Simulating idealized cases with the COSMO-model (draft version)

Ulrich Blahak

January 13, 2015

Contents

1 Introduction **1**

2 General concept **2**

3 Getting started **3**

4 Possible domain structures and boundary conditions **6**
4.1 Different types of boundary conditions for idealized 3-dimensional runs 6
4.2 2-dimensional runs 7
4.3 1-dimensional setup ("single column") 8

5 Configuring idealized cases — a cookbook **8**

6 Documentation of implemented features **26**
6.1 Vertical grid levels 26
6.2 Reference atmosphere 26
6.3 Orography and other surface parameters 26
6.4 Constant atmospheric parameters 31
6.5 Soil parameters 31
6.6 Initial fields of temperature, moisture and windspeed 31
6.7 Initial field of pressure 31
6.8 Boundary values for prognostic fields 33
6.9 Artificial convection triggers 33
6.10 Running the soil model 33
6.11 Running the convection scheme 33
6.12 Running the radiation scheme 34
6.13 Miscellaneous features 34

7 Code structure of `src_artifdata.f90` **34**
7.1 Code tree 34
7.2 Subroutine definitions 34

8 Implementing own idealized features into `src_artifdata.f90` **34**

9 Known issues / problems **34**

10 Table of new namelist parameters for idealized cases **36**

Bibliography **48**

1 Introduction

In the last year the implementation of the possibility to conduct idealized runs with the COSMO-model has been completely re-designed. Instead of having several different `src_artifdata.f90`-files for each single particular idealised case (and consequently having to edit and compile a new binary for each new setup and each time a certain parameter has been changed), there is now only one such module, which has been written in a modular fashion and which allows the user to configure his desired idealized setup via namelist-parameters. Or, if the provided mechanisms are not suitable, it is possible to easily extend the code by, e.g., own configurations of initial- and boundary conditions at well-defined places within the code. The modular fashion allows the user to define his or her idealized case freely, from, e.g., simple 2D flow-over-hill configurations to study mountain waves to full 3D runs with more complex orography and certain convection triggers to study the interaction of convective cells and their cold pools with orography and mountain wave flows. LES-like studies with periodic boundary conditions and prescribed surface fluxes are also possible.

Idealized cases can now be run also on single-processor machines like ordinary PCs, if the model domain is configured in a way that the main memory of the system is sufficient. This is especially effective during the process of developing new model code which is not concerned with parallelization: Set up a small test case tailored to test your development, compile with no optimization on your PC, run it and get an answer in a very short amount of time. If you use OpenMPI on your PC, you can even have parallel runs on your desktop!

This document describes how to set up such idealized cases and the relevant new namelist parameters. For all other namelist paramters, we refer to the "normal" COSMO-model User Guide, Schättler (2012). New model users should first work their way through that User Guide before trying to run idealized cases. For beginners, there is also a model tutorial available from the CLM (Climate Limited area Modeling community) webpage, which is provided for the COSMO/CLM/ART Training Course every year.

This document is organized as follows: Section 2 explains the general guidelines for the new implementation, Section 3 gets you started with setting up a simple example run, Section 4 describes the general possibilities for choosing model domains and boundary conditions, Section 5 gives an overview on how to configure idealized cases via the newly provided namelist parameters in a new namelist called `ARTIFCTL`, and Section 7 describes the new code structure of `src_artifdata.f90` in preparation for Section 8, which shows where and how to extend the code by own new idealized features. Some remarks on known issues / problems (Section 9) follow. Last but not least Section 10 with a table of all new namelist parameters rounds out the document.

The COSMO-model distribution comes with a number of exemplary runscripts which configure the namelists and model runs for some common idealized setups. These are described in more detail in Section 5.

2 General concept

As has been mentioned in the introduction, the approach is to have one module for defining the initial and boundary conditions for idealized cases and to enable the user, in a modular fashion, to configure all necessary ingredients for his or her idealized experiment by namelist parameters. In this way, it is possible to easily set up, e.g., parameter sensitivity studies without having to recompile the model every time a parameter has changed.

The following ingredients are provided in the current code for idealized simulations:

- Vertical coordinate type and distribution of levels
- Orography function
- Reference atmosphere
- Surface parameters (e.g., roughness length, albedo, ...)
- Soil parameters (if soil model is turned on)
- Specifying constant surface fluxes (if soil model is turned off)

- Initial fields of the prognostic variables (including a hydrostatic pressure initialization from temperature T resp. moist pot. temperature θ_m, surface pressure p_s and rel. humidity $\overline{H}$).
- Periodic boundary conditions independently choosable for the X- and Y-direction
- Open lateral boundary conditions (in contrast to the periodic conditions, these are at the moment not separable into X- and Y-direction)
- No-slip- or free-slip lower boundary conditions for the windspeed, optionally also no-surface-flux conditions for sensible and latent heat.
- Artificial convection triggers:
  - "Warm bubbles" imposed on the initial atmospheric fields at model start or artificial heating rates in certain limited areas within a specifyable time period during the model run.
  - For runs with soil model: "Heat islands" in the soil as initial condition at model start or artificial limited area heating rates within a specifyable time period during the model run.
  - For runs without soil model: (Temporal) "heat island" at the surface
- Time-constant surface sensible and/or latent heat fluxes, with possibility to add time-constant spatial white noise to these fluxes
- Additional noise to w and T at model start time in the boundary layer (lowest 100 hPa).

There is a number of already implemented features of common test cases and some more not so common things implemented, which may be combined in a modular fashion (e.g., flow over hills combined with artificial convection triggers like "warm bubbles"). It is the user's responsibility that the chosen combination makes any physical sense.

Note: the idealized features so far implemented do not consider Coriolis force! In particular, there is no geostrophic pressure initialization. All setups have only been tried so far without Coriolis force (`lcori = .FALSE.`). Also, the use of the old leapfrog dynamics has not been well tested in this idealized framework.

Concerning the metric terms for earth curvature effects and geographic spherical grid distortion, the user has to decide whether to include them or not (parameter `lmetr` in namelist `RUNCTL`). In most cases, the user wants to run a cartesian grid (`lmetr = .FALSE.`). This is especially recommended if periodic boundary conditions are applied. Otherwise (`lmetr = .TRUE.`), the grid is spherically distorted, and depending on the setup and the type of lateral boundary conditions (in particular periodic conditions), care has to be taken that the model domain is exactly centered around the "real" equator (see Section 3 on how to do this) to avoid (maybe) undesired lateral circulations. Such circulations can arise in case of non-equator-centered grids because of the transformation of the user specified initial wind directions into the rotated lat/lon-grid on the sphere.

3 Getting started

First, compile the model as usual. There are two new preprocessor switches which are relevant for the compilation with respect to idealized simulations:

- `HAS_IOMSG`: This switch enables a Fortran2003 feature which greatly simplifies the debugging of errors in the extensive namelist for the idealized cases (file `INPUT_IDEAL` containing the namelist `ARTIFCTL`, see later in this document). When issuing the `READ` statement for the namelist, as usual errors are catched by an integer code returned by the `IOSTAT` parameter, but also a detailed error message is retrieved via the new `IOMSG` parameter. For Fortran2003 compatible compilers, this feature can be activated by adding something like `-DHAS_IOMSG` to the compiler options (file `FOPTS`), or by activating (uncommenting) the line

  ```
  #define HAS_IOMSG
  ```

  in the header of the source code `src_artifdata.f90`.

- `__PGI_FORTRAN__`: This is relevant for the PGI Fortran compiler when computing uniform random numbers with the standard random number generator of the compiler. In a subroutine called `seed_random_number()` in file `src_artifdata.f90`, there is a call to the system function `DATE_AND_TIME()`. With the PGI-compiler, the data types of the arguments of this function seem to need the specific data type `INTEGER*8`, which is not the case with other compilers. Therefor, the distinction is made by help of the preprocessor switch, and if you work with the PGI-compiler and get problems during compilation with the `DATE_AND_TIME`-function, add `-D__PGI_FORTRAN__` to your compiler options (file `FOPTS`) or activate (uncomment) the line

  ```
  #define __PGI_FORTRAN__
  ```

  in the header of the source code file `src_artifdata.f90`.

After compilation, turn your attention to the definition of certain namelist parameters.

Set `lartif_data = .TRUE.` in namelist `RUNCTL` to activate the idealized mode.

As has been said earlier, normally choose `lmetr = .FALSE.` and `lcori = .FALSE.` (`RUNCTL`), and choose the grid spacing and the domain size in namelist `LMGRID`. The grid spacing parameter is in degrees and can be computed from the desired grid lenghts Δx and Δy and the earth radius $R_E = 6371245\,\mathrm{m}$:

$$\texttt{dlon} = \frac{\Delta x}{R_E} \tag{1}$$

$$\texttt{dlat} = \frac{\Delta y}{R_E} \tag{2}$$

$$\tag{3}$$

If you want to center the domain around a certain domain reference point given by the geographical coordinates (longitude, latitude) = (θ_r, ϕ_r), choose

$$\texttt{startlon\_tot} = -\left(\frac{\texttt{ie\_tot}}{2} - 0.5\right) * \texttt{dlon} \tag{4}$$

$$\texttt{startlat\_tot} = -\left(\frac{\texttt{je\_tot}}{2} - 0.5\right) * \texttt{dlat} \tag{5}$$

$$\texttt{pollon} = \begin{cases} \theta_r - 180^\circ & \text{if } \phi_r \geq 0 \\ \theta_r & \text{if } \phi_r < 0 \end{cases} \tag{6}$$

$$\texttt{pollat} = 90^\circ - |\phi_r| \tag{7}$$

In these equations, you find all the relevant namelist parameters from the namelist `LMGRID` for the specification of the model domain and its geographic location.

However, depending on other settings, these parameters might or might not be of relevance:

- If `lmetr = .FALSE.` and the radiation scheme is not used, only `dlon`, `dlat`, `ie_tot` and `je_tot` are important.
- If `lmetr = .TRUE.` and periodic boundary conditions in y-direction are applied, one is tempted to set `pollat` to 90° ($phi_r = 0$) and apply the above centering, to avoid spurious flow errors at the Nothern and Southern boundaries. However, if `lmetr = .TRUE.`, the initial u and v-components are computed from the specified input wind directions (through namelist parameters or input ASCII-files, see below) as if they were geographical wind components. This means, u and v are transformed to the rotated coordinate system. In combination with periodic boundary conditions in y-direction, this leads to spurious circulations at the Northern and Southern boundaries even if the domain is centered around the true equator. Therefore, in case of periodic y-conditions, do not set `lmetr = .TRUE.`.
- In case of using the radiation scheme (`lrad = .TRUE.`), the geographic location matters for computing the sun zenith angle, the hour angle and the aerosol optical thickness from the worldwide aerosol climatology. These will be computed from these parameters as in a real case simulation,

$$\theta_{rot} = \texttt{startlon\_tot} + (i-1) * \texttt{dlon} \tag{8}$$

$$\phi_{rot} = \texttt{startlat\_tot} + (j-1) * \texttt{dlat} \tag{9}$$

where i and j are the global grid indices in x and y direction. The rotated coordinates of the model grid θ_{rot} and ϕ_{rot} are converted to geographical coordinates by COSMO standard functions as described in the COSMO-documentation (Doms, 2011, Schättler, 2012), and these enter the radiation calculations. Therefore, the parameters `pollon`, `pollat`, `startlon_tot` and `startlat_tot` become important in this case.
Note that in case of periodic boundary conditions in x and/ or y-direction, Eqs. (8) and/ or (9) are replaced by the constant value(s) θ_r and/ or ϕ_r, which are connected to `pollon`, `pollat` by Eqs. (6) and (7).

The next Section 4 briefly describe the different possibilities for the general setup of the model grid and the boundary conditions in the idealized framework. Choose the suitable ones for your purpose.

Finally, create the namelist `ARTIFCTL` in the file `INPUT_IDEAL` (cf. Section 5 and the following sections thereafter), which contains parameters to define all other aspects of your idealized setup (Section 2).

To assist you in setting up all the different parameters in different namelists, you will find a subdirectory `RUNSCRIPTS` in your COSMO-model distribution (at least in the NWP version of the code) with useful material. This directory contains a number of exemplary runscripts, either containing all possible parameters in `ARTIFCTL` or tailored for certain typical idealized cases, each with a particular setup of boundary conditions and general model parameters. These runscripts may serve as starting points for setting up own configurations. The content of this directory is:

- `run_ideal`: it contains the full namelist `ARTIFCTL` with lots of comments, see Section 5 and following sections. It is meant as a reference. It can however be run as is, and it starts the simulation of a three-layered flow (three different Brunt-Vaisala-frequencies in the PBL, free troposphere and tropopause region) past an idealized hill - valley - hill - orography with an orography base height of 500 m (!).

- `run_ideal_empty`: This runscript contains an empty namelist ARTIFCTL and thus uses the defaults for all the namelist parameters to provide a minimal functioning example. These defaults lead to the following setup:

  - model top height 22000 m,
  - flat terrain,
  - no-slip/no-fluxes lower boundary condition,
  - $u = 20\,\mathrm{m/s}$
  - 3 layered-polytrope atmosphere as in `run_ideal` above

  Besides this, soil- and radiation schemes are turned off, radiative (open) lateral boundary conditions are applied and the horizontal grid spacing is 1 km

- `run_ideal_hill`: 2d flow over an idealized hill,

- `run_ideal_hill_2`: similar to `run_ideal_hill`, but contains more parameters. It is intended as a base for the user to experiment a little, e.g., to set up a case with flow over several hills, also taking into account surface friction and surface heat fluxes.

- `run_ideal_wk82`: idealized warm-bubble triggered convection after Weisman and Klemp, 1982

- `run_ideal_wk82_2mom`: the same, but using the Seifert-Beheng two-moment cloud microphysical scheme, which is available for COSMO 5.0, but requires additional modules from subdirectory `LOCAL/TWOMOM`. Due to coding standard issues, these modules are not yet part of the "official" COSMO-model code.

- `run_ideal_asciitest`: to demonstrate the use of ASCII-files to initialize orography, soil- and other constant variables in the model. This script uses some sample data of such ASCII files, which can be generated by running the simple Fortran90-program `gen_soildata_ascii.f90` which can also be found in the subdirectory `RUNSCRIPTS` (see below).

- `run_ideal_cos-soil`: This runscript starts the simulation of a flat terrain with an initial soil temperature perturbation (a circular area with 10 km radius and a +10 K disturbance at its center, diminishing with the $\cos^2$ of the center distance) and using the soil model.

- `run_ideal_hotspot-sfc`: This runscript starts the simulation of a flat terrain with a timec-constant surface temperature perturbation (a Gaussian disturbance with +10 K at its the center, diminishing with the center distance and having a half width radius of 10 km), without using the soil model.

- `run_ideal_les_200m`: shallow convection in LES-like mode, horizontal grid spacing 200 m, constant heat flux of 300 W/m^2 at the surface, initial white noise in the PBL on temperature and vertical velocity, periodic boundary conditions. The alternative turbulence scheme from Herzog et al., 2002b, Herzog et al., 2002a, is used ("dry" version `itype_turb = 7`) in 3D-mode (`l3dturb = .TRUE.`, `l3dturb_metr = .TRUE.`). Alternatively, the "moist" version `itype_turb = 8` can be used, which differs from "7" only in that it uses liquid water potential temperature instead of potential temperature.

- `run_ideal_restart`: yes, it is possible to do idealized restart runs!

- `gen_soildata_ascii.f90`: Fortran90 program to generate exemplary ASCII data files, which can be used to initialize orography and soil parameters. These are meant to demonstrate the required ASCII-format for such files. This should assist the user in creating and applying realistic orography and soil parameters for semi-realistic cases. More information can be found in Section 5.

- `raso_wk_q14_u05.dat`: Exemplary data file for input radiosonde data for temperature, moisture and wind initialization, to demonstrate the required ASCII-format. More information can be found in Section 5.

Theses runscripts are meant to be easily portable to other computer architectures. There are platform specific sections for certain supercomputers (NEC-SX8r/NEC-SX9 of DWD, IBM at DKRZ Hamburg, a Linux-Cluster at KIT, CRAY-XC30 at DWD) and for an ordinary Linux-PC with or without OpenMPI. For using OpenMPI on your Linux-PC, you need multi-core CPUs and the `openmpi` and `openmpi-devel` packages from your Linux-distribution. Compile with `mpif90` (which is most likely an MPI-frontend to `gfortran`) and start your executable with `mpirun -np <num_proc>`.

If you want to hook in a different platform, copy the code for the already present entry most similar to your machine and change the relevant parameter settings.

After this, fill out the user parameter section at the beginning (choose machine, number of processors, name and path of executable, output directory, etc.), adapt the namelists and start the runscript.

4 Possible domain structures and boundary conditions

As has been said before, for most idealized simulations it should be ok to set `lmetr = .FALSE.`. This leads to idealized runs on quasi-cartesian grids. If, for a particular case, it should be important to take into account the earth curvature terms in the model equations (`lmetr = .TRUE.`), care has to be taken to avoid spurious circulations particularly at the Southern and Northern model boundaries. This may happen if periodic boundary conditions are chosen. In this respect, relaxation conditions and "open" boundary conditions seem to be not so problematic.

If `lmetr = .FALSE.`, one can still take in to account the Coriolis force. The rationale is:

- `lcori=.FALSE.` $\Longrightarrow f = 0$

- `lcori=.TRUE.` $\Longrightarrow f = 2\sin(45°)$

The second case presents the popular f-plane approximation. However, it is important to know that no geostrophic pressure initialization is currently available for model initialization. This means, that there is no pressure-balanced wind field at model start and the wind direction is likely to change with time as the model state tries to adjust to an equilibrium between inertia, fricional, pressure gradient- and Coriolis forces. Depending on the boundary conditions, this might lead to unrealistic behaviour, especially for relaxation- and periodic boundary conditions!

