=.false.,                  ! Keep theta fixed
      L_NOMPHYS_THETA=.false.,              ! Don't update theta with
                                            ! microphysics increments
      L_NOADV_THETA=.false.,                ! Don't update theta with
```

```
                                              ! advection increments
     L_NOADV_HYDROMETEORS=.false.,            ! Don't update hydrometors with
                                              ! advection increments
     L_NODIV_HYDROMETEORS=.false.,            ! Don't update hydrometors with
                                              ! divergence increments
     L_SEDIMENT=.true.,                       ! Don't allow sedimentation
                                              ! in microphysics scheme
                                              ! (requires implementation in
                                              ! microphysics code)
     ISURFACE= 1,                             ! Switch for surface condition
                                              !     1= fixed surface values
                                              !     2= use supplied surface fluxes
                                              ! (NB ISURFACE is generally
                                              ! set to 1)
     L_NOADV_AEROSOLS=.False. ! Don't advect aerosols
                                              ! NB This doesn't include sedimentatio
     L_NODIV_AEROSOLS=.False. ! Don't apply divergence terms to
                                              ! aerosols
     L_PERIODIC_BOUND=.True. ! True for periodic boundaries (default)
                                              ! ! False for fixed boundaries
/
&ADDCONTROL
! The addcontrol namelist is included to allow users to add in any
! additional controls to their microphysics scheme, e.g. changes to
! fall-speed coefficients, number concentrations, switches to use
! alternate parametrizations ...
/
```

8 Building and running the model

The code is built in the top level directory using the `makefile`. However this further uses the files `makefile`, `compiler_options.inc`, `depends.inc`, `makefile.inc` and `mkdepends.py`[1] in the `src/` directory. A few customizations to the `compiler_options.inc` may be necessary before building...

1. Set the variable COMPILER to the path for your compiler.

[1]This requires python 2 or later to be installed. If this is not available on your system, then you may need to build the model manually.

2. Set the variable NCPATH to the path to your netcdf library directory (this assumes the directories `lib` and `include` are subdirectories.)
3. Set any compiler flags you wish to use.

For a standard build of the original 1D model, once you have unpacked the model simply `cd` into the top level directory (NOT `/src`) and type

```
>make CASE=1D all
>./bin/KiD_1D.exe
```

This will run the model with the default namelist. By default the ifort compiler is used, as specified in point 1 above, to change this modify `COMPILER` in `src/compiler_options.inc`. You can also specify this at the command line, i.e.

```
>make COMPILER=gfortran CASE=1D all
```

The preprocessor directives that define the grid are passed through to the model from `src/defines.inc`. It is recommended that you add in new definitions if wishing to run at different resolutions.

There are currently 5 different case configurations defined;
1D, SC_2D, CU_2D, ISDAC_2D, SQUALL_2D
to build exectuables for them all, use

```
>make build_cases
```

(again a different compiler can be specified as above). To run with all the default namelists, use

```
>make run_cases
```

This should run the cases sequentially and produce netcdf files (and corresponding namelists) in output (Note that 2D cases take a lot longer to run than previous 1D cases). If you wish to exploit multiple cpus on your system (currently no OMP support in KiD), you can use the -j option, i.e.

```
>make -j run_cases
```

(use the -l option to limit load on the system). As with previous versions if your compiler allows it, custom namelists and output can be specified at the command line when running, e.g.

```
>./bin/KiD_SC_2D.exe namelists/SC_2D.nml output/output.nc
```

If no command line arguments are given, default namelists will be used. If no output file is specified, the default naming convention for the output will be used.

9 Test cases

Currently there are a few 1-D and two 2-D test-cases supplied. A brief description of some of these cases is given here. For full details of the 1-D cases see the subroutine `test_cases.f90` and for the 2-D cases see `test_cases_2d.f90`.

9.1 1-D Test cases

9.1.1 Case 1 (namelist file = `namelists/warm1.nml`)

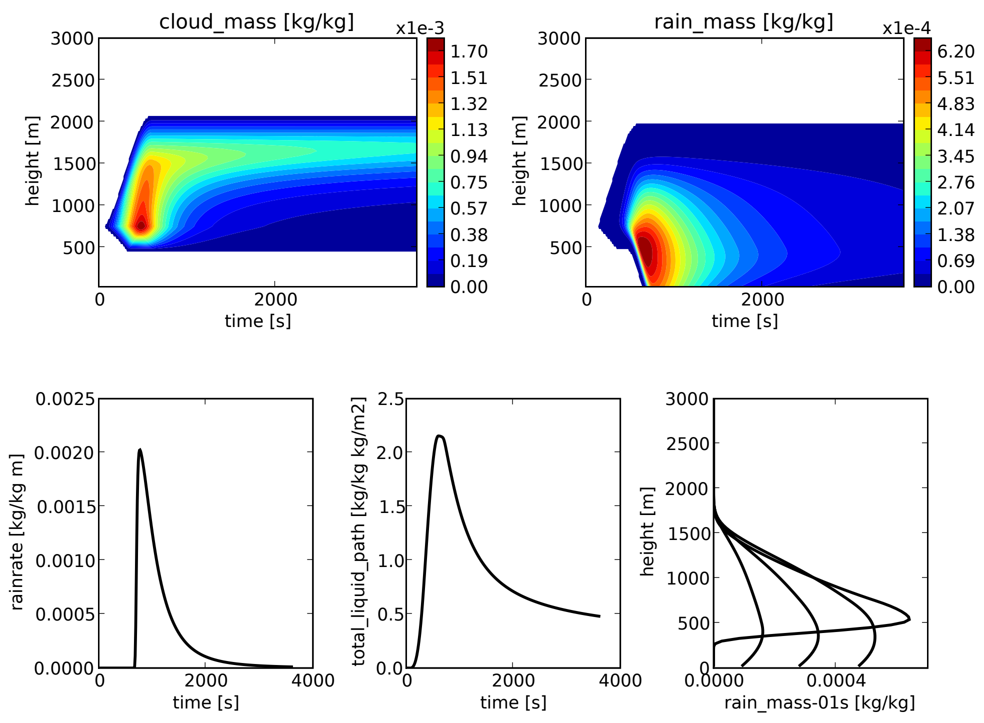

Figure 1: Output from the warm rain case 1 for a single moment bulk microphysics scheme. Contour plots of cloud mass and rain mass, line plots of surface rain rate, total liquid water path and profiles of rain mass at 10, 15, 20 and 30 minutes.