The next subsections summarize the possibilities for specifying the boundary conditions for idealized cases.

4.1 Different types of boundary conditions for idealized 3-dimensional runs

The boundary conditions for 3-dimensional runs are summarized here (`l2dim = .FALSE.`). 2-dimensional runs will be discribed separately in Subsection 4.2.

Lateral relaxation towards the initial state, similar to real-case relaxation conditions, except that the boundary conditions do not vary in time:

```
l2dim = .FALSE.
lperi_x = .FALSE.
lperi_y = .FALSE.
lradlbc = .FALSE.
```

The result is an exponential blending function in the lateral relaxation zone with choosable width `rlwidth`. Choose e-folding width of relaxation zone `rlwidth` properly (should be about 7 grid points, but some experimentation might be needed depending on the case). Choose inverse relaxation time scale `crltau_inv` properly (unit: one over multiple of `dt`, default 1.0).

Lateral relaxation can be combined with periodic BCs:

For example, periodic in X-dir. and relaxation in Y-dir.:

```
l2dim = .FALSE.
lperi_x = .TRUE.
lperi_y = .FALSE.
lradlbc = .FALSE.
```

or periodic in Y-dir. and relaxation in X-dir.:

```
l2dim = .FALSE.
lperi_x = .FALSE.
lperi_y = .TRUE.
lradlbc = .FALSE.
```

Full periodic BCs can be achieved by:

```
l2dim = .FALSE.
lperi_x = .TRUE.
lperi_y = .TRUE.
lradlbc = .FALSE.
```

"Open" boundary conditions (wave-absorbing conditions):

```
l2dim = .FALSE.
lperi_x = .FALSE.
lperi_y = .FALSE.
lradlbc = .TRUE.
```

Open BCs cannot be combined with other BCs at the moment! And open BCs might need a little bit lateral boundary relaxation to damp away any small wave reflections at the boundaries. This can be achieved by typically setting

```
relax_fac = 0.02.
```

which is the reduction factor for the above "normal" lateral boundary relaxation. `relax_fac` should be a small value on the order of a few percent. If `relax_fac` > 0, the above namelist parameters for lateral relaxation take effect (`rlwidth`, `crltau_inv`). But again, this might need a little experimentation. Perhaps your case might work without this!

Constant inflow boundary conditions:

Not implemented yet, but could be realized by using the "open" boundary conditions above and overlaying the constant inflow boundary conditions hard-coded in the subroutine `gen_bound_data()` in `src_artifdata.f90`.

"Channel" boundary conditions:

Not implemented yet. These conditions would set the v component to 0 at the Northern and Southern domain boundary and would probably involve a lateral relaxation layer.

Constant horizontal flow at the model top to "drive" the flow below:

Not implemented yet.

4.2 2-dimensional runs

Beside 3-dimensional runs, also 2-dimensional runs are possible in the fashion of an X-Z-slice, where all gradients in Y-direction are disregarded in the model equations. This is achieved by setting

`l2dim = .TRUE.`
`nprocy = 1`
`lperi_y = .FALSE.` (important! Cannot be `.TRUE.` in this case!)
`je_tot = 2*nboundlines+1` (usually 7, because `nboundlines` usually 3)

The actual computations are done only at $j =$ `nboundlines` $+ 1$. Boundary lines in Y-direction are simply overwritten with the values at this j-location in every timestep, so that all gradients in Y-direction become 0.

2-dimensional runs can be performed using the following BCs in X-direction:

Lateral relaxation:

`lperi_x = .FALSE.`
`lperi_y = .FALSE.`
`lradlbc = .FALSE.`
Choose `rlwidth` and `crltau_inv` properly

Periodic:

`lperi_x = .TRUE.`
`lperi_y = .FALSE.`
`lradlbc = .FALSE.`

"Open" (wave reflection absorbing):

`lperi_x = .FALSE.`
`lperi_y = .FALSE.`
`lradlbc = .TRUE.`
Choose `relax_fac` properly as well as `rlwidth`, `crltau_inv`

4.3 1-dimensional setup ("single column")

A simple "single column mode" can be achieved by setting (in the respective namelists)

To achieve this, set
`l2dim = .FALSE.`
`lperi_x = .TRUE.`
`lperi_y = .TRUE.`
`lradlbc = .FALSE.`
`ie_tot = 2*nboundlines+1` (usually 7, because `nboundlines` usually 3)
`je_tot = 2*nboundlines+1` (usually 7)

This is however only possible for serial runs using 1 processor (`nprocx = 1`, `nprocy = 1`)!

The actual computations are done only at $i = j =$ `nboundlines` $+ 1$. Boundary lines in X- and Y-direction are simply overwritten with the values at this i/j-location in every timestep, so that all horizontal gradients become 0.

5 Configuring idealized cases — a cookbook

As previously mentioned, idealized simulations are to be set up by specifying a number of namelist parameters. These parameters are part of the namelist `ARTIFCTL`, which has to reside in the file `INPUT_IDEAL` in the directory from where the model run is initiated.

In the following, a printout of the complete namelist `ARTIFCTL` is given, as distributed with the exemplary COSMO-model job runscript `run_ideal` in the source code tree, subdirectory `RUNSCRIPTS`. This namelist includes a cookbook-like documentation, along with all implemented namelist parameters. The features behind these parameters will be explained in more detail in the following sections.

If you click on one of the parameters (blue text in red boxes), you jump to the respective entry in the namelist parameter table in Section 10, where you can find additional informations on the data type, the description and the default values.

For defining your own idealised setup, you may start from the following namelist and modify it according to your needs. If you find that something what you need is still missing, you may proceed by modifying the source file `src_artifdata.f90`. To assist you in doing so, the Section 7 contains more information about the code structure and a calling tree of the different subroutines, from which you should be able deduce where to hook in your special needs (Section 8 contains more informations on this topic).

```
 &ARTIFCTL
!
!=============================================================
!=============================================================
! Namelist file for idealized simulations:
!=============================================================
!=============================================================
!
!
!=============================================================
! 0) Debug mode
!=============================================================
!
! In debug mode, one gets additional runtime messages
! from the code, which might be useful for debugging.
! To enable the debug mode, set below ldebug_artif = .true.
! and idbg_level to some value > 0.
!
! Currently implemented messages/mechanisms depending on idbg_level:
!
! idbg_artif_level > 0 : print the subroutine name at the
!                        beginning of each (major) subroutine
! idbg_artif_level > 3 : additionally, write ASCII-files
!                        (or BIN-files on the NEC) containing
!                        the T- and QV- increment resp. heating rate for each artif.
!                        temperature-, moisture or heating rate disturbance
!                        triggered in the simulation.
!                    *** On the NEC, you need the conversion program
!                        "bin2ascii_convrates3d.f90" by Ulrich Blahak to
!                        generate ASCII-files from the BIN-files ***
! idbg_artif_level > 4 : print additional checking output for the iterative hydrostatic
!                        pressure initialization
!
ldebug_artif = .true.,
!
  idbg_artif_level = 5
```

```
!
!
!
!=============================================================
! 1) Parameters for the reference atmosphere
!    (DO NOT TOUCH, UNLESS YOU ARE SURE WHAT YOU ARE DOING!):
!=============================================================
!
! irefatm = 1 : old reference atmosphere, based on a constant
!               logarithmic temperature gradient;
!    relevant : p0sl, t0sl, dt0lp
!
! irefatm = 2 : (RECOMMENDED) new reference atmosphere, based on an exponential
!               temperature profile with surface temperature t0sl,
!               asymptotic stratosphere temperature t0sl - delta_t,
!               and scaling height h_scal;
!    relevant : p0sl, t0sl, delta_t, h_scal
!
! irefatm = 3 : new reference atmosphere with constant dry
!               Brunt-Vaisala-frequency bvref;
!    relevant : p0sl, t0sl, bvref
!
irefatm = 2,
  p0sl = 1e5,        ! reference surface (z = 0.0 m) pressure [Pa]
  t0sl = 300.0,      ! reference surface temperature [K]
  dt0lp = 42.0,      ! logarithmic temperature gradient [K]
  delta_t = 75.0,    ! temperature diff. surface - stratosphere [K]
  h_scal = 10000.0, ! e-folding height of exponential adjustment to the stratosphere temp. [m]
  bvref = 0.01,      ! constant Brund-Vaisala-frequency for irefatm=3 [1/s]
!
!
!
!=============================================================
!=============================================================
!
!=============================================================
! 2) Variables for the vertical coordinate specifications:
!=============================================================
!
! The specification of the vertical coordinate levels is as follows:
!
! First, choose the general type of coordinate specification by:
!
! ivctype = 1 : pressure based coordinate (sigma-type, i.e., p/psurf)
! ivctype = 2 : height based coordinate   (RECOMMENDED)
! ivctype = 3 : height based SLEVE coordinate
!
! Second, for each of these coordinate types, you may choose
! coordinate subtypes for specifying the the type of the vertical
! grid distance calculation by the character string "zspacing_type",
! and further namelist parameters for each subtype define the
! specific details. The latter are listed under "RELEVANT" below.
! The "raw" height level values specified by the below settings
! are valid for flat terrain and are blended into terrain following
! coordinates in a way that the terrain influence decreases linearily
! with height, up to the height "vcflat", above which the levels become flat.
!
! for ivctype = 1:
!
!    zspacing_type = 'predefined' or 'vcoordvec':
!
!        'predefined' -- take predifined vertical levels
!                        depending on the setting of ke;
!                        for this, ke currently may be 20, 32, 35 or 40.
!
!                       RELEVANT:  ke, vcflat
!
```

```
!         'vcoordvec'  -- the values in namelist vector "vcoordvec"
!                         will be taken for the full levels (up to 999 values).
!                         For this, the number of values larger than -999.99
!                         has to be equal to ke+1. Values have to be
!                         monotonically increasing between about 0.02 and 1.0
!
!                        RELEVANT:  ke, vcoordvec, vcflat
!
! for ivctype = 2:
!
!    zspacing_type = 'predefined', 'linear', 'galchen' or 'vcoordvec':
!
!         'predefined' -- take predifined vertical levels
!                         depending on the setting of ke;
!                         for this, ke currently may be 20, 32, 35 or 40.
!
!                        RELEVANT:  ke, vcflat
!
!         'linear'     -- vertical equidistant grid, full levels starting at 0.0.
!                         dz is given by zz_top / ke.
!
!                        RELEVANT:  zz_top, ke, vcflat
!
!         'galchen'    -- Gal-Chen coordinate (see COSMO documentation),
!                         full levels starting at 0.0.
!                         The exponent in the Gal-Chen formula can be
!                         given as namelist parameter exp_galchen; a typical
!                         value is, e.g., 2.6, which leads to a moderately increasing
!                         grid distance with height.
!
!                        RELEVANT:  zz_top, ke, exp_galchen, vcflat
!
!         'vcoordvec'  -- the values in namelist vector "vcoordvec"
!                         will be taken for the full levels (up to 999 values).
!                         For this, the number of values larger than -999.99
!                         has to be equal to ke+1. Values have to be
!                         monotonically decreasing between model top and 0.0
!
!                        RELEVANT:  ke, vcoordvec, vcflat
!
! for ivctype = 3 (SLEVE-coordinate):
!
!        same possibilities for "zspacing_type" and the RELEVANT other
!        namelist parameters as for ivctype = 2;
!        however, the blending of the orography into flat surfaces
!        above "vcflat" are determined by the further parameters
!        "nfltvc", "svc1" and "svc2".
!
!-------------------------------------------------------------------------
! NOTE: "vcflat" has to be smaller than the lower height of the
!       upper relaxation layer "rdheight" !!!
!-------------------------------------------------------------------------
!
ivctype = 2,
  zspacing_type = 'galchen',! sub-type of coordinate spec.
  exp_galchen = 2.6,         ! exponent in the Gal-Chen formula
  vcflat = 11000.0,          ! height, above which coordinate levels become flat [m]
  zz_top = 23000.0,          ! height of model top, if it has to be specified explicitly [m]
  vcoordvec = -999.99,-999.99   ! vert. coordinate vector (# = ke+1, max. 999 values)
  nfltvc = 100,              ! number of filter appl. for SLEVE coordinate
  svc1 = 8000.0,             ! "coarser" length scale for SLEVE coordinate [m]
  svc2 = 5000.0,             ! "fine" length scale for SLEVE coordinate [m]
!
!
!
!=========================================================================
!=========================================================================
```

```
!
!
!==========================================================
! 3) Initialization of the artificial orography and soil parameters:
!==========================================================
!
! The specification of the orography is done as follows:
!==========================================================
!
! "Realistic" orography might be read from an external
! ASCII-file, if the switch "linit_realoro=.TRUE.".
! Otherwise, a constant base height "href_oro" is set (default is 0.0).
!
! If orography is read from a file, the following parameters
! are also RELEVANT:
!
!     orofile        (string)   -- Path and name of the orogaphy file
!
! FORMAT OF THE FILE:
!         - arbitrary number of header lines (starting with '#' or '!')
!         - 1 line with field dimensions i,j
!         - 1 long column with the data (first index varies first).
!
! FOR EXAMPLE:
!
!>> BEGIN ASCII-FILE:
! # orography height [m]
! 461 421
! 0.001
! 0.002
! 0.001
! ...
!<< END ASCII-FILE:
!
!
!   The locations of the height values in the file correspond to grid point values
!   of the model. No interpolation is done. The domian covered by the
!   file might be larger than the actual model domain.
!   Orography is then centered in a way that the mid - point of the
!   rotated grid (the grid defined in the namelist LMGRID) equals the
!   center point of the domain in the file, i.e.,
!     ie_tot/2+1 = ie_tot_file/2+1    ,
!     je_tot/2+1 = je_tot_file/2+1    .
!
!   It is possible to further shift the model domain relative to
!   the center file domain by:
!
!   i_shift_realoro  (integer)   -- Shift in X-dir. of the domain (in GPs)
!   j_shift_realoro  (integer)   -- Shift in Y-dir. of the domain (in GPs)
!
!   Then, the relation between the center points is
!     ie_tot/2+1 - i_shift_realoro = ie_tot_file/2 + 1    ,
!     je_tot/2+1 - j_shift_realoro = je_tot_file/2 + 1    .
!
!
!=========================
linit_realoro = .false.,
!=========================
!
! For linit_realoro = .true.:
  orofile = 'dummy.dat',
  i_shift_realoro = 0,
  j_shift_realoro = 0,
!
! For linit_realoro = .false.: Orography base height (normally 0.0, but
!   might be necessary to set it > 0.0 if valleys are specified below)
  href_oro=500.0,
```

```
!
!
! On top of either the constant base height (default is 0.0) or the orography
! from the file, one may (additionally) define artificial hills and/or valleys.
! This is dependent on the switch itype_topo. Current implemented settings are:
!
! itype_topo = 0:      No additional hills/valleys
! itype_topo = 1:      Package to define arb. number of hills/valleys,
!                      which depends on namelist settings described below.
!
!=====================
itype_topo = 1,
!=====================
!
!    ------------------------------------------------------------------
!    For itype_topo=1, the following namelist switches take effect:
!    ------------------------------------------------------------------
!
!    NOTE: all of the following switches can be comma-separated lists,
!          so that each member of the list corresponds to one hill/valley.
!          Currently a max. of 50 members are allowed (50 hills/valleys).
!
!    Generally, hills/valleys can be 3D (shape can be Gaussian or
!    "bellshaped", two horizontal major axes define
!    an elliptic shape of the height level curves, max. height can be defined)
!    or 2D (Gaussian or "bellshaped" in X-direction, elongated in
!    Y-direction with a definable length and "rounded" edges),
!    they can be of asxmetric shape in X- and Y-direction, and
!    additionally, they can be horizontally rotated by a certain angle
!    (mathematically positive).
!
!    A valley is simply defined by a negative hill height below!
!
!
!   lhill is the "master" switch. Set as much elements to true as much
!     hills/valleys you want, e.g., lhill=.true.,.true.,.true., generates
!     3 hills/valleys.
  lhill = .true., .true., .true.,
!
!       lhill_2d determines if the resp. hill/valley is 2-dim. (oriented along the
!       y-axis) or 3-dim. in the case of lhill_2d=.false.:
      lhill_2d = .true.,.true.,.true.,
!
!       shape of mountain:
!         - 'gauss' (symmetric)
!         - 'bellshaped' (symmetric)
!         - 'gauss-asym' (asymmetric)
!         - 'bellshaped-asym' (asymmetric)
      hill_type = 'gauss','gauss','gauss',
!
!       Coordinates of mountain center point(s) in GPs, can be real numbers, not just inte
      hill_i = 80.5, 60.5, 100.5,
      hill_j = 80.5, 80.5, 80.5,
!
!     hillheight (in m):
!       top height of the mountain(s) (if negative, depth of the valley(s)):
      hillheight = 500.0, -500.0, 750.0,
!
!     hill_rotangle (in degrees):
!       Clockwise rotation angle(s) around the mountain center(s)
!       with respect to its horizontal orientation:
      hill_rotangle = 0.0,0.0, 0.0,
!
!     zhillcutfact (dimensionless, range 0...infinity):
!       Factor to limit the mountain(s) width to a finite value by
!       removing the tails of the mountain(s). This works as follows:
!       If, e.g., zhillcutfact = 0.02, the mountain height is increased by
```

```
!       0.02*hillheight, and subsequently 0.02*hillheight are chopped away from
!       the bottom, so that the mountain looses its tails and has the same
!       height as before. (for valleys, it is similar but chop from the top).
!       NOTE: this slightly distorts the mountain shape
!       and leads to a difference between the "true" halfwidths and the
!       specified half width parameters below.
      zhillcutfact = 0.02, 0.02, 0.02,
!
!     hill_combineaction:
!       This integer flag determines the way in which the mountain is combined with the
!       previously present mountains (mountains are processed in the order of apperance
!       in the mountian(s) list):
!          hill_combineaction = 1 :    add to the previous orography (subtract in the case of
!          hill_combineaction = 2 :    take the maximum (minimum in case of a valley)
!                                      of this mountain and the previous mountain(s)
      hill_combineaction = 1,
!
!     hill_width_x (in m):
!       Half width radius to mountain center in X-direction:
!       (both for 2D- and 3D-mountain(s)):
      hill_width_x = 10000.0, 20000.0, 15000.0,
!
!     hill_width_y (in m):
!       For 3D-mountain(s) (lhill_2d=.false.)
!       - Half width radius to mountain center in Y-direction:
!       For 2D-mountain(s) (lhill_2d=.true.)
!       - 1/2 overall width, centered around the mountain center in Y-direction:
      hill_width_y = 300000.0, 300000.0, 300000.0,
!
!     hillsideradius_y (in m):
!       Effective for 2D-mountain(s) (lhill_2d=.true.) only:
!       To round the "sharp" edges of a finite 2D-mountain
!       at its northern and southern sides,
!       the half of a 3D-mountain with X-halfwidth "hillwidth_x"
!       is added at each side, which has the following
!       half width radius in Y-dir.:
      hillsideradius_y = 10000.0, 10000.0, 10000.0,
!
!     Asymetric hills: (if chill_type = 'gauss-asym' or 'bellshaped-asym')
!       Asymmetry factors for the hill(s)/valley(s) in X- and Y-direction, defined
!       as the ratio between the west (south) and east (north) side half widths.
!       E.g., a factor of hillasym_x=1.5 means that the west side is 1.5 times as wide
!       as the east side, and that the west side has a half width radius of
!       hillwidth_x.
      hillasym_x = 1.0, 1.0, 2.0
      hillasym_y = 1.0, 1.0, 1.0,
!
!
!=======================================================
!=======================================================
!
!
!=============================================================
! 4) Now come the soil parameters:
!=============================================================
!
! The first thing to note is that, regardless of using the soil model
! or not, the surface temperature t_s (for snow free areas!)
! as a baseline, is initialized to be the atmospheric temperature at the surface,
! so that minimal sensible heat fluxes would occur. If the soil model
! is used, it might, depending on the below settings, overtaken also
! for all the soil levels at land points. If it is not overtaken by the soil levels,
! it will be reset to the value of the uppermost soil level.
! In any case, it is relevant at sea and lake points.
!
! However, this baseline initialization for t_s can be replaced by a
! user chosen constant value t_surf_c or by values from a file "tsurffile",
```

```
! depending on the setting of the switch "itype_soil_tw", see below.
!
! ------------------------------------------------------------------
!
! But first, the switch "itype_soil_c" determines how the (more or less)
! time constant soil parameters are initialized (these are held
! constant during the simulation):
!
!     itype_soil_c = 1 :  spatial and time constant values are specified via the below
!                         namelist parameters for constant fields.
!
!     itype_soil_c = 2 :  time constant and spatially varying fields are read from 2D ASCI
!                         (same format as orography file)
!
!
! If itype_soil_c = 2, then files for the following constant parameters are necessary
!     (probably more in the future):
!
! z0      : roughness length [m] (Give filename in NL-parameter "z0file"       )
! fr_land : land fraction    [-] (                               "frlandfile"   )
! plcov   : plant cover      [-] (                               "plcovfile"    )
! lai     : leaf area index  [-] (                               "laifile"      )
! soiltyp : type of soil     [-] (                               "soiltypefile" )
! if lsoil=.true.:
!   rootdp  : root depth       [m] (                             "rootdpfile"   )
! if lforest=.true.:
!   for_e   : aerea fraction of evergreen forest   [-] (         "forefile"     )
!   for_d   : aerea fraction of deciduous forest   [-] (         "fordfile"     )
! if seaice=.true. .or. llake=.true.:
!   h_ice   : ice thickness   [m] (                              "hicefile"     )
! if lsso=.true.:
!   sso_stdh : std. dev. of sub-grid scale orography [m] (       "ssostdhfile"  )
!   sso_gamma: anisotr. of sub-grid scale orography [-] (        "ssogammafile" )
!   sso_theta: angle betw. princ. axis of orogr. and E [rad] (   "ssothetafile" )
!   sso_sigma: mean slope of sub-grid scale orography [-] (      "ssosigmafile" )
!
! NOTE: "soiltypefile" is read in any case, not only for lsoil=.true., because
!       some other parameterizations (e.g., albedo computation in the radiation
!       scheme) depend on it.
!
! ------------------------------------------------------------------
!
! Now, the switch "itype_soil_tw" determines how the (more or less)
! time varying soil parameters are initialized (these have a time
! index in the model and may change over time):
!
!     itype_soil_tw = 1 :  spatially constant values are specified via the below
!                          namelist parameters for varying fields.
!
!     itype_soil_tw = 2 :  varying fields are read from 2D ASCII-files
!                          (same format as orography file)
!
! If itype_soil_tw = 2, then files for the following time varying
!   parameters are necessary:
!
!   t_surf  : baseline surface temperature [K] (                 "tsurffile"  )
! if lsoil=.true.:
!   t_soil  : soil temperature [K] (Give filename in NL-parameter "tsoilfile"  )
!   wf_soil : soil water saturation [-] (                        "wfsoilfile" )
!   t_snow  : snow temperature [K]      (                        "tsnowfile"  )
!   w_snow  : snow water equivalent [m H2O]  (                   "wsnowfile"  )
!   w_i     : interception storage on plants [m H2O]   (         "wifile"     )
!
! ------------------------------------------------------------------
!
!
! These ASCII-files should be generated by the users themselves. This is
```

```
! achievable with some phantasy, the DWD web interface, int2lm, wgrib, Fortran,
!   matlab, idl, ...
!
! For example, ASCII files may be extracted from the extpar-grib-files
! by wgrib or the script "grib_decode" from Ulrich Blahak.
!
!
! FORMAT OF THE FILES:
!         - arbitrary number of header lines (starting with '#' or '!')
!         - 1 line with field dimensions i,j
!         - 1 long column with the data (first index varies first).
!
! FOR EXAMPLE:
!
!>> BEGIN ASCII-FILE:
! # roughness length [m]
! 461 421
! 0.001
! 0.002
! 0.001
! ...
!<< END ASCII-FILE
!
!
!=============================
itype_soil_c = 1,
!=============================
!
! For itype_soil_c = 1: Constant fields
!=============================================
  z0_c       = 0.01,         ! z_0 in m
  fr_land_c  = 1.0,          ! land fraction, dimensionless
  soiltyp_c  = 3.0,          ! soiltype, dimensionless
  plcov_c    = 0.6,          ! plant cover, dimensionless
  lai_c      = 3.0,          ! LAI, dimensionless
  rootdp_c   = 0.7,          ! root depth in m
  for_e_c    = 0.2,          ! area fraction of evergreen forests
  for_d_c    = 0.2,          ! area fraction of decidious forests
  h_ice_c    = 0.1,          ! ice thickness in m
!
! itype_soil_c = 2: File names for 2D ASCII-reading of fields:
!=============================================================
  z0file        = 'dummy.dat',  ! z_0 in m
  frlandfile    = 'dummy.dat',  ! land fraction, dimensionless
  soiltypefile  = 'dummy.dat',  ! soiltype, dimensionlessx
  plcovfile     = 'dummy.dat',  ! plant cover, dimensionless
  laifile       = 'dummy.dat',  ! LAI, dimensionless
  rootdpfile    = 'dummy.dat',  ! root depth in m
  forefile      = 'dummy.dat',  ! area fraction of evergreen forests
  fordfile      = 'dummy.dat',  ! area fraction of decidious forests
  hicefile      = 'dummy.dat',  ! ice thickness in m
  ssostdhfile   = 'dummy.dat',  ! std. dev. of sub-grid scale orography [m]
  ssogammafile  = 'dummy.dat',  ! anisotr. of sub-grid scale orography [-]
  ssothetafile  = 'dummy.dat',  ! angle betw. princ. axis of orogr. and E [rad]
  ssosigmafile  = 'dummy.dat',  ! mean slope of sub-grid scale orography [-]
!
!
!=============================
itype_soil_tw = 1,
!=============================
!
! For itype_soil_tw = 1: Constant fields
!=============================================
  t_surf_c   = -1.0,      ! baseline t_s (if < 0, the atmosphere temperature at the surface is u
  t_soil_c   = -1.0,      ! t_soil in K (if < 0, t_s is taken instead) for all soiltypes except
  wf_soil_c  = 0.3,       ! soil water saturation, dimensionless (0 ... 1)
  t_snow_c   = -1.0,      ! t_snow in K (if < 0, t_s is taken instead)
```

```
  w_snow_c   = 0.0,       ! snow water equivalent in m H2O
  w_i_c      = 0.0,       ! interception storage on plants in m H2O
  t_ice_c    = -1.0,      ! T at the snow-ice or air-ice interf. in K (if < 0, t_s is taken in
  t_water_c  = -1.0,      ! T at the water-ice or water-air interface in K (if < 0, t_s is tak
!
! itype_soil_tw = 2: File names for 2D ASCII-reading of fields:
!=======================================================================
  tsurffile  = 'dummy.dat',   ! baseline t_s (for values < 0, the atmosphere temperature at
                                ! Should also contain the correct t_s of water/sea ice point
  tsoilfile  = 'dummy.dat',   ! t_soil in K (for values < 0, t_s is taken)
  wfsoilfile = 'dummy.dat',   ! soil water saturation, dimensionless (0 ... 1)
  tsnowfile  = 'dummy.dat',   ! t_snow in K (for values < 0, t_s is taken)
  wsnowfile  = 'dummy.dat',   ! snow water equivalent in m H2O
  wifile     = 'dummy.dat',   ! interception storage on plants in m H2O
!
!
!
!=======================================================================
!=======================================================================
!
!
!=======================================================================
! 5) Variables for the initialization of the (thermo)dynamic profiles:
!=======================================================================
!
! Initialisation of (thermo-)dynamic profiles:
!
! The possibilities are reading from a radiosonde file or specifying
! T-, Qv- and wind-profiles analytically. Here it is possible to
! mix file reading and analytic specification in the following ways,
! determined by the switch "itype_artifprofiles":
!    itype_artifprofiles = 1 :   file only
!    itype_artifprofiles = 2 :   analytic only
!    itype_artifprofiles = 3 :   T / Qv analytic; U / V from file
!    itype_artifprofiles = 4 :   T / Qv from file, U / V analytic
!
! For itype_artifprofiles = 1/3/4, one has to give a radiosonde file name in "rasofile".
!   (example: raso_wk_q14_u05.dat in this directory)
!
! This text file is expected to have a certain format:
!   - arbitrary number of header lines (starting with '#' or '!')
!   - 1 column description line (NOT starting with '#' or '!')
!   - arbitrary number of data lines,
!     line(s) of space separated numbers as follows (without the "! "),
!     each of which is read with the "*" format
!
! >>(begin textfile)
! #
! #
! #
! P [hPa],    Z [m],     T [K],   Dewp [K], Relhum [%], r [g/kg],  WS [m/s],  WD [deg]
!  1000.0000      0   300.0000   292.4112    63.1962   14.19878    0.00000   270.0000
!   988.7517    100   299.1311   292.2245    65.7473   14.19415    0.16660   270.0000