Case 1 uses a simple updraught, which is sinusoidal in time and constant in height, to advect vapour and hydrometeors. The temperature field is kept fixed so as to minimize feedback from the different microphysical schemes (in a full 3D simulation the temperature within an updraught core would be further influenced by entrainment mixing and so reduce the temperature variations that may be seen from vertical advection alone). Thus the updraught velocity is given as

$$w(z,t) = \begin{cases} w_1 \sin(\pi t/t_2), & \text{if } t < t_1, \\ 0.0, & \text{if } t > t_2, \end{cases} \tag{3}$$

where default values for the parameters are given in table 9.2.2.

This case is the same as the W2 case described in Shipway and Hill (2012) with the same initial profiles of temperature and moisture. The initial profiles were set to be similar to those used in the GCSS RICO composite intercomparison (`ipctrl=1`) and the depth of the simulation is set to z_1=3000m.

Note: As described in Shipway and Hill (2012), the initial moisture profile for this case is drier at the surface compared to that used in the *warm1* case, which was included in the in the first release of the KiD model. The reason for this change is to make the simulated liquid water increase with height and thus, increase the realism of the simulation and the simulated processes.

9.1.2 Case 2 (`namelists/warm2.nml`)

Case 2 uses a similar form for the case 1 updraught, but extends it to multiple repeated updraught/downdraughts which are sinusoidal in time and constant in height. Again these are set to advect vapour and hydrometeors, but the temperature field is kept fixed. Thus the updraught velocity for case 2 is given as

$$w(z,t) = w_1 \sin(\pi t/t_2), \text{ for } \forall t > 0, \tag{4}$$

where now default values are w_1=2ms^{-1}, t_1=7200s and t_2=600s, thus producing 6 updraught/downdraughts in succession.
The simulation depth and the inital profiles of temperature and moisture are as case 1.

9.1.3 Case 3 (`namelists/warm3.nml`)

Case 3 uses a similar form for the updraught/downdraught of case 2, but there is an exponential temporal decay in the magnitude of the velocity, such that the updraught velocity for case 3 is given as

$$w(z,t) = w_1 \sin(\pi t/t_2)\, e^{-t/t_3}, \text{ for } \forall t > 0, \tag{5}$$

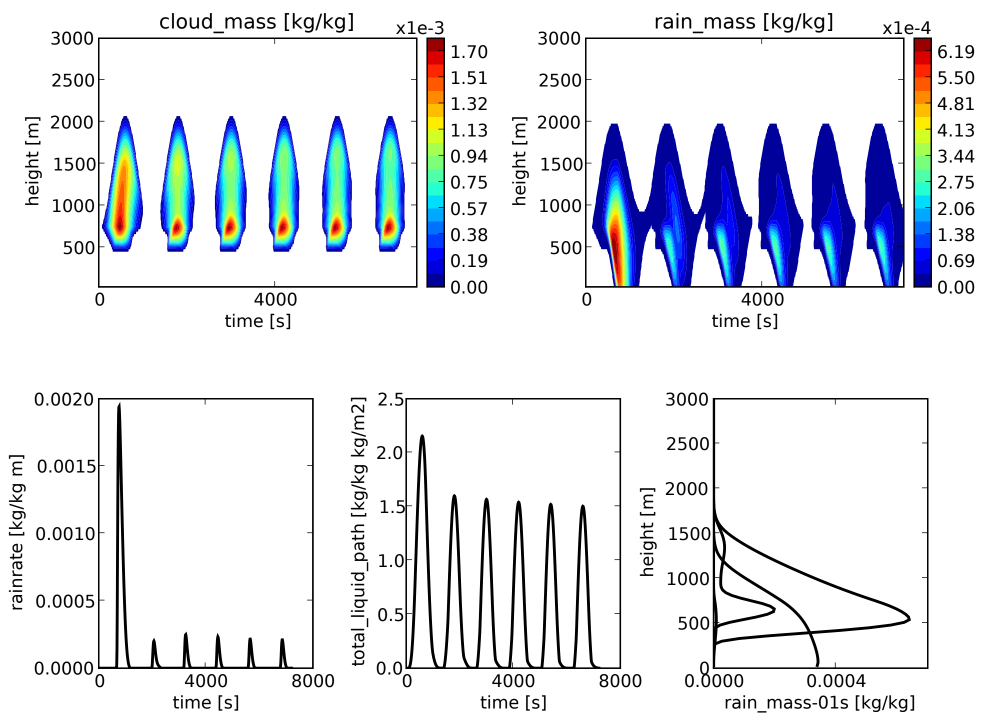

Figure 2: As figure 1 but for case 2.

where again details of the parameters are given in table 9.2.2
The simulation depth and the inital profiles of temperature and moisture are as case 1.