!   977.5984    200   298.2979   292.0383    68.2801   14.19016    0.33284   270.0000
!   966.5405    300   297.4824   291.8525    70.8582   14.18678    0.49834   270.0000
!   955.5781    400   296.6796   291.6672    73.5007   14.18399    0.66274   270.0000
!   944.7110    500   295.8868   291.4822    76.2191   14.18176    0.82570   270.0000
!   933.9390    600   295.1022   291.2976    79.0218   14.18008    0.98688   270.0000
!   923.2617    700   294.3247   291.1134    81.9164   14.17893    1.14594   270.0000
!   912.6788    800   293.5534   290.9296    84.9094   14.17830    1.30260   270.0000
!   902.1900    900   292.7876   290.7461    88.0072   14.17817    1.45656   270.0000
!   891.7948   1000   292.0267   290.5629    91.2161   14.17853    1.60756   270.0000
!   881.4928   1100   291.2702   290.3800    94.5423   14.17937    1.75536   270.0000
!   871.2825   1200   290.5169   289.8368    95.7824   13.85286    1.89974   270.0000
!   861.1619   1300   289.7662   289.0174    95.3386   13.29073    2.04052   270.0000
!   851.1299   1400   289.0184   288.2001    94.8862   12.74963    2.17751   270.0000
!   841.1862   1500   288.2734   287.3847    94.4256   12.22870    2.31059   270.0000
```

```
!        ...   ...        ...        ...        ...        ...        ...        ...
! <<(end textfile)
!
! *NOTE 1*: ONLY THE COLUMNS WITH Z, T, RELHUM, WS AND WD ARE TAKEN, THE REST IS
! IGNORED! THE REST MAY SERVE FOR PLOTTING PURPOSES WITH USER PLOT SOFTWARE!
! HOWEVER, THERE IS ONE OCCASION WHERE THE COLUMN WITH P IS USED TO INTERPOLATE
! THE PRESSURE AT THE TOPOGRAPHY HEIGHT TO SERVE AS STARTING POINT FOR THE
! MODELS OWN HYDROSTATIC PRESSURE INITIALIZATION, SEE THE COMMENT ON lps_from_file BELOW!
!
! *NOTE 2*: THE COLUMN WITH T MAY ALSO CONTAIN THE POT. TEMPERATURE THETA INSTEAD.
!           THEN YOU HAVE TO SET THE FLAG rasofile_t_is_theta = .true. BELOW!
!
! This Format is inspired by http://weather.uwyo.edu/upperair/europe.html, but
!    slightly changed (column order, units)
!
!
! For itype_artifprofiles = 1/3, two additional logical switches
!                                "lps_from_file" and "rasofile_t_is_theta" take effect,
!                                see the comments on these switches below!
!
! For itype_artifprofiles = 2/3, one has to specify the type of the T / Qv - profiles
! in "itype_anaprof_tqv":
!    itype_anaprof_tqv = 1 :   Weisman-Klemp (1982) - type T and Qv-profiles,
!                              determined by the namelist variables with suffix "_wk"
!    itype_anaprof_tqv = 2 :   Arbitrary number of polytrope layers, specified by namelist
!                              *vectors* with suffix "_poly". These layers have
!                              a constant vertical T-gradient (for simplicity: ordinary
!                              temperature, not virtual temperature).
!    itype_anaprof_tqv = 3 :   Arbitrary number of layers with const.
!                              Brunt-Vaisala-freq. N, specified by namelist
!                              *vectors* with suffix "_nconst". These layers have
!                              a constant N with respect to moist unsaturated air.
!
! For itype_artifprofiles = 2/4, one has to specify the type of the U / V - profiles
! in "itype_anaprof_uv":
!
!    itype_anaprof_uv  = 1 :    Weisman-Klemp (1982) - type U - profile
!                               U(z) = u_infty * tanh((z-hmin_wk)/href_wk)
!    itype_anaprof_uv  = 2 :    Arbitrary numbers of constand gradient U(z) - layers
!    itype_anaprof_uv  = 3 :    U = const. = u_infty (West-East-Flow)
!    itype_anaprof_uv  = 4 :    V = const. = u_infty (South-North-Flow)
!
!
!=============================================================================
!
itype_artifprofiles = 2,
!
!=============================================================================
!
  itype_anaprof_tqv = 3,
  itype_anaprof_uv = 2,
!
!=============================================================================
!
!=============================================================================
!   For radiosonde file input (itype_artifprofiles = 1 / 3 / 4):
!=============================================================================
!
! Path to and name of raso file:
  rasofile = '$rasodatei$',
!
! If lps_from_file = .true., trust the pressure in the rasofile
! and use it to interpolate the surface pressure;
! otherwise do own analytic pressure calculation for this purpose:
  lps_from_file=.false.,
!
! If the rasofile contains pot. temperature theta instead of
```

```
! ordinary temperature t, set this flag to .true.:
  rasofile_t_is_theta = .false.,
!
!=============================================================================
!   Parameters for the analytic T- and Qv-profiles:
!=============================================================================
!
!-----------------------------------------------------------------------------
!   Parameters for the Weisman/Klemp-testcase (itype_anaprof_tqv = 1):
!-----------------------------------------------------------------------------
!
! Base height for the profiles, has to be lower than MINVAL(hsurf):
    hmin_wk = 0.0,
!
! Pressure at height hmin_wk in Pa:
    p_base_wk = 1e5,
!
! Tropopause height in m:
    h_tropo_wk = 12000.0,
!
! Potential temperature at height hmin_wk:
    theta_0_wk = 300.0,
!
! Potential temperature at tropopause height (343.0 in the original literature):
    theta_tropo_wk = 343.0,
!
! t_tropo_wk:
!  Estimate of temperature at tropopause height,
!  which is a parameter in the analytic Theta(z)-formula for an isothermal layer and is NO
!  really a specification of the *true* tropopause temperature!
!  The true tropopause temperature is determined by theta_tropo_wk and the resulting
!  pressure at that height. Because pressure is not known a priori, WK82 introduced
!  t_tropo_wk as an a priori estimate. However, this only leads to
!  a constant temperature in the whole tropopause layer if the "correct" t_tropo_wk
!  is specified. Otherwise a non-constant T-profile results, which is also the case
!  with the original literature value, which is 213.0 K and which is off by about
!  4 K compared to the "true" resulting value of about 217 K.
!  If t_tropo_wk < -900, then the exact tropopause temperature is determined and the
!  temperature in the tropopause layer is truly constant.
!  THE DEFAULT VALUE IS -999.99. IF YOU SPECIFY 213.0, YOU CAN REPRODUCE THE
!  ERROR IN THE ORIGINAL LITERATURE!
    t_tropo_wk = 213.0,
!
! Exponent of the potential temperature profile in the troposphere:
    expo_theta_wk = 1.25,
!
! Exponent of the relative humidity profile in the troposphere:
    expo_relhum_wk = 1.25,
!
! Constant relative humidity in the tropopause region:
    rh_min_wk = 0.25,
!
! Relative humidity at height hmin_wk:
    rh_max_wk = 1.0,
!
! Max. specific humidity, which is imposed after specifying the relhum profile:
    qv_max_wk = 0.014,
!
!-----------------------------------------------------------------------------
!   For piecewise polytrope atmosphere layers (itype_anaprof_tqv = 2):
!-----------------------------------------------------------------------------
!
! number of layers
    nlayers_poly = 3,
! Pressure at height h_poly(1) in Pa:
    p_base_poly = 1e5,
! For each layer, the base heights in m:
```

```
    h_poly = 0.0, 2000.0, 12000.,
! For each layer, temperature at base height in K:
    t_poly = 293.16, 273.66, 213.66,
! For each layer, temperature gradient in K/m:
!  (POSITIVE for DECREASING temperature with height):
    tgr_poly = 0.00975, 0.006, 0.0,
! For each layer, relative humidity at base height:
    rh_poly = 0.5, 0.9, 0.3,
! For each layer, gradient of relative humidity in 1/m:
!  (POSITIVE for DECREASING relhum with height):
    rhgr_poly = -2.0e-4, 6e-5, 0.0,
!
!-------------------------------------------------------------------
!   For piecewise const. Brunt-Vaisala-freq. (N) layers (itype_anaprof_tqv = 3):
!-------------------------------------------------------------------
!
! number of layers
    nlayers_nconst = 3,
! Pressure at height h_nconst(1) in Pa:
    p_base_nconst = 1e5,
! Pot. temp. at height h_nconst(1) in K:
    theta0_base_nconst = 300.0,
! For each layer, the base heights in m:
    h_nconst = 0.0, 1500.0, 10000.,
! For each layer, the value of N in 1/s:
    N_nconst = 0.001, 0.01, 0.01,
! For each layer, relative humidity at base height:
    rh_nconst = 0.5, 0.8, 0.3,
! For each layer, gradient of relative humidity in 1/m:
!  (POSITIVE for DECREASING relhum with height):
    rhgr_nconst = 0.0, 0.0, 0.0,
!
!
!=====================================================================
!   Wind profile (horizontal components):
!=====================================================================
!
!-------------------------------------------------------------------
!   for Weisman-Klemp tanh-profile (itype_anaprof_uv = 1):
!-------------------------------------------------------------------
! U(z) =  U_infty * tanh(z/href_wk)
    href_wk = 3000.0,
    u_infty = 5.0,
!
!-------------------------------------------------------------------
!   for piecewise linear wind layers (itype_anaprof_uv = 2):
!-------------------------------------------------------------------
!
! Number of layers:
    nlayers_linwind = 3,
! For each layer, base height in m:
    h_linwind = 0.0, 1500.0, 12000.0,
! For each layer, windspeed at base height in m/s:
    u_linwind = 10.0, 15.0, 20.0,
! For each layer, windspeed gradient in 1/s
!  (POSITIVE for INCREASING windspeed with height):
    ugr_linwind = 0.0, 0.0, 0.0,
!
!-------------------------------------------------------------------
!   const. U = u_infty (itype_anaprof_uv = 3):
!-------------------------------------------------------------------
!
!  ... set u_infty above!
!
!-------------------------------------------------------------------
!   const. V = u_infty (itype_anaprof_uv = 4):
!-------------------------------------------------------------------
```

```
!
!   ... set u_infty above!
!
!
!
!=======================================================================
!   Initialization of w: Switch to enable w in a way
!   that the streamlines follow the terrain following
!   vertical coordinate surfaces:
!=======================================================================
!
linitw_followeta = .true.,
!
!
!=======================================================================
!   Additional analytic uv-boundary-layer:
!=======================================================================
!
! Here one may impose an analytic boundary layer wind profile
! in case of non-vanishing windspeed at the ground and at the
! same time no-slip boundary conditions. The following formula
! (so-called "exponent wind profile") is applied, in the notation
! of the two relevant namelist parameters "zo_boundary" and
! "exponent_windprof_boundary":
!
!    if zo_boundary > 0.0:
!      if z < zo_boundary:
!        U(z) = U(zo_boundary) * ( (z-hsurf)/(zo_boundary) )**exponent_windprof_boundary
!
zo_boundary = 0.0,
  exponent_windprof_boundary = 0.25,
!
!
!=======================================================================
!=======================================================================
!
!
!=======================================================================
! 6) Possibility to specify constant turb. diffusion coefficients
!    in combination with itype_turb = 100
!    (set itype_turb in namelist PHYCTL accordingly)
!=======================================================================
!
! Diffusion coefficient for vertical diffusion of heat in m^2/s:
tkvhfix = 300.0,
! Diffusion coefficient for horizontal diffusion of heat in m^2/s:
tkhhfix = 300.0,
! Diffusion coefficient for vertical diffusion of momentum in m^2/s:
tkvmfix = 300.0,
! Diffusion coefficient for horizontal diffusion of momentum in m^2/s:
tkhmfix = 300.0,
!
!
!=======================================================================
!=======================================================================
!
!
!=======================================================================
! 7) Possibility to specify constant sensible and latent heat fluxes
!    at the surface, if the soil model is turned off (lsoil=.false.)
!    ( set lsoil in namelist PHYCTL accordingly, and do not use
!     itype_tran=3 )
!=======================================================================
!
! Master switch to apply pre-defined constant surface fluxes:
lsensiflux_fix = .false.,
! Magnitude of the constant sensible heat flux [W m**-2]:
```

```
  sensiflux_c = 300.0,
! Additionally specify constant latent heat flux [W m**-2]:
  llatentflux_fix = .false.,
! in case of llatentflux_fix = .true., the ratio of latent to sensible flux (Bowen ratio):
    latentflux_LzuS = 0.4,
! Relative level of additional white noise on the surface fluxes:
!  (noise will be added only if H0_rel_noise >= 1E-5!)
  H0_rel_noise = 0.05,
! In case of noise (H0_rel_noise >= 1E-5), the seed for the random number generator:
!  (set to -999 to use the system time instead of a fixed seed!)
  iseed_noise_H0 = 608,
!
!
!=============================================================================
!=============================================================================
!
!
!=============================================================================
! 8) Possibility to specify "no-surface-fluxes" lower boundary
!    condition, i.e., no heat-, moisture-, and/or momentum fluxes
!    at the ground. This includes the free-slip momentum BC.
!=============================================================================
!
! if lnosurffluxes_m = .true., there are no surface momentum fluxes:
!   (technically, the exchange coefficient tcm at the
!    surface is set to 0.0). THIS IS THE FREE-SLIP BOUNDARY CONDITION.
!
lnosurffluxes_m = .true.,
!
! if lnosurffluxes_h = .true., there are no surface heat/moisture fluxes:
!   (technically, the exchange coefficient tch at the
!    surface is set to 0.0)
!
lnosurffluxes_h = .true.,
!
!
!
!=============================================================================
!=============================================================================
!
!
!=============================================================================
! 9) Parameters for artificial convection triggers (e.g., warm bubbles)
!=============================================================================
!
!-----------------------------------------------------------------------------
! Vector of master switches for one or more (up to 50) triggers.
!   For each trigger you want, set .true. in the list:
!
ltempdist = .true., .false., .false.,.false.,
!
!-----------------------------------------------------------------------------
!
! For each trigger, set corresponding parameters in the following lists:
!
! Type of convection trigger (e.g., temperature disturbance):
!
! **** Choose one of: ***
!
!  ctype_tempdist = 'cos' :   3D cos^2 - shaped T-disturbance (default)
!
!                   'mcnider' :   3D T-disturbance after McNider and Kopp (JAM, 1990)
!
!                   'hotspot' :   2D Gaussian T-disturbance in the lowest level
!
!                  'squall3D' :   3D T-disturbance in the shape of an
!                                 elongated ridge in Y-dir.
```

```
!                              superimposed with small variations with
!                              random amplitude to initiate
!                              3D flow structures.
!                              (borrowed from George Bryan, NCAR)
!                              FOR THIS IDEALIZED CASE, SET:
!                              - lperi_y = .true.
!                              - je_tot in a way to ensure continuity and periodicity
!                                of the small T-disturbances
!                                in y-direction. These disturbances have
!                                distances of bub_radx/2 and
!                                radii of bub_radx/4 and start at point
!                                iy=nboundlines+1+bub_radx/(2*dy).
!                               A proper choice of je_tot is such that
!                                 je_tot - 2*nboundlines = k * bub_radx/(2*dy) + 1,
!                                     k = natural number
!                                For example: if nboundlines=3, bub_radx=10000.0
!                                  and dx=1000.0, je_tot=47 would be appropriate.
!                              To reproduce the original from George Bryan:
!                              - bub_centi = middle of domain
!                              - bub_centz = 1400.0
!                              - bub_radx = 10000.0
!                              - bub_radz = 1400.0
!
!                  'SK94' :    Disturbance for the Skamarok-Klemp (1994)
!                              testcase of a traveling gravity wave.
!                              CAUTION: The moisture is set to 0.0 in this test case.
!                              To reproduce the literature, choose
!                              - bub_dT = 0.01
!                              - bub_radx = 5000.0
!                              - Atmosphere with const. N=0.01, p0=1e5 and T0=250.0
!                              - Rel. hum. = 0.0
!
!
!             'cos-soil' :    2D horizontal cos^2 - shaped T-disturbance in the soil,
!                             homogeneous throughout the soil depth
!                             (for cases with activated soil model)
!
!         'hotspot-soil' :    2D gaussian shaped T-disturbance in the uppermost
!                             soil level (for cases with activated soil model)'
!
!         'cos-soil-hrd' :    2D cos^2 - shaped heating rate disturbance in
!                             the soil (for cases with activated soil model)
!
!     'hotspot-soil-hrd' :    2D gaussian shaped heating rate disturbance
!                             in the uppermost soil level
!                             (for cases with activated soil model)'
!
!
!               'cos-hrd' :   3D cos^2 - shaped heating rate disturbance in the atmosphere
!
!           'mcnider-hrd' :   3D heating rate disturbance, using the scaled shape
!                             function of McNider and Kopp (JAM, 1990)
!
!           'hotspot-hrd' :   2D Gaussian heating rate disturbance in the lowest level
!
! 'AS2005_hucmtexas-hrd' :    3D heating rate disturbance
!
!
!           'hotspot-sfc' :   2D Gaussian T-disturbance at the surface
!                             (only for cases with t_g=const. and deactivated soil model)
!
!  IF SEVERAL TRIGGERS ARE SPECIFIED FOR ONE LOCATION,
!  THE SUM OF THE DISTURBANCES IS TAKEN.
!
!-------------------------------------------------------------------------------
!
  ctype_tempdist = 'cos', 'cos-hrd', 'mcnider', 'hotspot',
```

```
!
!-----------------------------------------------------------------------
!
! Depending on ctype_tempdist, other parameters take effect, which
! have to be set below. Each of these is also a vector whose elements correspond
! to the elements in the flaglist ltempdist:
!
!      'cos' :    bub_centi, bub_centj, bub_centz
!                 bub_radx, bub_rady, bub_radz, bub_rotangle, bub_dT,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise, lbub_rhconst
!
!  'mcnider' :    bub_centi, bub_centj
!                 bub_zi_mcnider, bub_zmax_mcnider, bub_h0_mcnider,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise, lbub_rhconst
!
!  'hotspot' :    bub_centi, bub_centj
!                 bub_radx, bub_rady, bub_rotangle, bub_dT,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise, lbub_rhconst
!
!      'SK94' :   bub_centi, bub_dT, bub_radx
!
!  'squall3D' :   bub_centi, bub_centz, bub_dT,
!                 bub_radx, bub_radz
!
!  'cos-soil' :   bub_centi, bub_centj
!                 bub_radx, bub_rady, bub_rotangle, bub_dT,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise
!
! 'hotspot-soil': bub_centi, bub_centj
!                 bub_radx, bub_rady, bub_rotangle, bub_dT,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise, lbub_rhconst
!
!  'cos-soil-hrd' : htempdist, bub_centi, bub_centj, bub_timespan,
!                 bub_radx, bub_rady, bub_rotangle, bub_heatingrate,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise
!
! 'hotspot-soil-hrd': htempdist, bub_centi, bub_centj, bub_timespan,
!                 bub_radx, bub_rady, bub_rotangle, bub_heatingrate,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise, lbub_rhconst
!
!  'cos-hrd' :    bub_centi, bub_centj, bub_centz, bub_timespan,
!                 bub_radx, bub_rady, bub_radz, bub_rotangle, bub_heatingrate,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise, lbub_rhconst
!
! 'mcnider-hrd' : bub_centi, bub_centj, bub_timespan,
!                 bub_zi_mcnider, bub_zmax_mcnider, bub_h0_mcnider, bub_heatingrate,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise, lbub_rhconst
!
! 'hotspot-hrd' : bub_centi, bub_centj, bub_timespan,
!                 bub_radx, bub_rady, bub_rotangle, bub_heatingrate,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise, lbub_rhconst
!
! 'AS2005_hucmtexas-hrd': htempdist, bub_centi, bub_centj, bub_centk, bub_timespan,
!                 bub_heatingrate
!
! 'hotspot-sfc' : htempdist, bub_centi, bub_centj, bub_timespan
!                 bub_radx, bub_rady, bub_rotangle, bub_dT,
!                 ladd_bubblenoise_t, bub_dT_bubblenoise
!
!
! Time of first occurence of warm bubble in h:
   htempdist = 0.0, 1.0,
! Center of the disturbance in x-dir. in units of gridpoints (real numbers allowed):
   bub_centi = 51.0, 45.5,
! Center of the disturbance in y-dir. in units of gridpoints (real numbers allowed):
   bub_centj = 80.5, 80.5,
! Height of center of disturbance in m (or in no. of levels from the bottom
```

```
! for 'AS2005_hucmtexas-hrd'):
    bub_centz = 1400.0, 1400.0,
    bub_centk = 5, 5,
! Time span of disturbance in units of time steps, for const. heating rate bubbles:
    bub_timespan = 1, 100,
! Radius/half width in x-dir of the disturbance in m:
    bub_radx = 10000.0, 10000.0,
! Radius/half width in y-dir of the disturbance in m:
    bub_rady = 10000.0, 10000.0,
! Vertical radius/half width of the disturbance in m:
    bub_radz = 1400.0, 1400.0,
! Horizontal rotation angle of the main axes (counterclockwise) of
! the disturbances in degrees:
    bub_rotangle = 0.0, 0.0,
! Amplitude parameter of the disturbance in K:
! (must be >= 0.0, except for 'cos'-disturbances, where also a "cold bubble" may be specif
    bub_dT = 3.0, 3.0,
!
! Heating rate parameter of the disturbance in K/s:
    bub_heatingrate = 0.03, 0.03
!
! lbub_rhconst:
!  Choose between constant relative humidity during temp.
!   dist. heating (.true.=default) or no moisture
!   modification within the temperature disturbance region (.false.):
    lbub_rhconst = .false., .true.,
!
! Switch to add random noise to the disturbance:
    ladd_bubblenoise_t = .true., .true.,
! In case of ladd_bubblenoise_t=.true., relative noise level, such that
! dT_bubble = dT_bubble * (1 + bub_dT_bubblenoise * random_noise[-1,1] :
     bub_dT_bubblenoise = 0.1, 0.1,
!
! Boundary layer height parameter of the T-disturbance parameterization of McNider & Kopp
    bub_zi_mcnider = 4000.0, 4000.0,
! Top height of the T-disturbance after McNider & Kopp in m:
    bub_zmax_mcnider = 6000.0, 6000.0,
! Surface heat flux parameter of the T-disturbance parameterization of McNider & Kopp in W
    bub_h0_mcnider = 600.0, 600.0,
!
!
!
!=============================================================================
!=============================================================================
!
!
!=============================================================================
! 10) Possibility to add white noise at some time in the boundary layer,
!    i.e., in the lowest 100 hPa of the atmosphere.
!=============================================================================
!
! In case of ladd_noise_t=.true., absolute noise level, such that
! T' = dT_noise * random_noise[-1,1]
!
!-----------------------------------------------------------------------------
! Master switch:
ladd_noise_t = .false.,
!-----------------------------------------------------------------------------
!
! Time of noise specification in h:
  hadd_noise = 0.0,
! Amplitude of T-noise in K:
  dT_noise = 0.02,
! Integer seed for the random number generator for the T-noise: (default = 606)
!  (set to -999 to use the system time instead of a fixed seed!)
  iseed_noise_t = 606,
```

```
! Amplitude of W-noise in m/s:
  dW_noise = 0.02,
! Integer seed for the random number generator for the W-noise: (default = 607)
!  (set to -999 to use the system time instead of a fixed seed!)
  iseed_noise_w = 607,
!
!
!
!=============================================================================
!=============================================================================
!
!
!=============================================================================
! 11) Possibility to specify the time of condensation- and microphysics
!     activation in h different from 0.0
!     (in case of lcond=.true. and/or lgsp=.true.).
!     This is meant to enable the possibility of
!     having a "dry" flow spinup period before
!     switching on condensation and cloud processes:
!=============================================================================
!
! Hour of condensation- and microphysics activation in h since model start:
hcond_on = 0.0,
!
/
```

6 Documentation of implemented features

In der Reihenfolge der namelist die verschiedenen Moeglichkeiten fuer Vertikalgitter, Referenzatmosphaere, externe Parameter, Randbedingungen, Anfangsbedingungen, Druckinitialisierung, Konvektionstrigger beschreiben (mit Formeln).

6.1 Vertical grid levels

6.2 Reference atmosphere

6.3 Orography and other surface parameters

The meaning of the different namelist parameters for the hill(s) is illustrated by way of two examples.

The first (depicted in Figure 1 as a horizontal height contour line plot) consists of several mountains and one valley (depth 100 m) in the model domain. To keep the orography height above 0 m, we set a constant orography base height of 100 m instead of the default 0 m. The following namelist snippet shall define this example, where we have chosen the model grid length to 1 km (X and Y-direction):

```
!=============================================================================
! 3) Initialization of the artificial orography and soil parameters:
!=============================================================================
!
!=============================
linit_realoro = .false.,
!=============================
!
href_oro=100.0,
!
!
!=============================
itype_topo = 1,
!=============================
```

```
!
!   lhill is the "master" switch. Set as much elements to true as much
!     hills/valleys you want, e.g., lhill=.true.,.true.,.true., generates
!     3 hills/valleys.
  lhill = .true.,.true.,.true.,.true.,.true.,.true.,.true.,
!
!       lhill_2d determines if the resp. hill/valley is 2-dim. (oriented along the
!       y-axis) or 3-dim. in the case of lhill_2d=.false.:
      lhill_2d = .true.,.false.,.true.,.true.,.false.,.true.,.false.,
!
!       shape of mountain:
!         - 'gauss' (symmetric)
!         - 'bellshaped' (symmetric)
!         - 'gauss-asym' (asymmetric)
!         - 'bellshaped-asym' (asymmetric)
      hill_type = 'gauss','gauss','gauss','gauss','gauss','gauss','gauss',
!
!       Coordinates of mountain center point(s) in GPs, can be real numbers, not just inte
      hill_i =  46.0, 81.0, 106.0, 181.0, 181.0, 181.0, 181.0,
      hill_j = 141.0, 51.0, 141.0,  55.0,  55.0, 145.0, 145.0,
!
!     hillheight (in m):
!       top height of the mountain(s) (if negative, depth of the valley(s)):
      hillheight = 500.0, 500.0, -100.0, 250.0, 250.0, 250.0, 350.0,
!
!
!     hill_rotangle (in degrees):
!       Clockwise rotation angle(s) around the mountain center(s)
!       with respect to its horizontal orientation:
      hill_rotangle = 30.0, -30.0, -20.0, 40.0, -60.0, 40.0, -60.0,
!
!     zhillcutfact (dimensionless):
!       Factor to limit the mountain(s) width to a finite value by
!       removing the tails of the mountain(s). This works as follows:
!       If, e.g., zhillcutfact = 0.02, the mountain height is increased by
!       0.2*hillheight, and subsequently 0.02*hillheight are chopped away from
!       the bottom, so that the mountain looses its tails and has the same
!       height as before. (for valleys, it is similar but chop from the top).
!       NOTE: this slightly distorts the mountain shape
!       and leads to a difference between the "true" halfwidths and the
!       specified half width parameters below.
      zhillcutfact = 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02,
!
!     hill_combineaction:
!       This integer flag determines the way in which the mountain is combined with the
!       previously present mountains (mountains are processed in the order of apperance
!       in the mountian(s) list):
!          hill_combineaction = 1 :    add to the previous orography (subtract in the case
!          hill_combineaction = 2 :    take the maximum (minimum in case of a valley)
!                                      of this mountain and the previous mountain(s)
      hill_combineaction = 1, 1, 1, 1, 1, 1, 2,
!
!     hill_width_x (in m):
!       Half width radius to mountain center in X-direction:
!       (both for 2D- and 3D-mountain(s)):
      hill_width_x = 10000.0,15000.0,10000.0,7500.0,10000.0,7500.0,10000.0,
!
!     hill_width_y (in m):
!       For 3D-mountain(s) (lhill_2d=.false.)
!       - Half width radius to mountain center in Y-direction:
!       For 2D-mountain(s) (lhill_2d=.true.)
!       - 1/2 overall width, centered around the mountain center in Y-direction:
      hill_width_y = 35000.0,25000.0,25000.0,30000.0,30000.0,30000.0,30000.0,
!
!
!     hillsideradius_y (in m):
!       Effective for 2D-mountain(s) (lhill_2d=.true.) only:
```

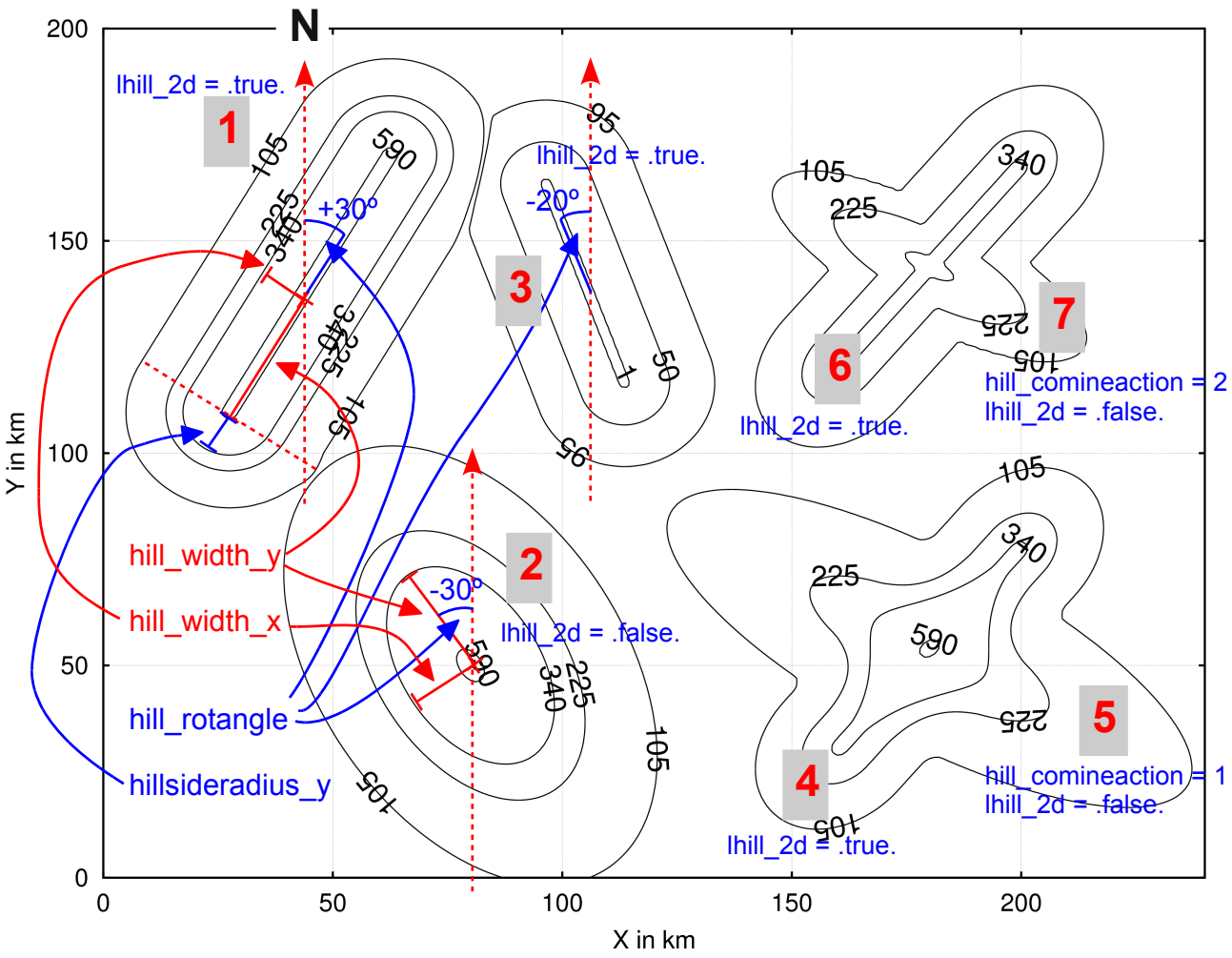

Figure 1: Resulting contour lines of the orography for the exemplary namelist snippet from Subsection 6.3, assuming `itype_topo = 1`. All hills are `gauss`-shaped, and isolines have been chosen in a way that for each hill / valley, one line is very close to the mountain foot, one line represents the half height and one line is close to the top height. This figure is designed to show the "horizontal" parameters, whereas Figure 2 figure clarifies the "vertical" parameters.

```
!       To round the "sharp" edges of a finite 2D-mountain
!       at its northern and southern sides,
!       the half of a 3D-mountain with X-halfwidth "hillwidth_x"
!       is added at each side, which has the following
!       half width radius in Y-dir.:
      hillsideradius_y = 10000.0, 10000.0, 10000.0, 10000.0, 10000.0, 10000.0, 10000.0,
!
!     Asymetric hills: (if chill_type = 'gauss-asym' or 'bellshaped-asym')
!       Asymmetry factors for the hill(s)/valley(s) in X- and Y-direction, defined
!       as the ratio between the west (south) and east (north) side half widths.
!       E.g., a factor of hillasym_x=1.5 means that the west side is 1.5 times as wide
!       as the east side, and that the west side has a half width radius of
!       hillwidth_x * 0.5.
      hillasym_x = 1.0, 1.0, 1.0,
      hillasym_y = 1.0, 1.0, 1.0,
```

The height contour lines of the resulting orography are depicted in Figure 1, along with explanations of relevant namelist parameters. The numbers (red on grey) attached to the hills correspond to the hill's rank in the above namelist and defines the order of their processing. The units of grid points for the hill center positions directly correspond to km in the figure. Note in particular the different meaning of `hillwidth_y` for 2D hills (`lhill_2d = .TRUE.`) and 3D-hills (`lhill_2d = .FALSE.`), the meaning of `hill_rotangle` and the effect of choosing `hillcombineaction = 1` (add/subtract to the previous orography depending on the sign of the `hill_height`) and `hillcombineaction = 2` (take max/min of this hill and the previous orography, depending on the sign of the `hill_height`).

The second example (depicted in Figure 2 as a vertical cross section) also consists of several mountains and one valley (depth 100 m) in the model domain, but in a purely 2D-configuration (X-Z). Again, we set a constant orography base height of 100 m instead of the default 0 m. As above, the following namelist snippet shall define this example, where we have chosen the model grid length to 1 km (X) and this time a 2D setup (`l2dim = .true.`):

```
!=============================================================================
! 3) Initialization of the artificial orography and soil parameters:
!=============================================================================
!
href_oro=100.0,
!
!
itype_topo = 1,
!
!   lhill is the "master" switch. Set as much elements to true as much
!     hills/valleys you want, e.g., lhill=.true.,.true.,.true., generates
!     3 hills/valleys.
  lhill = .true.,.true.,.true.,.true.,.true.,.true.,.true.,.true.,
!
!       lhill_2d determines if the resp. hill/valley is 2-dim. (oriented along the
!       y-axis) or 3-dim. in the case of lhill_2d=.false.:
      lhill_2d = .true.,.true.,.true.,.true.,.true.,.true.,.true.,.true.,
!
!       shape of mountain:
!         - 'gauss' (symmetric)
!         - 'bellshaped' (symmetric)
!         - 'gauss-asym' (asymmetric)
!         - 'bellshaped-asym' (asymmetric)
      hill_type = 'gauss','gauss','gauss-asym','gauss-asym','gauss','gauss','gauss','gauss',
!
!       Coordinates of mountain center point(s) in GPs, can be real numbers, not just inte
      hill_i =  51.0, 151.0, 271.0, 381.0, 576.0, 636.0, 811.0, 901.0,
      hill_j =   4.0,   4.0,   4.0,   4.0,   4.0,   4.0,   4.0,   4.0,
!
!     hillheight (in m):
!       top height of the mountain(s) (if negative, depth of the valley(s)):
      hillheight = 200.0, -100.0, 200.0, 200.0, 300.0, 300.0, 200.0, 200.0,
!
!
!     hill_rotangle (in degrees):
!       Clockwise rotation angle(s) around the mountain center(s)
!       with respect to its horizontal orientation:
      hill_rotangle = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
!
!     zhillcutfact (dimensionless, range 0...infinity):
!       Factor to limit the mountain(s) width to a finite value by
!       removing the tails of the mountain(s). This works as follows:
!       If, e.g., zhillcutfact = 0.02, the mountain height is increased by
!       0.02*hillheight, and subsequently 0.02*hillheight are chopped away from
!       the bottom, so that the mountain looses its tails and has the same
!       height as before. (for valleys, it is similar but chop from the top).
!       NOTE: this slightly distorts the mountain shape
!       and leads to a difference between the "true" halfwidths and the
!       specified half width parameters below.
      zhillcutfact = 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02,
```

```
!
!     hill_combineaction:
!       This integer flag determines the way in which the mountain is combined with the
!       previously present mountains (mountains are processed in the order of apperance
!       in the mountian(s) list):
!          hill_combineaction = 1 :    add to the previous orography (subtract in the case of
!          hill_combineaction = 2 :    take the maximum (minimum in case of a valley)
!                                      of this mountain and the previous mountain(s)
      hill_combineaction = 1, 1, 1, 1, 2, 2, 1, 1,
!
!     hill_width_x (in m):
!       Half width radius to mountain center in X-direction:
!       (both for 2D- and 3D-mountain(s)):
      hill_width_x = 15000.0,15000.0,25000.0,25000.0,40000.0,40000.0,40000.0,40000.0,
!
!     hill_width_y (in m):
!       For 3D-mountain(s) (lhill_2d=.false.)
!       - Half width radius to mountain center in Y-direction:
!       For 2D-mountain(s) (lhill_2d=.true.)
!       - 1/2 overall width, centered around the mountain center in Y-direction:
      hill_width_y = 30000.0,30000.0,30000.0,30000.0,30000.0,30000.0,30000.0,30000.0,
!
!
!     hillsideradius_y (in m):
!       Effective for 2D-mountain(s) (lhill_2d=.true.) only:
!       To round the "sharp" edges of a finite 2D-mountain
!       at its northern and southern sides,
!       the half of a 3D-mountain with X-halfwidth "hillwidth_x"
!       is added at each side, which has the following
!       half width radius in Y-dir.:
      hillsideradius_y = 10000.0, 10000.0, 10000.0, 10000.0, 10000.0, 10000.0, 10000.0, 10000.0
!
!     Asymetric hills: (if chill_type = 'gauss-asym' or 'bellshaped-asym')
!       Asymmetry factors for the hill(s)/valley(s) in X- and Y-direction, defined
!       as the ratio between the west (south) and east (north) side half widths.
!       E.g., a factor of hillasym_x=1.5 means that the west side is 1.5 times as wide
!       as the east side, and that the west side has a half width radius of
!       hillwidth_x.
      hillasym_x = 1.0, 1.0, 2.0, 0.5, 1.0, 1.0, 1.0, 1.0,
      hillasym_y = 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
```

A vertical cross section through the resulting orography is depicted in Figure 2, along with explanations of relevant namelist parameters. Again, the numbers (red on grey) attached to the hills correspond to the hill's rank in the above namelist and define the order of their processing. The units of grid points for the hill center positions directly correspond to km in the figure. Note the effects of choosing `hillcombineaction = 1` (add/subtract to the previous orography depending on the sign of the `hill_height`) and `hillcombineaction = 2` (take max/min of this hill and the previous orography, depending on the sign of the `hill_height`). Now the mountains no. 3 and 4 are asymetric by choosing `hill_type = 'gauss-asym'` and setting `hill_asym_x` to values different from 1.0.

6.4 Constant atmospheric parameters

6.5 Soil parameters

6.6 Initial fields of temperature, moisture and windspeed

6.7 Initial field of pressure

Based on the initial vertical profiles of temperature T and specific moisture q_v as function of height z (combined in the profile of virtual temperature T_v), the hydrostatic pressure in each model column is computed by integrating the hydrostatic approximation upwards, starting from the surface pressure p_s. Because q_v and therefore T_v itself depends on pressure, the problem becomes implicit and a fixpoint

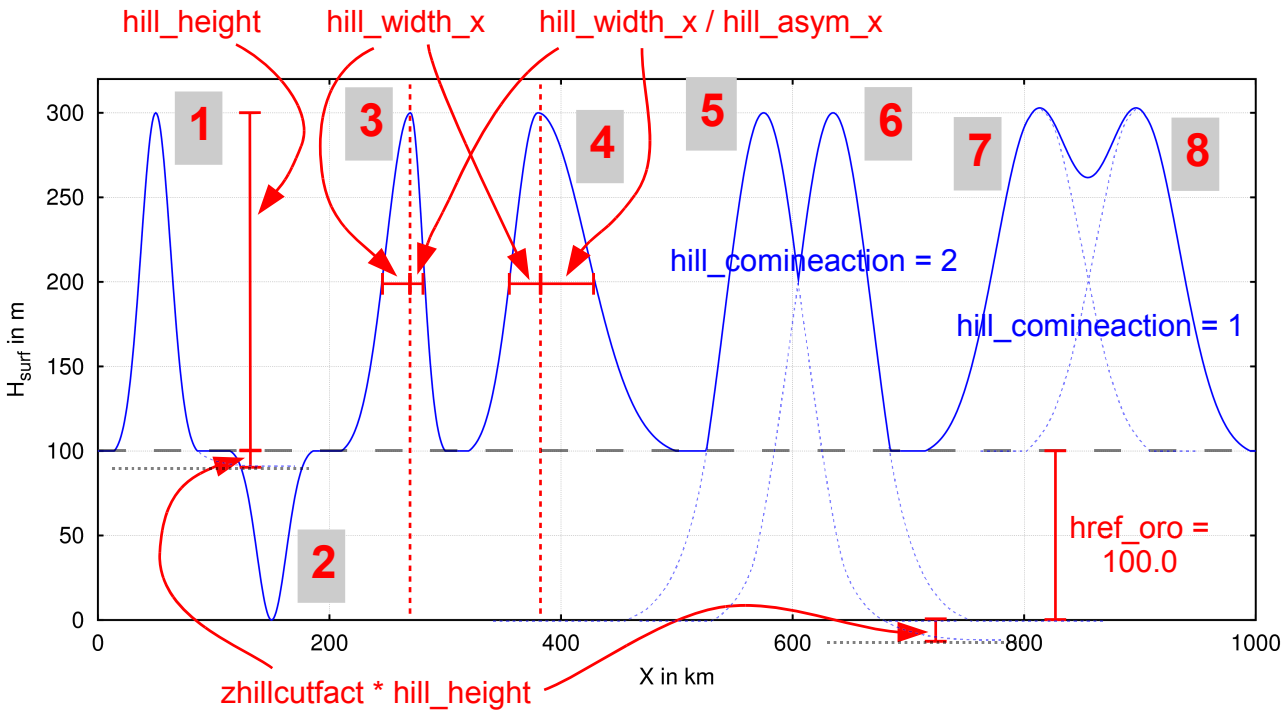

Figure 2: Resulting contour lines of the orography for the second exemplary namelist snippet from Subsection 6.3. All hills are `gauss`-shaped. In contrast to Figure 1, this example is designed to depict the "vertical" parameters.