9.1.4 Case 5 (`namelists/warm5.nml`)

Case 5 is chosen to replicate an updraught core modelled on data taken from the simulations of Abel and Shipway(2007). Unlike the previous cases, the updraught strength varies in height as well as time and is given as

$$w(z,t) = A(t)\exp(-((z(k) - B(t))/C(t))^2), \tag{6}$$

where

$$A(t) = w_1(1 - \left(\frac{t - t_2}{t_3}\right)^4), \tag{7}$$

$$B(t) = \begin{cases} z_2 + (z_3 - z_2)t/t_2, & \text{if } t < t_2, \\ z_4, & \text{if } t > t_2, \end{cases} \tag{8}$$

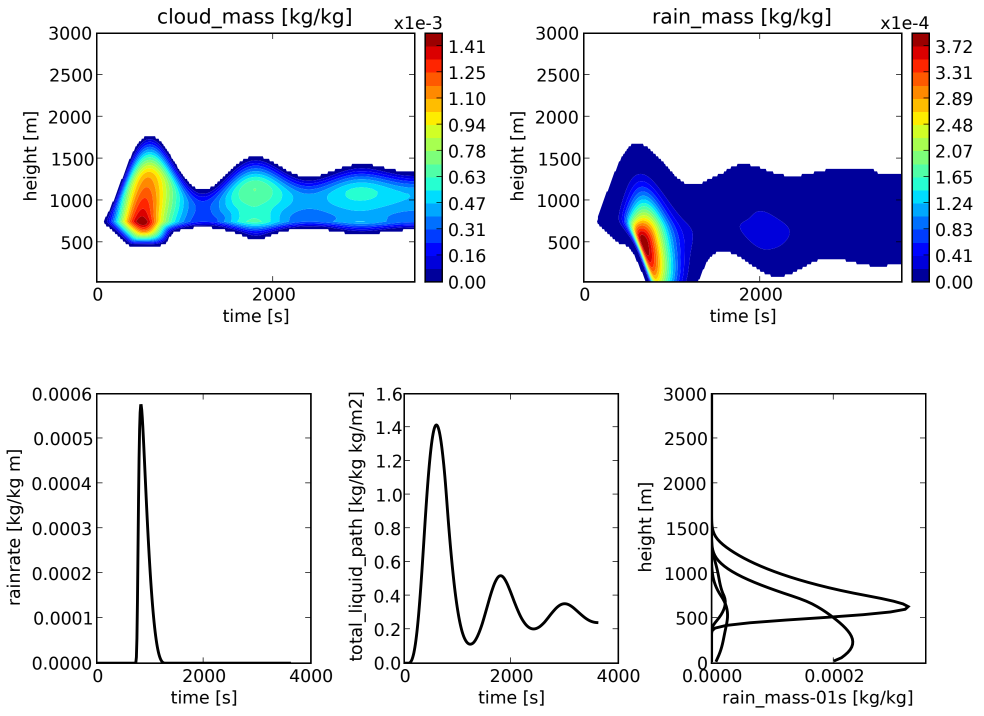

Figure 3: As figure 1 but for case 3.

$$\text{utop}(t) = \begin{cases} z_3 + (z_6 - z_3)t/t_2, & \text{if } t < t_2, \\ z_6, & \text{if } t_2 < t < t_3, \\ z_6 + (z_5 - z_6) * (t - t_3)/(t_4 - t_3), & \text{if } t > t_3, \end{cases} \tag{9}$$

and $C(t) = \max(\text{utop}(t) - B(t), B(t))$. Details of the parameters are given in table 9.2.2
The profiles of temperature and moisture are those described by Shipway and Abel(2008) (`ipctrl=3`).

9.1.5 Case 6 (`namelists/warm6.nml`)

Case 6 is as case 5, but for a shallower weaker updraught, using the profiles of temperature and moisture taken from the GCSS RICO composite intercomparison case (`ipctrl=2`). The form for the updraught is as in relations (6)-(9), but with parameters as given in table 9.2.2.

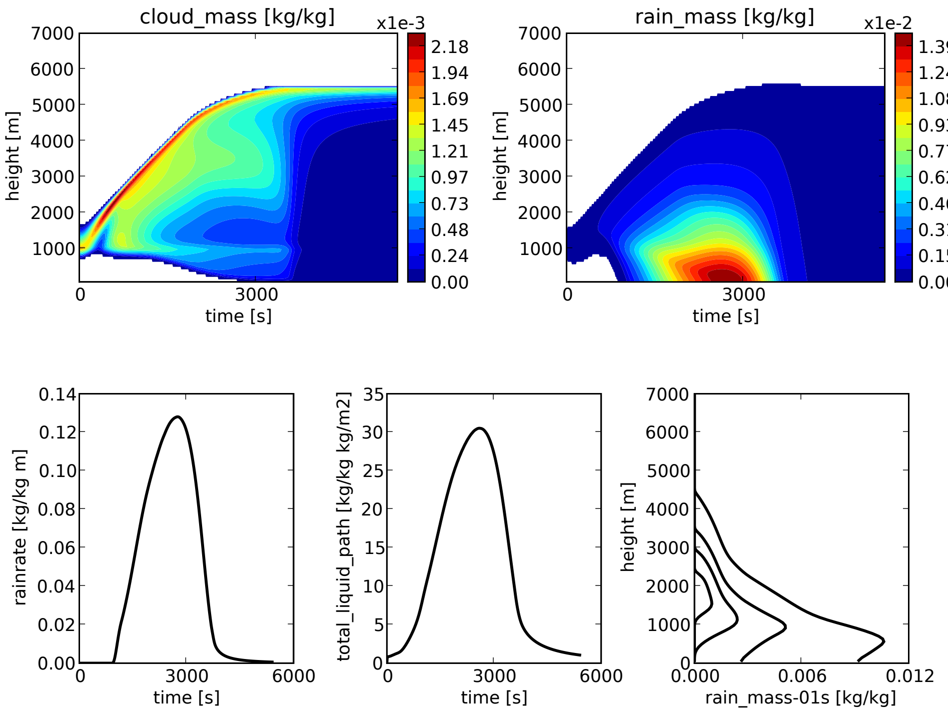

Figure 4: As figure 1 but for case 5.