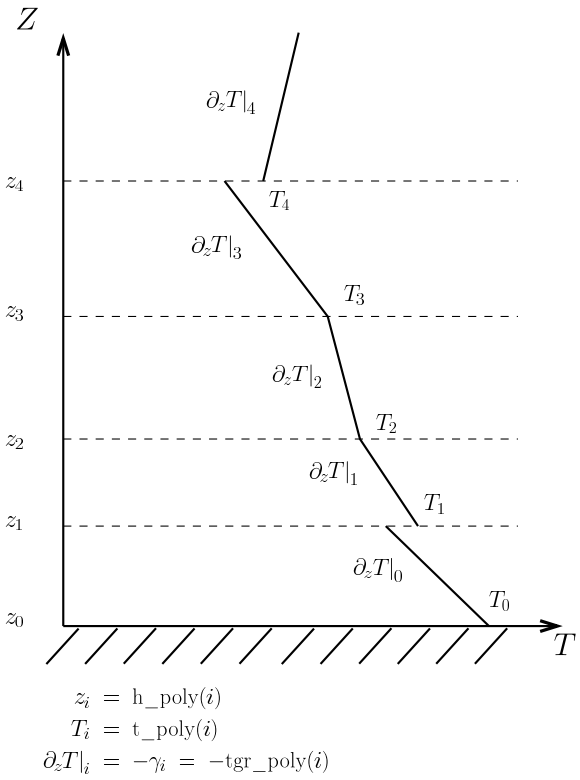

Figure 3:

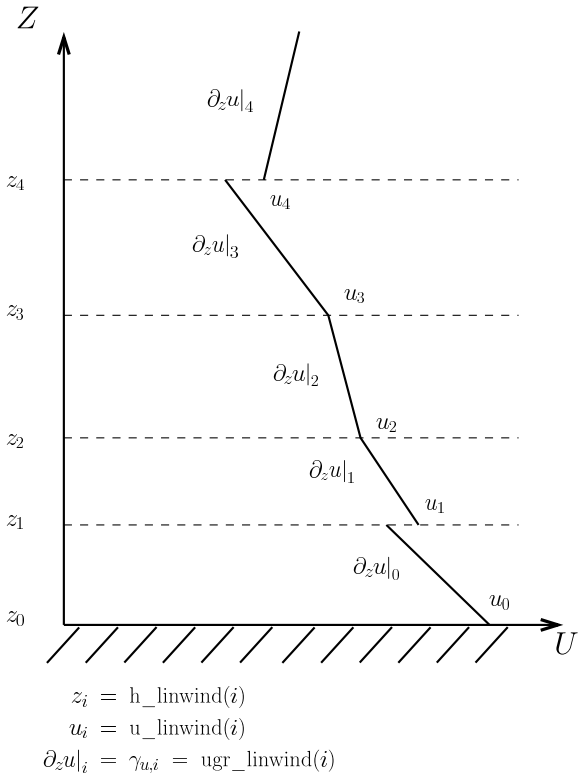

Figure 4:

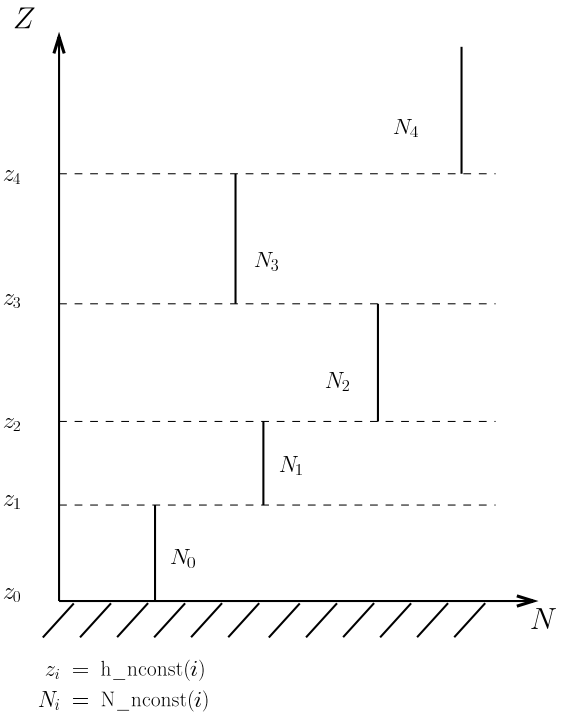

Figure 5:

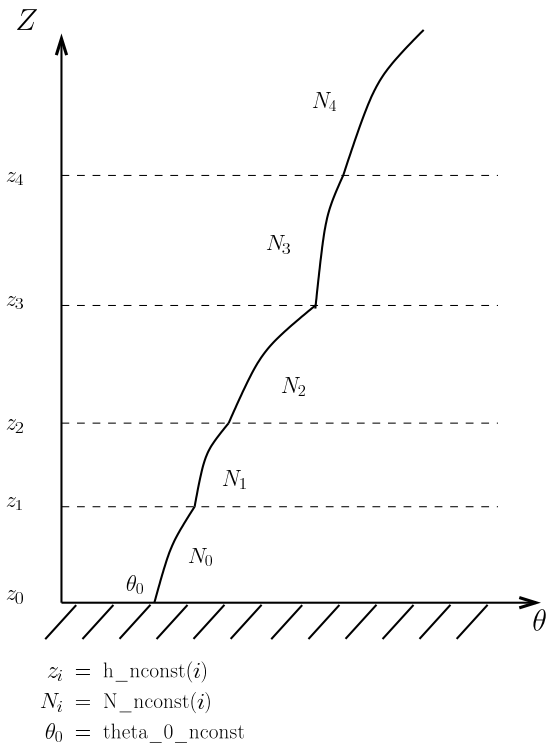

Figure 6: Definition of layer indices and corresponding namelist parameters for `itype_anaprof_tqv=3` (layers with constant Brunt-Vaisala-Frequency)

iteration is used to solve it. The iteration runs until convergence (change from one iteration to the next less than a certain threshold) or until a maximum number of iterations (currently 40) is reached. If no convergence could be achieved, the program stops with an extensive diagnostic message.

It is possible to give the potential temperature instead of normal temperature in a radiosonde file. If you do so, please set the namelist parameter `rasofile_t_is_theta = .true.`.

The integration of the hydrostatic equation is discretized consistently to the used dynamical core and fast waves solver. Different subroutines are used for the leapfrog and Runge-Kutta core and for the old and new fast-waves solver in the Runke-Kutta core.

There is also a subroutine which computes an analytic solution based on treating the vertical profile of T_v as a linear spline.

Note that there is currently no geostrophic pressure initialization. Therefor, the Coriolis force would b a problem and is normally disregarded by setting `lcori = .false.`.

6.8 Boundary values for prognostic fields

If you run with specified boundary conditions and lateral relaxation layer, the boundary data are taken from the initial conditions and are held constant in time.

Time-dependent conditions would be possible but have to be specified by the user in the source code `src_artifdata.f90`, subroutine `gen_bound_data()` under the CASE statement for `action == 2`. In doing so, the user can use the time measure `ntstep*dt` (time since model start in seconds).

6.9 Artificial convection triggers

Formeln mit gemeinsamen Parametern, die dann in einer Tabelle auf namelist-parameter gemappt werden.

6.10 Running the soil model

You can switch on the soil model without restrictions. If you do so, you have to specify how the soil parameters are intitialized, see above Subsection 6.5.

6.11 Running the convection scheme

You can switch on the convection schemes without restrictions.

6.12 Running the radiation scheme

You can switch on the radiation scheme during idealized experiments by `lrad=.true.`, and you can use all features of the real-case mode, i.e., calling the scheme only every nth time step or every nth time interval and/or calculating fluxes at a coarser resolution grid (however, only a doubling of the grid lengths is working correctly in the COSMO-model at the moment).

There are however some things to know:

- If running with the radiation scheme, the shortwave daily cycle of the shortwave radiation balance is determined by the model start day and time and the forecast hour (the usual namelist parameters `ydate_ini` and `hstop`.
- If running the radiation scheme with periodic boundary conditions, the local sun zenit angle and the areosol climatology have to be also periodic to avoid spurious boundary effects by non-periodic short- and longwave fluxes. This is achieved in the radiation code by specifying a constant geographic position in the direction of periodicity(ies) when computing the sun zenit angle and the aerosol concentration from the climatology. The constant position is taken as the reference position of the model domain, i.e., the point where the rotated 0-meridian crosses the rotated equator.
  It is therefore best to center the model domain around this rotated (0,0)-point by setting `startlon_tot`, `startlat_tot` accordingly.
- If the soil model is also running, the soil reacts in the "normal" way to the solar input. If the soil model is not running, pre-specified surface temperatures are held constant at their initial value during the model run (except when the user specifies disturbances in the soil temperature, see above) and the radiation scheme only works on the atmophere.

6.13 Miscellaneous features

e.g., hcond_on, itype_turb=100, itype_interp, no-slip and free-slip lower boundary condition

7 Code structure of `src_artifdata.f90`

7.1 Code tree

7.2 Subroutine definitions

8 Implementing own idealized features into `src_artifdata.f90`

9 Known issues / problems

- Convergence problems in iterative pressure initialization may happen if:

- the relative humidity (input to the pressure initialization) is specified too large, so that the resulting vapor pressure would be larger than the total pressure. This is an unphysical situation and cannot be realized by the atmosphere.

  In that situation, the pressure iteration limits the relative humidity to the maximum allowed value $p/E_{sat,w}(T)$, and you get a corresponding warning message. The respective layers are then pure layers of water vapor ($Q_V = 1$). This may happen, e.g., at larger heights if you specify an atmosphere with strongly increasing temperature with height connected with unrealistically high values of relative humidity for that height.

  However, if this is connected with low pressure at very high altitudes (> 30 km), there have been convergence problems observed. Reduce the relat. humidity at high altitudes to, e.g., a value of 0 above 25 km height, if you encounter errors.

- if the initial profiles of temperature T are specified via potential temperature θ instead of absolute temperature, e.g., for Weisman-Klemp-type profiles or for layers with $N = \text{const}$. Then there might occur problems if the atmosphere is very high (`zz_top` > 30 km or so) and much warmer at larger heights than the reference atmosphere.

  In the iteration, T is computed from θ and the iterated pressure, and on the first iteration, the pressure is taken from the reference atmosphere. Now, if the reference atmosphere is much colder than the initialized atmosphere will be, the pressure is much lower at high altitudes, which leads to very low T for the first iteration, and if $T = 35.86\,\text{K}$, the saturation vapor pressure becomes 0.0 (leading to a crash) and increases towards lower temperatures (leading to divergent iteration).

  Further, a high temperature at high altitudes itself can be another reason for a diverging iteration. If it is connected with the above limitation of a too large specified relative humidity, the computation of θ is the cause for "jumpy" and non-converging behaviour of the iterated pressure and temperature, because it assumes (consistent with the basic COSMO equations) that the gas constant R_m takes correctly into account the different moisture components, whereas c_p is simply taken as that of dry air. This inconsistency is normally not critical, but may become a problem in this special situation. Also, this assumption is questionable if Q_v attains values larger than, say, 0.1.

  Reduce the atmosphere height and/ or decrease the temperature/moisture at high altitudes and/ or make the reference atmosphere warmer at high altitudes.

- Results of the iterative hydrostatic pressure initialization depend on:

  - the choice of the reference atmosphere. The dependence is only very minor, but it is there. This is presumably due to the fact that the reference atmosphere is analytically hydrostatic, whereas the "numerical" hydrostatic balance is slightly different and depends on the numerics. The computation of this numerical balance involves in turn the reference atmosphere temperature and pressure, which might be slightly inconsistent.

- The initial profiles are defined analytically on full levels, which are assumed to be in the geometric center of the model half levels throughout the source code of `src_artifdata.f90`.

  This is true for the Runge-Kutta dynamics, but is not true for the Leapfrog dynamics. For the latter, the full levels are situated in the mass-weighted center of the half levels (?). This leads to a slight offset of the initial profiles when using Leapfrog dynamics, which is (among other reasons) a source of differences between simulation results obtained with Runge-Kutta- or Leapfrog dynamics.

- Unreproducible model crashes with the ifort-compiler (in our case it was version 10.1.022) in case of `lsoil = .false.` combined with all surface temperature disturbances (types "hotspot-sfc", "cos-soil", "cos-soil-hrd", "hotspot-soil", "hotspot-soil-hrd"). "Floating invalid" operation occurs in the call to `cloud_diag` from `turbtran.incf`. Observed even without any optimizations and for serial runs when floating point exceptions are trapped. Reason unknown, most probable a compiler bug. Does not happen if using the GNU gfortran compiler.

- Radiation scheme: `nradcoarse > 2` does not work! If chosen >2, the fluxes at the surface are totally nonsense!

10 Table of new namelist parameters for idealized cases

Vector dimensions for some namelist parameters:

lcbuf = 100
nhill_max = 50
nvcoordvec_max = 999
ntempdist_max = 50
nlayers_poly_max = 10
nlayers_nconst_max = 10
nlayers_linwind_max = 10

Name	Kind (Dim.)	Description / Remarks	Default
ldebug_artif	L (1)	Debug mode for additional runtime messages. Set ldebug_artif = .true. to enable and idbg_level to some value > 0.	ldebug_artif_d = .FALSE.
idbg_artif_level	I (1)	Debug level for artificial elements: • idbg_artif_level > 0 : print the subroutine name at the beginning of each (major) subroutine. • idbg_artif_level > 3 : additionally, write ASCII-files (or BIN-files on the NEC) containing the T- and QV- increment resp. heating rate for each artif. temperature-, moisture or heating rate disturbance triggered in the simulation. **On the NEC, you need the conversion program "bin2ascii_convrates3d.f90" by Ulrich Blahak to generate ASCII-files from the BIN-files.** • idbg_artif_level > 4 : print additional checking output for the iterative hydrostatic pressure initialization.	idbg_artif_level_d = 0
irefatm	I (1)	Parameter to choose the reference atmosphere: • irefatm = 1 : old reference atmosphere, based on a constant logarithmic temperature gradient. • irefatm = 2 : (RECOMMENDED) new reference atmosphere, based on an exponential temperature profile. • irefatm = 3 : new reference atmosphere with constant dry Brunt-Vaisala-frequency bvref.	irefatm_d = 2
p0sl	R (1)	Reference surface ($z = 0.0$ m) pressure [Pa]] (irefatm=1,2,3)	p0sl_d = 1.0E5
t0sl	R (1)	Reference surface temperature [K] (irefatm=1,2,3)	t0sl_d = t0_melt + 15.0
dt0lp	R (1)	Logarithmic temperature gradient [K]] (irefatm=1)	dt0lp_d = 42.0
delta_t	R (1)	Temperature diff. surface - stratosphere [K]] (irefatm=2)	delta_t_d = 75.0
h_scal	R (1)	e-folding height of exponential adjustment to the stratosphere temp. [m]] (irefatm=2)	h_scal_d = 10000.0
bvref	R (1)	Constant Brund-Vaisala-frequency [1/s]] (irefatm=3)	bvref_d = 0.01
ivctype	I (1)	General Type of vertical coordinate specification: • ivctype = 1 : pressure based coordinate (σ-type, i.e., p/p_s) • ivctype = 2 : height based coordinate (RECOMMENDED) • ivctype = 3 : height based SLEVE coordinate	ivctype_d = 2

Name	Kind (Dim.)	Description / Remarks	Default
zspacing_type	C(len=lcbuf) (1)	Type of vertical coordinate spacing. Depending on ivctype, one of 'predefined', 'linear', 'galchen' or 'vcoordvec'	zspacing_type_d = 'galchen'
exp_galchen	R (1)	Exponent in the Gal-Chen formula	exp_galchen_d = 2.6
vcflat	R (1)	Blending height above which coordinate surfaces become flat [m]. Has to be smaller than rdheight!	vcflat_d = 11000.0
zz_top	R (1)	Top of model domain [m]	zz_top_d = 22000.0
vcoordvec	R (nvcoordvec_max)	If zspacing_type = 'vcoordvec', then this is the vector of height values for the coordinate surfaces, monotonically decreasing from zz_top to 0.0. The number of values larger than -999.99 has to be equal to ke+1 (-999.99 beeing the missing value).	vcoordvec_d(:) = -999.99
nfltvc	I (1)	Parameter for the SLEVE coordinate (ivctype = 3)	nfltvc_d = 100
svc1	R (1)	Parameter for the SLEVE coordinate (ivctype = 3)	svc1_d = 8000.0
svc2	R (1)	Parameter for the SLEVE coordinate (ivctype = 3)	svc2_d = 5000.0
linit_realoro	L (1)	If .true., orography height is read from an external ASCII file instead of using analytical formulas like Gauss-hills or similar. The name of the ASCII-file has to be given in the parameter orofile, see below.	linit_realoro_d = .FALSE.
orofile	C(len=250) (1)	Name of the ASCII orography file. If this file name does not contain the absolute path to the file, it is searched in the directory from which the job script is submitted. The domain covered by the file can be larger than the actual model domain. Model domain is centered around the mid point of the domain in the file, as long as i_shift_realoro and j_shift_realoro are not specified. It is possible to add further analytical hills or valleys to this orography.	orofile_d = 'noname.dat'
i_shift_realoro	I (1)	Shift of the model domain in X-direction relative to the center of the ASCII file domain in units of grid points.	i_shift_realoro_d = 0
j_shift_realoro	I (1)	Shift of the model domain in Y-direction relative to the center of the ASCII file domain in units of grid points.	j_shift_realoro_d = 0
href_oro	R (1)	If linit_realoro = .false., this is the constant base height of the orography [m], to which further analytical hills or valleys might be added, depending on 'itype_topo', 'lhill' and others.	href_oro_d = 0.0
itype_topo	I (1)	On top of either the constant base height 'href_oro' or the orography from a file, (additionally) define artificial hills and/or valleys. Current implemented settings are: • itype_topo = 0 : No additional hills/valleys • itype_topo = 1 : Package to define arb. number of hills/valleys, depending on parameters described below (e.g., 'lhill')	itype_topo_d = 1
lhill	L (nhill_max)	For 'itype_topo' = 1: List of switches to define (additional) mountains / valleys. Set as much elements to .true. as you want to have mountains / valleys. See Figure 1 and Figure 2	lhill_d = .FALSE.
lhill_2d	L (nhill_max)	For each mountain / valley set .true. in the 'lhill' list, define if it should be a 2D-mountain (elongated ridge in Y-direction) or a 3D-mountain (ellipsoidal contour lines).	lhill_2d_d = .TRUE.
hill_type	C(len=lcbuf) (nhill_max)	List of types of shape function(s) of the hill profile. Either 'gauss' or 'bellshaped' are implemented, but it is possible to implement own shapes.	hill_type_d = 'gauss'
hill_i	R (nhill_max)	List of moutain center point(s) in i-direction in units of gridpoints. The value(s) might be real numbers, e.g., 27.8	hill_i_d = (ie_tot-1) * 0.5 + 1.0

Name	Kind (Dim.)	Description / Remarks	Default
hill_j	R (nhill_max)	List of moutain center point(s) in j-direction in units of gridpoints. The value(s) might real numbers, e.g., 38.9	hill_j_d = (je_tot-1) * 0.5 + 1.0
hillheight	R (nhill_max)	List of hill height parameter(s) in m (a negative value defines a valley).	hillheight_d = 1000.0
hill_rotangle	R (nhill_max)	List of clockwise rotation angle(s) of the hill main axes with respect to north [degrees]	hill_rotangle_d = 0.0
zhillcutfact	R (nhill_max)	List of relative proportion(s) of the hillheight, which should be chopped from the hill bottom to cut the infinite tails and make the mountain horizontally finite.	zhillcutfact_d = 0.0
hill_combineaction	I (nhill_max)	List of flag(s) to define how to combine the corresponding hill / valley with the previous orography: • hill_combineaction = 1 : add to the previous orography (subtract in the case of a valley) • hill_combineaction = 2 : take the maximum (minimum in case of a valley) of this mountain and the previous orography Note that mountains are processed in the order which they appear in the above list variables!	hill_combineaction_d = 1
hill_width_x	R (nhill_max)	List of mountain / valley half width radii in X-direction (both for 2D- and 3D-hills(s))	hill_width_x_d = 5000.0
hill_width_y	R (nhill_max)	For 3D-mountain(s) (lhill_2d=.false.): List of half width radii in Y-direction. For 2D-mountain(s) (lhill_2d=.true.): List of 1/2 overall width(s), centered around the mountain center in Y-direction	hill_width_y_d = 5000.0
hillsideradius_y	R (nhill_max)	For 2D-mountain(s) (lhill_2d=.true.): List of Y-radii of additional mountain half at either south and north side of the finite ridge to round the sharp edges and avoid the orography jump there.	hillsideradius_y_d = hill_width_x_d
hillasym_x	R (nhill_max)	List of asymmetry parameter(s) in X-direction	hillasym_x_d = 1.0
hillasym_y	R (nhill_max)	List of asymmetry parameter(s) in Y-direction	hillasym_y_d = 1.0
itype_soil_c	I (1)	Parameter to choose the scheme with which the more or less constant soil and surface parameters are initialized: • itype_soil_c = 1 : spatially constant values are specified via below namelist parameters. • itype_soil_c = 2 : spatially varying fields are read from 2D ASCII-files (same format as orography file)	itype_soil_c_d = 1
z0_c	R (1)	If itype_soil_c = 1 : roughness length [m], constant in space and time	z0_c_d = 0.1
fr_land_c	R (1)	If itype_soil_c = 1 : land fraction [-], constant in space and time	fr_land_c_d = 1.0
soiltyp_c	R (1)	If itype_soil_c = 1 : soil type [-], constant in space and time. Possible values are from 1 to 10. For the soil model, only classes 1 to 8 are relevant, whereas 9 (sea water) and 10 (sea ice) also play a role elsewhere (e.g., albedo calculation in the radiation scheme). In the latter two cases, this parameter plays a role also if lsoil=.false.!	soiltyp_c_d = 3.0
plcov_c	R (1)	If itype_soil_c = 1 : plant cover [-], constant in space and time	plcov_c_d = 0.6
lai_c	R (1)	If itype_soil_c = 1 : LAI [-], constant in space and time	lai_c_d = 3.0
rootdp_c	R (1)	If itype_soil_c = 1 : root depth [m], constant in space and time	rootdp_c_d = 0.7
for_e_c	R (1)	If itype_soil_c = 1 : area fraction of evergreen forests [-], constant in space and time	for_e_c_d = 0.2

Name	Kind (Dim.)	Description / Remarks	Default
for_d_c	R (1)	If itype_soil_c = 1 : area fraction of deciduous forests [-], constant in space and time	for_d_c_d = 0.2
h_ice_c	R (1)	If itype_soil_c = 1 : initial sea/lake ice thickness [m], constant in space	h_ice_c_d = 0.1
z0file	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for roughness length [m]; if this file cannot be found in the directory from where you started the model run, you have to provide the full path with the filename.	z0file_d = 'noname.dat'
frlandfile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for land fraction [-]. This file is always read, regardless of lsoil.	frlandfile_d = 'noname.dat'
soiltypefile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for soil type [-]. This file is always read, regardless of lsoil. Possible soil type values range from 1 to 10. For the soil model, only the classes 1 to 8 are relevant, whereas 9 (sea water) and 10 (sea ice) also play a role elsewhere (e.g., albedo calculation in the radiation scheme). In the latter two cases, this parameter plays a role also if lsoil=.false.! Therefore, this file is always required.	soiltypefile_d = 'noname.dat'
plcovfile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for plant cover [-]. This file is always read, regardless of lsoil.	plcovfile_d = 'noname.dat'
laifile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for LAI [-]. This file is always read, regardless of lsoil.	laifile_d = 'noname.dat'
rootdpfile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for root depth [m]. Only read if lsoil=.true.	rootdpfile_d = 'noname.dat'
forefile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for area fraction of evergreen forests [-]. Only read if lforest=.true.	forefile_d = 'noname.dat'
fordfile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for area fraction of deciduous forests [-]. Only read if lforest=.true.	fordfile_d = 'noname.dat'
hicefile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for ice thickness [m]. This file is only read if lseaice=.true.	hicefile_d = 'noname.dat'
ssostdhfile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for the subscale orography standard deviation [m]. Only if lsso=.true.	ssostdhfile_d = 'noname.dat'
ssogammafile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for the anisotropy paramater in the SSO-scheme [-]. Only if lsso=.true.	ssogammafile_d = 'noname.dat'
ssothetafile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for the angle between the principal axis of the orography and East in the SSO-scheme [rad]. Only if lsso=.true.	ssothetafile_d = 'noname.dat'
ssosigmafile	C(len=250) (1)	If itype_soil_c = 2 : Name of the file for the mean slope of sub-grid scale orography in the SSO-scheme. Only if lsso=.true.	ssosigmafile_d = 'noname.dat'
itype_soil_tw	I (1)	This is relevant if using the soil model. Parameter to choose the scheme with which the more time varying soil and surface parameters are initialized: • itype_soil_tw = 1 : spatially constant values are specified via below namelist parameters. • itype_soil_tw = 2 : spatially varying fields are read from 2D ASCII-files (same format as orography file)	itype_soil_tw_d = 1

Name	Kind (Dim.)	Description / Remarks	Default
t_surf_c	R (1)	If itype_soil_tw = 1 : baseline initialization of the model atmospheric surface temperature (called t_s in the code) with a constant value. If t_surf_c < 0, the air temperature from at the surface is used instead, so that minimal sensible heat fluxes would occur. If the soil model is used, it might, depending on other settings (e.g., whether t_soil_c < 0 or not), be overtaken also for all the soil levels at land points. If it is not overtaken by the soil levels, it will be reset to the value of the uppermost soil level during initialization of the soil model. In any case, it is relevant at sea and lake points. But it can be overwritten by t_water_c and/or t_ice_c at water and/or ice points if these parameters are > 0 (if lsoil=.false., ice points include glaciers at land points, soiltype=1). To summarize, it takes effect: • if lsoil=.true., at all land points if t_soil_c < 0, • if lsoil=.true., at all water and lake points if t_water_c < 0, • if lsoil=.true., at all sea ice points, if t_ice_c < 0, • if lsoil=.false., everywhere but might be overwritten by t_water_c at lake or sea points and/or by t_ice_c at sea ice or glacier points. If itype_soil_tw = 2, the pendant is the ASCII file "tsurffile".	t_surf_c_d = -1.0
t_soil_c	R (1)	If itype_soil_tw = 1 : spatially homogeneous soil temperature T_{soil} in [K] (if < 0, temperature t_s is taken instead, which in turn depends on t_surf_c) for all soiltypes except glaciers (soiltyp=1)	t_soil_c_d = -1.0
wf_soil_c	R (1)	If itype_soil_tw = 1 : spatially homogeneous soil water saturation, [-] (0 . . . 1)	wf_soil_c_d = 0.0
t_snow_c	R (1)	If itype_soil_tw = 1 : spatially homogeneous snow temperature T_{snow} in [K] (if < 0, temperature t_s is taken instead, which in turn depends on t_surf_c)	t_snow_c_d = -1.0
w_snow_c	R (1)	If itype_soil_tw = 1 : spatially homogeneous snow water equivalent in [m H_2O]	w_snow_c_d = 0.0
w_i_c	R (1)	If itype_soil_tw = 1 : spatially homogeneous interception storage on plants in [m H_2O]	w_i_c_d = 0.0
t_ice_c	R (1)	If itype_soil_c = 1 : spatially homogeneous T at the snow-ice or air-ice interface [K] (if < 0, temperature t_s is taken instead, which in turn depends on t_surf_c)	t_ice_c_d = 270.0
t_water_c	R (1)	If itype_soil_c = 1 : spatially homogeneous T at the water-ice or water-air interface [K] (if < 0, temperature t_s is taken instead, which in turn depends on t_surf_c)	t_water_c_d = -1.0
tsurffile	C(len=250) (1)	If itype_soil_tw = 2 : Name of the file for the baseline surface temperature T_s [K] (field t_s in the code). It's meaning is analogous to parameter t_surf_c (itype_soil_tw=1), but the values can have a spatial variation. For values ¡ 0, again the atmospheric temperature from right at the surface defines t_s. As a difference, there are no separate ASCII files for the surface temperature of water and ice, therefore tsurffile should contain the correct values for the surface temperatures of all water and ice bodies. Provide the full path if the file cannot be accessed in the directory from where the model run was initiated.	tsurffile_d = 'noname.dat'
tsoilfile	C(len=250) (1)	If itype_soil_tw = 2 : Name of the file for soil temperature T_{soil} [K]	tsoilfile_d = 'noname.dat'
wfsoilfile	C(len=250) (1)	If itype_soil_tw = 2 : Name of the file for soil water saturation, [-] (0 . . . 1)	wfsoilfile_d = 'noname.dat'

Name	Kind (Dim.)	Description / Remarks	Default
tsnowfile	C(len=250) (1)	If itype_soil_tw = 2 : Name of the file for snow temperature T_{snow} in K	tsnowfile_d = 'noname.dat'
wsnowfile	C(len=250) (1)	If itype_soil_tw = 2 : Name of the file for snow water equivalent in [m H_2O]	wsnowfile_d = 'noname.dat'
wifile	C(len=250) (1)	If itype_soil_tw = 2 : Name of the file for interception storage on plants in [m H_2O]	wifile_d = 'noname.dat'
itype_artifprofiles	I (1)	Method of initialisation of (thermo-)dynamic profiles: • itype_artifprofiles = 1 : read profiles from ASCII file • itype_artifprofiles = 2 : analytic only • itype_artifprofiles = 3 : T / Q_v analytic; U / V from file • itype_artifprofiles = 4 : T / Q_v from file, U / V analytic For itype_artifprofiles = 1/3/4, one has to give a radiosonde file name in parameter 'rasofile'. (example: 'raso_wk_q14_u05.dat' in COSMO distribution)	itype_artifprofiles_d = 2
itype_anaprof_tqv	I (1)	For itype_artifprofiles = 2/3, one has to specify the type of the T / Q_v - profiles in 'itype_anaprof_tqv': • itype_anaprof_tqv = 1 : Weisman-Klemp (1982) - type T and Q_v-profiles, determined by the namelist variables with suffix '_wk' • itype_anaprof_tqv = 2 : Arbitrary number of polytrope layers, specified by **namelist vectors with suffix '_poly'**. These layers have a constant vertical T-gradient. • itype_anaprof_tqv = 3 : Arbitrary number of layers with constant Brunt-Vaisala-freq. N, specified by **namelist vectors with suffix '_nconst'**. These layers have a constant N with respect to moist unsaturated air.	itype_anaprof_tqv_d = 2
itype_anaprof_uv	I (1)	For itype_artifprofiles = 2/4, one has to specify the type of the U / V - profiles in 'itype_anaprof_uv': • itype_anaprof_uv = 1 : Weisman-Klemp (1982) - type U - profile $U(z) = U_\infty \tanh\left(\frac{z-\text{hmin\_wk}}{\text{href\_wk}}\right)$ • itype_anaprof_uv = 2 : Arbitrary numbers of constand gradient $U(z)$ - layers • itype_anaprof_uv = 3 : U = const. = u_infty (West-East-Flow) • itype_anaprof_uv = 4 : V = const. = u_infty (South-North-Flow)	itype_anaprof_uv_d = 3
rasofile	C(len=250) (1)	For itype_artifprofiles = 1/3/4: name of the radiosonde file (provide full path if it cannot be accessed from the directory where the model run has been started)	rasofile_d = 'noname.dat'
lps_from_file	L (1)	For itype_artifprofiles = 1/3 : If set to .true., trust the pressure given in the radiosonde file, so that it can be used to interpolate the surface pressure p_s onto the orography. Else, a separate and quite accurate surface pressure calculation is done within subroutine gen_ini_data().	lps_from_file_d = .FALSE.
rasofile_t_is_theta	L (1)	For itype_artifprofiles = 1/3 : If set to .true., indicates that the radiosonde file contains pot. temperature θ instead of ordinary temperature T.	rasofile_t_is_theta_d = .FALSE.
hmin_wk	R (1)	For itype_anaprof_tqv = 1 : profile base height h_{min} in [m] in the profile following Weisman and Klemp (1982): $\theta(z) = \begin{cases} \theta_0 + (\theta_{tr} - \theta_0)\left(\frac{z-h_{min}}{h_{tr}-h_{min}}\right)^{e_\theta} & z < h_{tr} \\ \theta_0 \exp\left(\frac{g}{c_p T_{tr}}(z - h_{tr})\right) & z \geq h_{tr} \end{cases}$ h_{min} must be $\leq$ MINVAL(hsurf)!	hmin_wk_d = 0.0