9.1.6 Case 7 (`namelists/warm7.nml`)

Case 7 attempts to simulate a quasi-steady stratocumulus layer. As in case 2, an oscillating updraught/downdraught is applied (varying in height as well as time), and an additional forcing term for the vapour field is enabled. The magnitude of the forcing is chosen so as to reach a steady precipitation rate of f_{v1}mm hour^{-1}. The updraught is given as

$$w(z,t) = \begin{cases} w_1 \dfrac{z}{z_6}\left(1 - \exp\left[-\left(\dfrac{z - z_6}{z_2}\right)^2\right]\right)\sin\left(\pi t/t_2\right), & \text{if } z < z_6, \\ 0.0, & \text{otherwise,} \end{cases} \qquad (10)$$

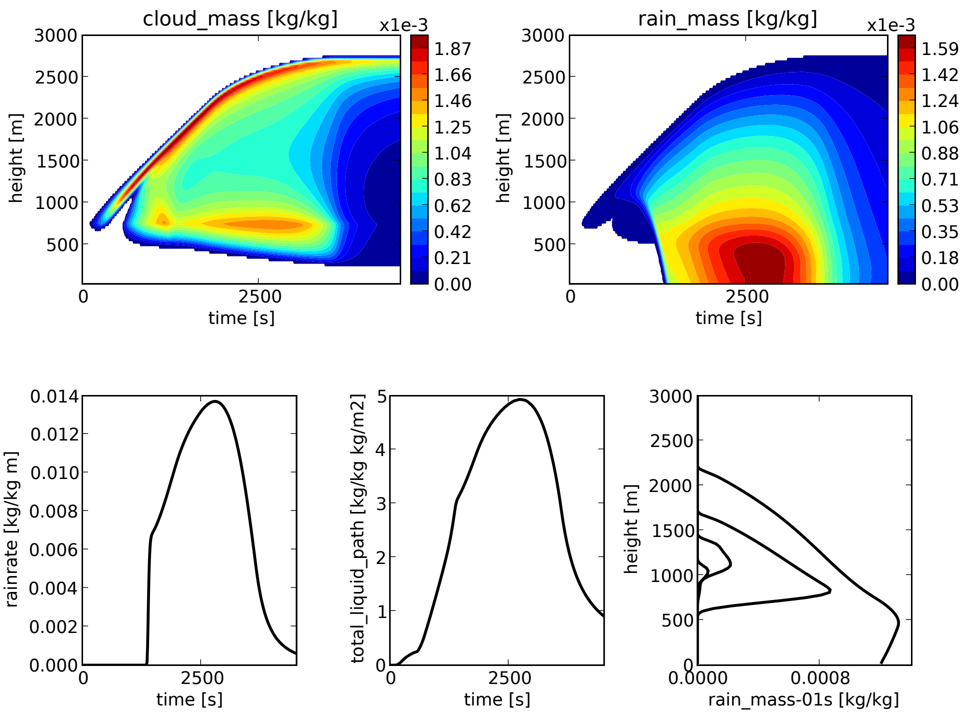

Figure 5: As figure 1 but for case 6.

and the forcing

$$\frac{dq}{dt}_{force}(z,t) = \frac{dq}{dt}_{force}(z,0) = \begin{cases} A\cos(\frac{1.25\pi}{2}), & \text{if } z < z_5 + z_4, \\ A\cos(\frac{z-z_5}{z_4}\frac{\pi}{2}), & \text{if } z_5 + z_4 < z < z_5 + 1.25z_4, \\ 0.0, & \text{otherwise}, \end{cases} \tag{11}$$

and A satisfies

$$\int_0^{z_1} \frac{dq}{dt}_{force}(z,0)dz = f_{q1}/3600. \tag{12}$$

The profiles of temperature and moisture are taken from the GCSS DYCOMS intercomparison case (`ipctrl=2`).

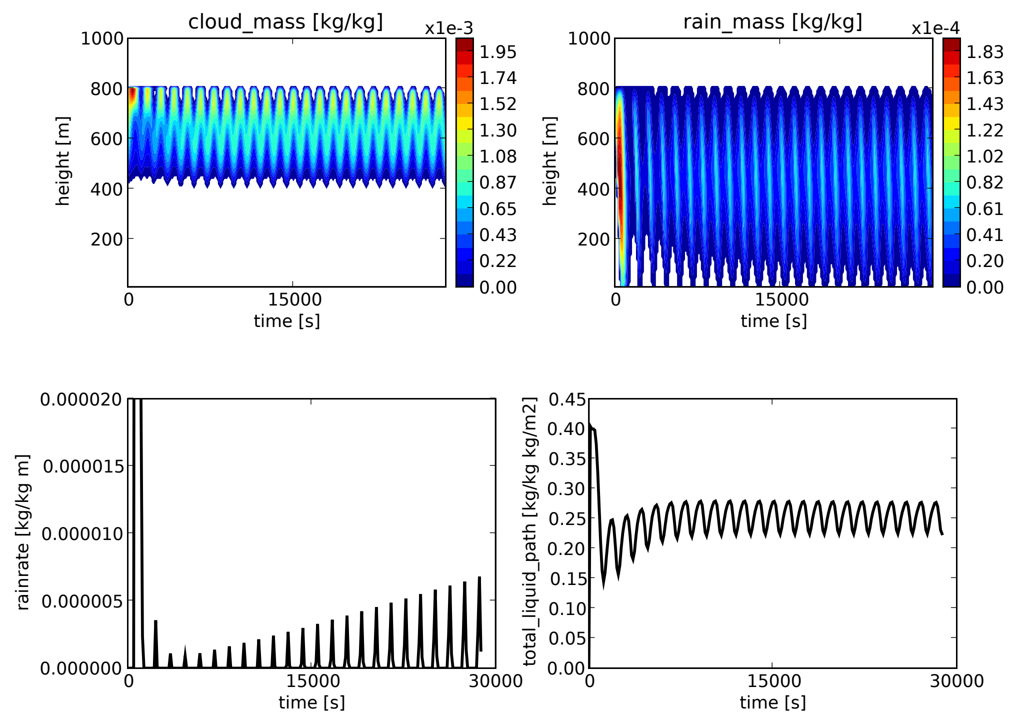

Figure 6: Output from the warm rain case 7 for a single moment bulk microphysics scheme. Contour plots of cloud mass and rain mass, line plots of surface rain rate and total liquid water path.

9.1.7 Case 8 (`namelists/mixed1.nml`)

Case 8 is designed as a test of a mixed-phase stratocumulus layer, with the profile taken from the GCSS SHEBA intercomparison (`ipctrl=5`. MPACE profile can also be used with `ipctrl=6`). In this case, an oscillating updraught/downdraught with a vertical profile linearly increasing in magnitude with height is imposed. Thus

$$w(z,t) = w_1 \left(\frac{z}{z_2}\right) \sin\left(\pi t/t_2\right). \tag{13}$$

9.1.8 Case 9 (`namelists/mixed2.nml`)

Case 9 is another mixed-phase stratocumulus case, but as in the warm case 7, a constant forcing of the vapour field is applied such that a quasi-equilibrium

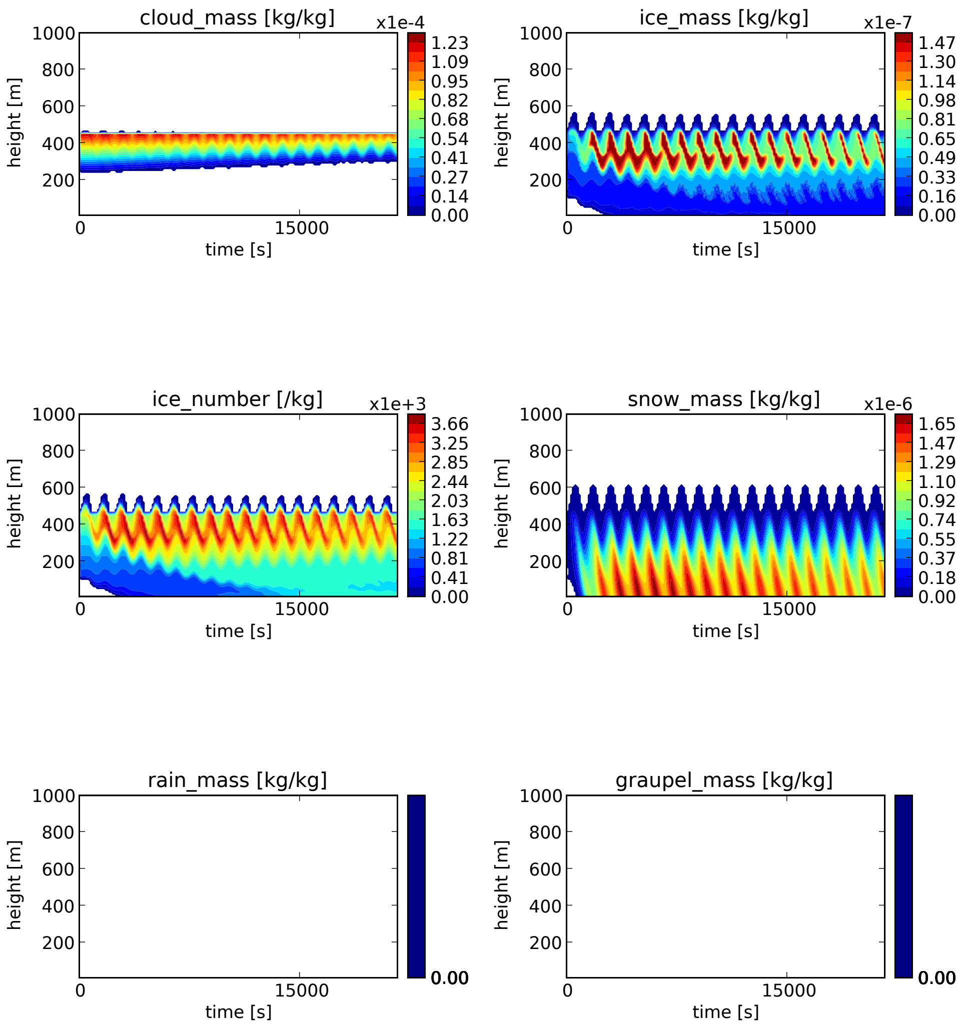

Figure 7: Output from the mixed-phase case 8 for a single moment bulk microphysics scheme. Contour plots of cloud mass, ice mass, ice number, snow mass, rain mass and graupel mass (no rain or graupel produced)

may be attained. The updraught profile is also similar to that used in case 7 and so $w(z,t)$ is of the form given in equation (10) and the forcing as described by relations (11) and (12).

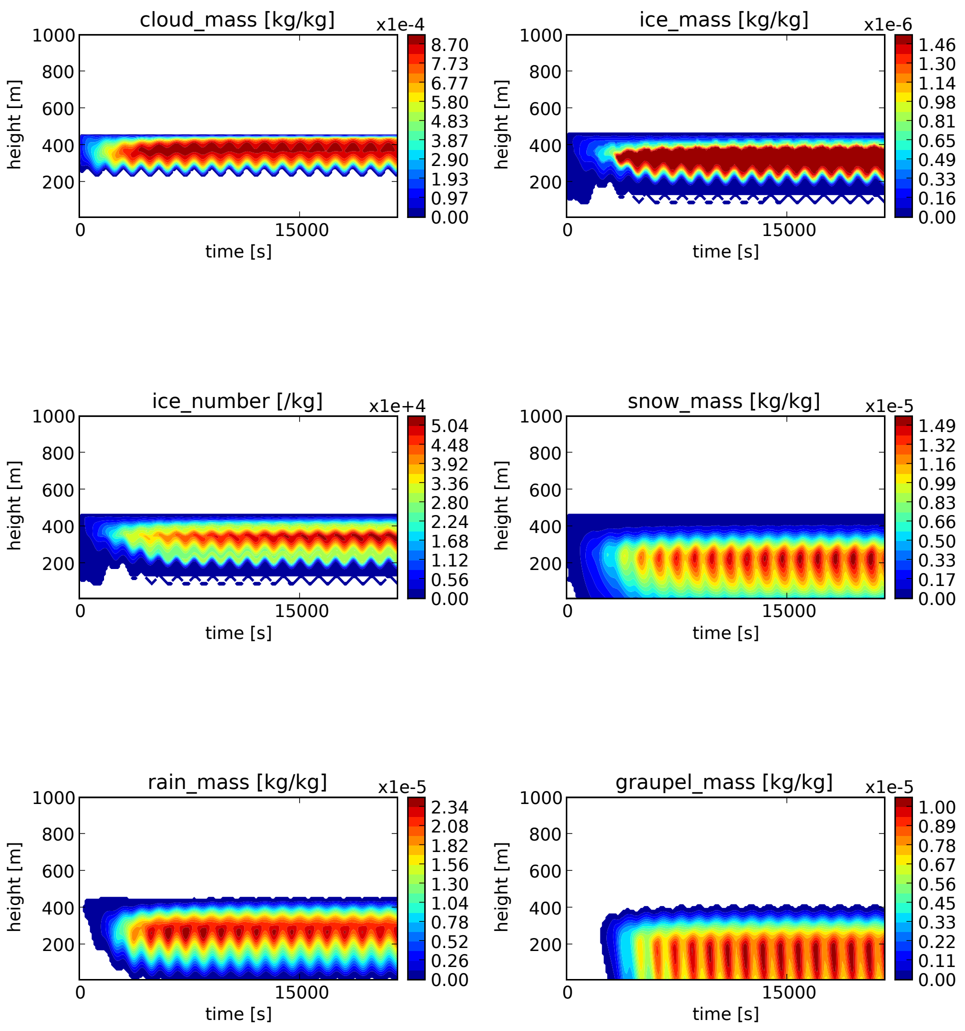

Figure 8: As figure 7 but for case 9.

9.1.9 Case 10 (`namelists/deep1.nml`)

Case 10 is developed as a steady-state test of deep convection. An updraught constant in time is applied to the hydrometeors (not vapour or temperature) while a source term for vapour, constant in time and varying with height, is

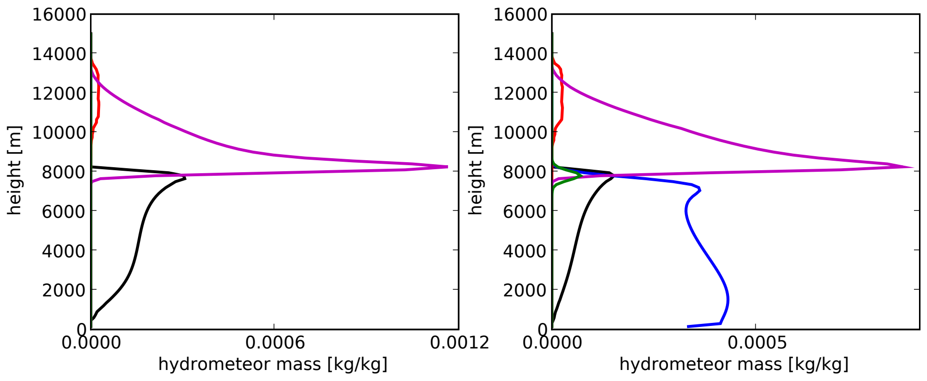

Figure 9: Vertical profiles of hydrometeor mass mixing ratio for the equilibrium deep case 10 using a single moment bulk microphysics scheme. Lines are black=cloud liquid, red=ice, magenta=snow, green=graupel and blue=rain. The left plot is a snapshot after 1hr of the simulation, the right is at 6hrs.

applied with a magnitude such that a steady rain rate of 5mmhr^{-1} may be expected. Thus the forms for the updraught and forcing are

$$w(z,t) = \begin{cases} w_1 \cos^4\left(\frac{\pi}{2}\frac{z-z_3}{z_2}\right), & \text{if } |z-z_2| < z_3, \\ 0.0, & \text{otherwise,} \end{cases} \tag{14}$$

$$\frac{dq}{dt}_{force}(z,t) = \frac{dq}{dt}_{force}(z,0) = \begin{cases} A\cos^2\left(\frac{\pi}{2}\frac{z-z_5}{z_4}\right), & \text{if } |z-z_4| < z_5, \\ 0.0, & \text{otherwise,} \end{cases} \tag{15}$$

and A satisfies

$$\int_0^{z_1} \frac{dq}{dt}_{force}(z,0)dz = f_{q1}/3600 \tag{16}$$

with f_{q1}=5mm hour^{-1}.
For this case the temperature profile is dry adiabtic and the vapour mixing ratio profile is constant, until the saturation level is reached. Above this level the temperature is pseudoadiabatic and the vapour saturated. At temperatures below 0C the vapour is saturated with respect to ice. The surface temperature, pressure and water vapour mixing ratio are 300 K, 1000 hPa and 18 g/kg, respectively.
While the vapour field evolves through the source term and the microphysical tendencies, it is not advected. The temperature field is held fixed for all time.

9.1.10 Case 11 (`namelists/deep1.nml`)

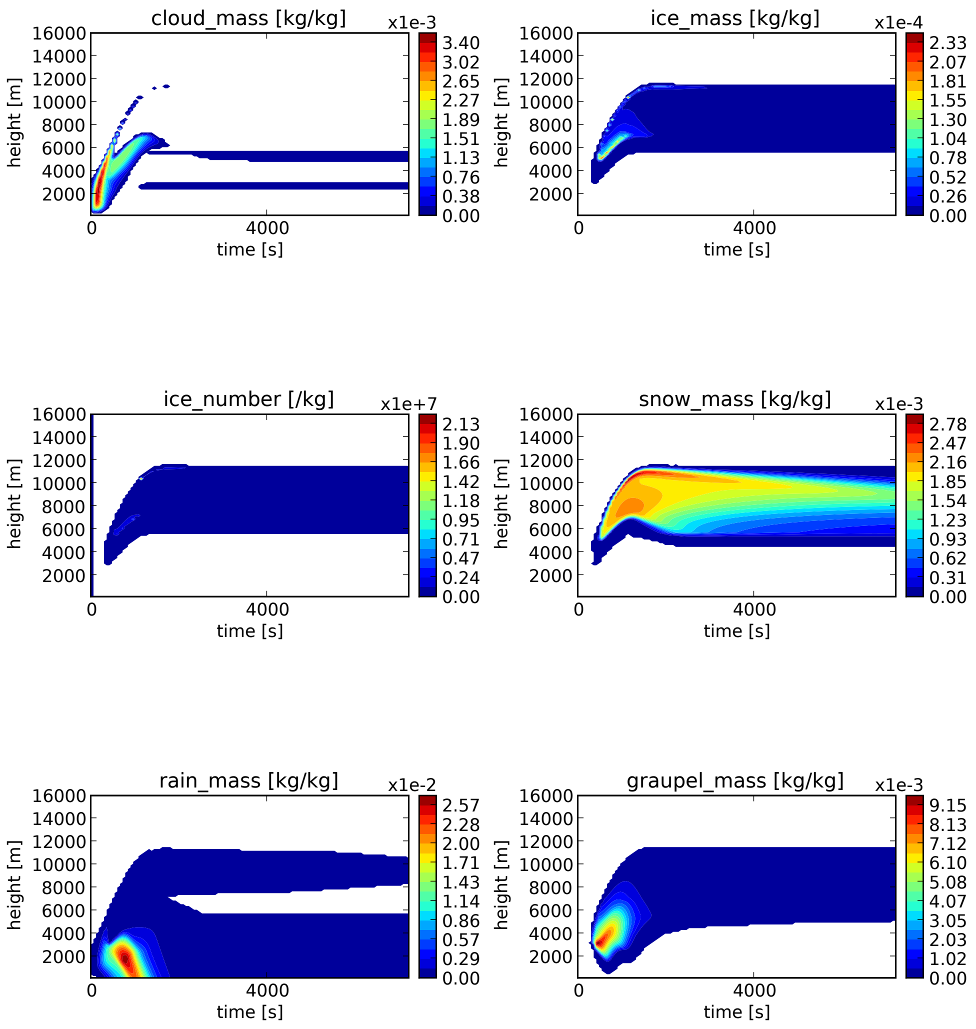

Figure 10: As figure 7 but for the deep transient case 11.

Case 11 is a transient test of a deep convective environment similar to that of the warm case 5. In this case the updraught is given by

$$w(z,t) = w_1 \exp\left[-\left(\frac{s(z,t)}{\sigma}\right)^2\right], \tag{17}$$

with

$$s(z,t) = \sqrt{\frac{(\nu+\eta-1)^2}{2z_3^2} + \frac{(\eta-\nu)^2}{2t_3^2}}, \tag{18}$$

$\sigma = \sqrt{\log(w_1/w_2)}$, $\eta = z/z_2$ and $\nu = t/t_2$.

9.2 2-D Test cases

9.2.1 Warm cumulus case (`namelists/cu_2d.nml`)

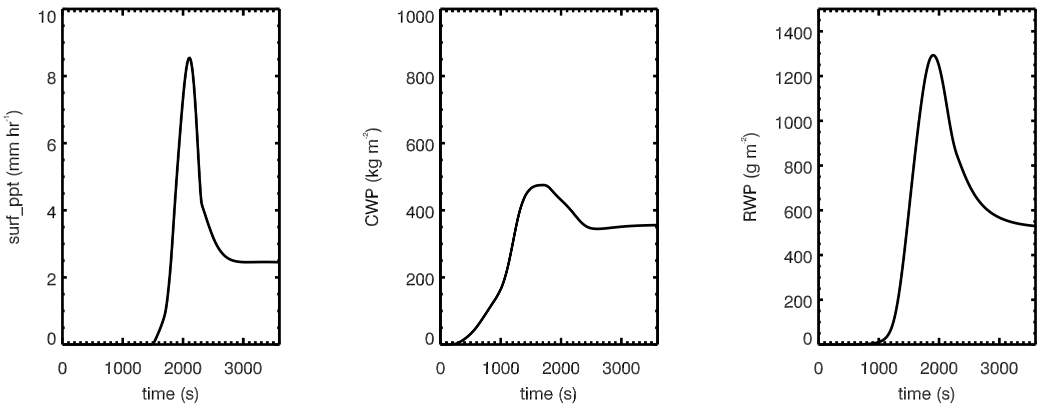

Figure 11: Output from the warm cumulus case for a single moment bulk microphysics scheme. Timeseries of surface precipitation, cloud water path and rain water path.

The cumulus case is the same as that presented in Morrison and Grabowski (2007). The 2-D flow field (supplied by H. Morrison), which varies in time, represents the development of a quasi-idealised shallow convective plume. The initial thermodynamic profiles are based on aircraft data from 10 August 1990 during the Hawaiian Rainband Project (HaRP). The initial profiles represent typical upstream summertime conditions of the windward shore of Big Island, Hawaii. They include 1) a 200 - 300 m deep well-mixed surface layer, 2) lifting condensation level between 600 and 700 m and 3) a weak inversion at about 2 km, with a dry layer above that limits the vertical extent of convection (Morrison and Grabowski, 2007)

The horizontal grid spacing is 50 m over a 9 km horizontal domain (180 grid points) and a 3 km vertical domain (60 grid points). The flow is characterised by low-level convergence and upper-level divergence with a narrow updraft in the centre of the domain. The maximum updraft speed is held constant 1 m/s for the first 15 minutes. This increases to 8 m/s by 25 minutes and then reduces to 2 m/s by 40 minutes.

The simulation time is 1 hour and there is no cloud water in the domain at the beginning of the simulation, i.e. cloud water develops as a result of the flow field evolution.

Periodic boundaries are not used. Instead a mask is applied to the boundaries so the boundaries maintain their initial values. This acts as a source of vapour and theta in the low level convergent flow

9.2.2 Warm stratocumulus case (`namelists/Sc_2d.nml`)

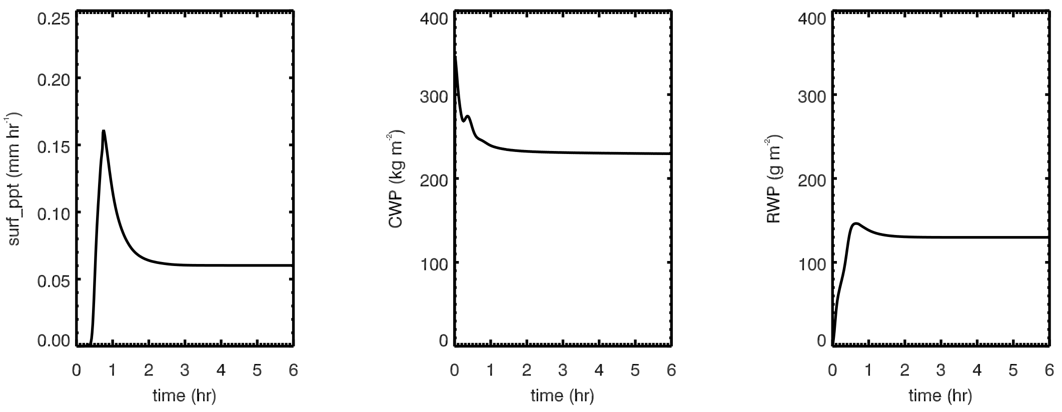

Figure 12: Output from the warm stratocumulus case with latent heat and sensible heat flux of 3 and -3 W m^{-2} respectively, for a single moment bulk microphysics scheme. Timeseries of surface precipitation, cloud water path and rain water path.

As with the cumulus case, the stratocumulus case is the same as that presented in Morrison and Grabowski (2007), with the original test case code provided by H.Morrison. The 2-D flow field, which is invariant in time, consists of a single eddy that spans the entire depth of the domain, with a maximum vertical velocity of 1.7 m/s. The initial conditions are based on an idealised stratocumulus-capped boundary layer, with constant equivalent potential temperature of 288 K. A large scale moisture flux (latent heat flux) into the domain equivalent to either 3 or 30 W m^{-2} is applied to the vapour

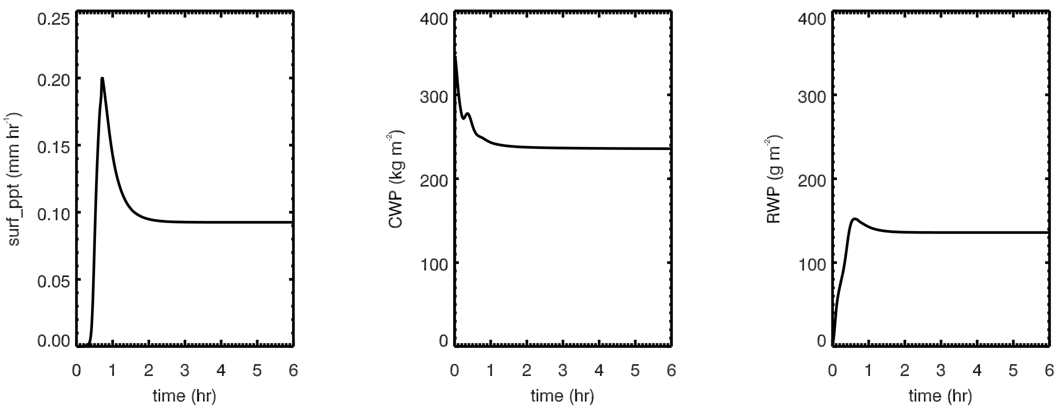

Figure 13: Output from the warm stratocumulus case with latent heat and sensible heat flux of 30 and -30 W m^{-2} respectively, for a single moment bulk microphysics scheme. Timeseries of surface precipitation, cloud water path and rain water path.

mixing ratio. A sensible heat flux of -3 to -30 W m^{-2} is applied to the temperature field to maintain a quasi balance of heat in the domain.

The horizontal grid spacing is 20 m over a 2 km horizontal domain (100 grid points) and a 1040 m vertical domain (52 grid points) but the extent of the Sc is 1km, as in Morrison and Grabowski (2007). The reason for the using 52 points instead of 50 is that some schemes may not include sedimentation for the top layer. As a result of this, hyrometeors and condensate can be advected up into the top layer, but not sedimented. Microphysics occurs in the top level so drops can grow large and then transport can move them back into the lower level, which can lead to numerical errors. Therefore, the extra levels are added to the case to allow processing up to 1000 m, then a mask is applied to the top two levels.

The simulation time is 6 hour with equilibrium being reached in 2 hours.

The initial profiles are supersaturated so cloud water forms on the first timestep of the simulation

Periodic boundaries are used.

	z_1	z_2(m)	z_3(m)	z_4(m)	z_5(m)	z_6(m)	t_1(s)	t_2(s)	t_3(s)	t_4(s)	w_1(ms^{-1})	w_2(ms^{-1})
case 1	3000	-	-	-	-	-	3600	600	-	-	2	-
case 2	3000	-	-	-	-	-	7200	600	-	-	2	-
case 3	3000	-	-	-	-	-	3600	600	1200	-	2	-
case 5	7000	650	1000	2000	3000	4000	5400	1800	3240	3600	7	-
case 6	3000	300	500	1000	1500	2000	4500	1800	3240	3600	3	-
case 7	1000	200	400	170	630	800	8×3600	600	-	-	1	-
case 8	1000	400	-	-	-	-	6×3600	600	-	-	.3	-
case 9	1000	200	400	100	350	450	6×3600	600	-	-	.3	-
case 10	15000	5000	5000	7000	7000	-	12×3600	-	-	-	10	-
case 11	16000	10000	.6	-	-	-	7200	1200	.2	-	16	6

Figure 14: Parameters used in each of the test cases. Each of these can be modified through the `control` namelist.

10 References

- Shipway, B.J and A.A. Hill, 2012, 'Diagnosis of systematic differences between multiple parametrizations of warm rain microphysics using a kinematic framework', Q.J.R. Meteorol. Soc.. doi: 10.1002/qj.1913.

- Field,P. R., A. J. Heymsfield, B. J. Shipway, P. J. DeMott, K. A. Pratt, D. C. Rogers, J. Stith, K. A. Prather, 2012: 'Ice in Clouds ExperimentLayer Clouds. Part II: Testing Characteristics of Heterogeneous Ice Formation in Lee Wave Clouds'. J. Atmos. Sci., 69, 10661079