Name	Kind (Dim.)	Description / Remarks	Default
p_base_wk	R (1)	For itype_anaprof_tqv = 1 : pressure p_base at the height 'hmin_wk' in [Pa]. Serves as reference for the hydrostatic pressure initialization.	p_base_wk_d = 100000.0
h_tropo_wk	R (1)	For itype_anaprof_tqv = 1 : tropopause height in [m], parameter h_{tr}	h_tropo_wk_d = 12000.0
theta_0_wk	R (1)	For itype_anaprof_tqv = 1 : pot. temperature base value θ_0 in [K]	theta_0_wk_d = 300.0
theta_tropo_wk	R (1)	For itype_anaprof_tqv = 1 : constant tropopause pot. temperature θ_{tr} in [K]	theta_tropo_wk_d = 343.0
t_tropo_wk	R (1)	For itype_anaprof_tqv = 1 : Estimate for the tropopause temperature T_{tr} in [K]. Has been set a priori fixed to 217 K in the original literature. If you set it to < -900, the exact tropopause temperature is automatically determined in the program and used instead, which deviates from the above estimate but is the 'correct' value).	t_tropo_wk_d = -999.99
expo_theta_wk	R (1)	For itype_anaprof_tqv = 1 : Exponent e_θ in the θ-profile [-]	expo_theta_wk_d = 1.25
expo_relhum_wk	R (1)	For itype_anaprof_tqv = 1 : Exponent e_{rh} in the relative humidity $R(z)$-profile following Weisman and Klemp (1982): $$R(z) = \begin{cases} R_{max} - (R_{max} - R_{min}) \left(\frac{z-h_{min}}{h_{tr}-h_{min}}\right)^{e_{rf}} & z < h_{tr} \\ R_{min} & z \geq h_{tr} \end{cases}$$	expo_relhum_wk_d = 1.25
rh_min_wk	R (1)	For itype_anaprof_tqv = 1 : Min. relat. humidity R_{min} in the $R(z)$-profile, equals the constant value in the tropopause [-]	rh_min_wk_d = 0.25
rh_max_wk	R (1)	For itype_anaprof_tqv = 1 : relative humidity base value R_{max} [-]	rh_max_wk_d = 1.0
qv_max_wk	R (1)	For itype_anaprof_tqv = 1 : Max. specific humidity, which is imposed after specifying the $R(z)$ profile in [-]. Is intended to result in a well-mixed boundary layer.	qv_max_wk_d = 0.012
nlayers_poly	I (1)	For itype_anaprof_tqv = 2 : Number n_{poly} of the desired polytrope atmosphere layers (i.e., layers with a constant temperature gradient γ). The resulting $T(z)$-profile in the i-th layer will be $$T(z) = \begin{cases} T_{0i} - \gamma_i(z - h_{poly,i}) & h_{poly,i} \leq z < h_{poly,i+1},\ i < n_{poly} \\ T_{0i} - \gamma_i(z - h_{poly,i}) & h_{poly,i} \leq z \leq z_{top},\ i = n_{poly} \end{cases}$$ where z_{top} is the model domain height parameter 'zz_top' from above. See Figure 3	nlayers_poly_d = 3
p_base_poly	R (1)	For itype_anaprof_tqv = 2 : Base pressure p_{base} at the lowest level $h_{poly,1}$ in [Pa]	p_base_poly_d = 100000.0
h_poly	R (nlayers_poly_max)	For itype_anaprof_tqv = 2 : List of polytrope layer base heights $h_{poly,1}$,...,$h_{poly,n_{poly}}$ in [m]. Must contain at least n_poly elements!	h_poly_d(:) = HUGE(1.0) h_poly_d(1) = 0.0 h_poly_d(2) = 2000.0 h_poly_d(3) = 11000.0
t_poly	R (nlayers_poly_max)	For itype_anaprof_tqv = 2 : List of polytrope layer base temperatures T_1, ...,T_{n_poly} in [K].	t_poly_d(:) = 288.16 t_poly_d(1) = 288.16 t_poly_d(2) = 275.16 t_poly_d(3) = 216.66
tgr_poly	R (nlayers_poly_max)	For itype_anaprof_tqv = 2 : List of polytrope layer T-gradients γ_1,..., $\gamma_{n_{poly}}$ in [K/m]. Positive for decreasing T with height!	tgr_poly_d(:) = 0.0 tgr_poly_d(1) = 0.0095 tgr_poly_d(2) = 0.0065 tgr_poly_d(3) = 0.0

Name	Kind (Dim.)	Description / Remarks	Default
rh_poly	R (nlayers_poly_max)	For itype_anaprof_tqv = 2 : List of relative humidity base values $R_1,\ldots,$ $R_{n_{poly}}$ of the polytrope layers. The resulting $R(z)$-profile in the i-th layer will be $$R(z) = \begin{cases} R_{0i} - \gamma_{rf,i}(z - h_{poly,i}) & h_{poly,i} \le z < h_{poly,i+1},\ i < n_{poly} \\ R_{0i} - \gamma_{rf,i}(z - h_{poly,i}) & h_{poly,i} \le z \le z_{top},\ i = n_{poly} \end{cases}$$	rh_poly_d(:) = 0.5 rh_poly_d(1) = 0.0 rh_poly_d(2) = 0.0 rh_poly_d(3) = 0.0
rhgr_poly	R (nlayers_poly_max)	For itype_anaprof_tqv = 2 : List of polytrope layer R-gradients $\gamma_{rf,1},\ldots,$ $\gamma_{rf,n_{poly}}$ in [1/m]. Positive for decreasing R with height!	rhgr_poly_d(:) = 0.0 rhgr_poly_d(1) = -0.0 / 2000.0 rhgr_poly_d(2) = 0.0 / 9000.0 rhgr_poly_d(3) = 0.0
nlayers_nconst	I (1)	For itype_anaprof_tqv = 3 : Number n_{nc} of the desired layers with a constant Brunt-Vaisala-frequency N, i.e., $$N(z) = \begin{cases} N_i \ge 0 & h_{nc,i} \le z < h_{nc,i+1},\ i < n_{nc} \\ N_i \ge 0 & h_{nc,i} \le z \le z_{top},\ i = n_{nc} \end{cases}$$ resulting in a corresponding temperature profile $T(z)$. See Figure 5 and Figure 6	nlayers_nconst_d = 3
p_base_nconst	R (1)	For itype_anaprof_tqv = 3 : Base pressure p_{base} at the lowest level $h_{nc,1}$ in [Pa]	p_base_nconst_d = 100000.0
theta0_base_nconst	R (1)	For itype_anaprof_tqv = 3 : Pot. temperature at the level $h_{nc,1}$ in [K]	theta0_base_nconst_d = 300.0
h_nconst	R (nlayers_nconst_max)	For itype_anaprof_tqv = 3 : List of height level values $h_{nc,1},\ldots,$ $h_{nc,n_{nc}}$ in [m]	h_nconst_d(:) = HUGE(1.0) h_nconst_d(1) = 0.0 h_nconst_d(2) = 1500.0 h_nconst_d(3) = 12000.0
N_nconst	R (nlayers_nconst_max)	For itype_anaprof_tqv = 3 : List of constant N-values for the layers, $N_1,\ldots,$ $N_{n_{nc}}$, in [1/s]	N_nconst_d(:) = 0.01 N_nconst_d(1) = 0.001 N_nconst_d(2) = 0.01 N_nconst_d(3) = 0.02
rh_nconst	R (nlayers_nconst_max)	For itype_anaprof_tqv = 3 : List of relative humidity base values $R_1,\ldots,$ $R_{n_{poly}}$ of the N =constant layers. The profile will be linear as in the polytrope layers case above (itype_anaprof_tqv = 2).	rh_nconst_d(:) = 0.0
rhgr_nconst	R (nlayers_nconst_max)	For itype_anaprof_tqv = 3 : List of N=constant layers R-gradients $\gamma_{rf,1},\ldots,$ $\gamma_{rf,n_{poly}}$ in [1/m]. Positive for decreasing R with height!	rhgr_nconst_d(1) = 0.0 rhgr_nconst_d(2) = 0.0 rhgr_nconst_d(3) = 0.0
href_wk	R (1)	For itype_anaprof_uv = 1 : Reference height h_{ref} of the wind profile after Weisman and Klemp (1982) in [m]: $$U = U_\infty \tanh\left(\frac{z - h_{min}}{h_{ref} - h_{min}}\right)$$ h_{min} is the parameter 'hmin_wk' from above! Wind is positive from West!	href_wk_d = 3000.0
u_infty	R (1)	For itype_anaprof_uv = 1 : Free-troposphere value U_∞ of the U wind component in [m/s]	u_infty_d = 20.0
nlayers_linwind	I (1)	For itype_anaprof_uv = 2 : Number n_{lw} of desired layers with constant windspeed gradient. The corresponding (westerly) wind profile will be: $$U(z) = \begin{cases} U_{0,i} + \gamma_{u,i}(z - h_{lw,i}) & h_{lw,i} \le z < h_{lw,i+1},\ i < n_{lw} \\ U_{0,i} + \gamma_{u,i}(z - h_{lw,i}) & h_{lw,i} \le z \le z_{top},\ i = n_{lw} \end{cases}$$ where z_{top} is again the model domain height parameter 'zz_top' from above. See Figure 4	nlayers_linwind_d = 2

Name	Kind (Dim.)	Description / Remarks	Default
h_linwind	R (nlayers_linwind_max)	For itype_anaprof_uv = 2 : List of constant-gradient windspeed layer base heights $h_{lw,1},\ldots,h_{lw,n_{lw}}$ in [m]. Must contain at least n_lw elements!	h_linwind_d(:) = HUGE(1.0) h_linwind_d(1) = 0.0 h_linwind_d(2) = 2500.0
u_linwind	R (nlayers_linwind_max)	For itype_anaprof_uv = 2 : List of windspeed base values $U_{0,1},\ldots,U_{0,n_{lw}}$ in [m/s]	u_linwind_d(:) = 0.0 u_linwind_d(1) = 0.0 u_linwind_d(2) = 20.0
ugr_linwind	R (nlayers_linwind_max)	For itype_anaprof_uv = 2 : List of windspeed gradients $\gamma_{u,1},\ldots,\gamma_{u,n_{lw}}$ in [1/s]. Positive for increasing windspeed with height!	ugr_linwind_d(:) = 0.0 ugr_linwind_d(1) = 20.0 / 2500.0 ugr_linwind_d(2) = 0.0
linitw_followeta	L (1)	If set to .true., initialization of W (not part of any analytical formula or radiosonde file) will be in such a way that the streamlines follow the terrain following coordinate surfaces of the model. If set to .false., $W = 0$ results everywhere in the domain.	linitw_followeta_d = .FALSE.
zo_boundary	R (1)	Upper height z_o above the surface in [m] for imposing an artificial windspeed boundary layer. Enabled, if $z_o > 0$. The resulting windspeed profile $\|v_h\|(z)$ in the boundary layer will be $$\|v_h\|(z) = \|v_h\|_{z_o}\left(\frac{z-h_{surf}}{z_o}\right)^{e_{bl}} \qquad h_{surf} \le z \le h_{surf}+z_o$$ h_{surf} is the model surface height. Makes sense if there is a non-vanishing windspeed at the surface in the initial profile and, at the same time, a no-slip-surface boundary condition is applied, to avoid spurious spin up at constant-inflow relaxation boundaries.	zo_boundary_d = 0.0
exponent_windprof_boundary	R (1)	In case of zo_boundary > 0, the exponent e_{bl}. Typical values range from 0.1 (stable atmosphere over water) to about 0.4 (very rough surface).	exponent_windprof_boundary_d = 0.333
tkvhfix	R (1)	If itype_turb = 100 (namelist INPUT_PHY): Constant value for the vertical turbulent diffusion coefficient for heat and moisture in [m^2/s]. This value will be used everywhere at any time in the model run, bypassing the turbulence scheme.	tkvhfix_d = 300.0
tkhhfix	R (1)	If itype_turb = 100 (namelist INPUT_PHY): Constant value for the horizontal turbulent diffusion coefficient for heat and moisture in [m^2/s]. This value will be used everywhere at any time in the model run, bypassing the turbulence scheme. Takes effect when l3dturb = .true. (namelist INPUT_PHY).	tkhhfix_d = 300.0
tkvmfix	R (1)	If itype_turb = 100 (namelist INPUT_PHY): Constant value for the vertical turbulent diffusion coefficient for momentum in [m^2/s]. This value will be used everywhere at any time in the model run, bypassing the turbulence scheme. Takes effect when l3dturb = .true. (namelist INPUT_PHY).	tkvmfix_d = 300.0
tkhmfix	R (1)	If itype_turb = 100 (namelist INPUT_PHY): Constant value for the horizontal turbulent diffusion coefficient for momentum in [m^2/s]. This value will be used everywhere at any time in the model run, bypassing the turbulence scheme. Takes effect when l3dturb = .true. (namelist INPUT_PHY).	tkhmfix_d = 300.0

Name	Kind (Dim.)	Description / Remarks	Default
lsensiflux_fix	L (1)	Master switch to turn on a time- and space- constant surface sensible heat flux. This is only possible if the soil model is turned of (lsoil = .false.) and works with the surface transfer schemes itype_tran = 1 and 2. The magnitude of the heat flux is given by the below namelist parameter sensiflux_c. It is also possible to also specifiy a non-zero latent heat flux. For this, set the namelist parameter llatentflux_fix = .true. and choose the Bowen-ratio latentflux_LzuS (see below). It is further possible to overlay white noise onto both fluxes with a relative amplitude H0_rel_noise (namelist parameter, see below).	lsensiflux_fix_d = .FALSE.
sensiflux_c	R (1)	If lsensiflux_fix = .true., this is the constant heat flux in [Wm^{-2}]. Positive values are upward fluxes.	sensiflux_c_d = 0.0
llatentflux_fix	L (1)	If lsensiflux_fix = .true., an additional latent heat flux can be switched on. Its magnitude has to be defined by the Bowen ratio, which is a namelist parameter (see below).	llatentflux_fix_d = .FALSE.
latentflux_LzuS	R (1)	If lsensiflux_fix = .true. and llatentflux_fix = .true., this specifies the ratio of latent to sensible heat flux (Bowen-Ratio).	latentflux_LzuS_d = 0.4
H0_rel_noise	R (1)	If lsensiflux_fix = .true., white noise with the relative amplitude H0_rel_noise can be overlayed over the sensible and latent heat fluxes (same noise function for both). The noise is spatial and is constant in time at fixed locations. If H_{noisy} denotes the noisy flux, H_0 the constant flux and α_{noise} the noise level H0_rel_noise, it is $$H_{noisy} = H_0(1 + \alpha_{noise}\epsilon_{[-1,1]})$$ $\epsilon_{[-1,1]}$ denotes white noise in the range $[-1, 1]$. The noise will be applied if H0_rel_noise $\geq$ 1E-5.	H0_rel_noise_d = 0.0
iseed_noise_H0	I (1)	If lsensiflux_fix = .true. and H0_rel_noise $\geq$ 1E-5 (noise overlay), this gives the seed for the random number generator. If the seed -999 is given, the system clock is used instead and the noise will be different for every new model run.	iseed_noise_H0_d = 608
lnosurffluxes_m	L (1)	If set to .true., the transfer coefficient for momentum tcm is set to 0.0 everywhere and any time. This represents a free-slip surface boundary condition and bypasses the surface transfer scheme.	lnosurffluxes_m_d = .FALSE.
lnosurffluxes_h	L (1)	If set to .true., the transfer coefficient for heat and moisture tch is set to 0.0 everywhere and any time. This represents a NO-heat-and-moisture-flux surface boundary condition and bypasses the surface transfer scheme.	lnosurffluxes_h_d = .FALSE.
ltempdist	L (ntempdist_max)	List of master switches to enable temperature- or heat flux disturbances in the initial- or boundary conditions relating to atmospheric T, surface T_s or soil T_{soil}, depending on the type of disturbance. The type is chosen by the character string list 'ctype_tempdist', where each element corresponds to the respective switch in the 'ltempdist' list.	ltempdist_d = .FALSE.
ctype_tempdist	C(len=lcbuf) (ntempdist_max)	List of type(s) of temperature disturbance(s); corresponding to the switch list 'ltempdist'. **For a more detailed discussion of the available different types of disturbances, consult Section 5 in this document (artif_docu.pdf)!**	ctype_tempdist_d = 'cos'
htempdist	R (ntempdist_max)	List of time(s) of (first) occurence of the disturbance in hours since model start.	htempdist_d = 0.0
bub_centi	R (ntempdist_max)	List of center(s) of the disturbance in in X-direction in units of gridpoints (real numbers allowed)	bub_centi_d = (ie_tot-1) * 0.5 + 1.0

Name	Kind (Dim.)	Description / Remarks	Default
bub_centj	R (ntempdist_max)	List of center(s) of the disturbance in in Y-direction in units of gridpoints (real numbers allowed)	bub_centj_d = (je_tot-1) * 0.5 + 1.0
bub_centz	R (ntempdist_max)	List of height(s) of center(s) of disturbance(s) in [m] above MSL. Effective for bubble types 'cos', 'squall3D', 'cos-hrd'.	bub_centz_d = 1400.0
bub_centk	I (ntempdist_max)	List of height(s) of center(s) of disturbance(s), given as height level index. Only INTEGER numbers allowed! Effective for bubble type 'AS2005_hucmtexas-hrd'.	bub_centk_d = 5
bub_timespan	I (ntempdist_max)	List of time span(s) of disturbance(s) in units of time steps. Effective for all constant-heating-rate type disturbances or for surface T_s-disturbance with deactivated soil model, type 'hotspot-sfc'.	bub_timespan_d = 100
bub_radx	R (ntempdist_max)	List of radius /half width(s) parameter(s) in X-direction of the disturbance(s) in [m]	bub_radx_d = 10000.0
bub_rady	R (ntempdist_max)	List of radius/ half width(s) parameter(s) in Y-direction of the disturbance(s) in [m]	bub_rady_d = 10000.0
bub_radz	R (ntempdist_max)	List of vertical radius/ half width(s) parameter(s) of the disturbance(s) in [m]	bub_radz_d = 1400.0
bub_rotangle	R (ntempdist_max)	List of the counter-clockwise horizontal rotation angle(s) of the main axes of the disturbances in degrees.	bub_rotangle_d = 0.0
bub_dT	R (ntempdist_max)	List of T-disturbance amplitude parameters ΔT_{bub} in [K]. Effective for all disturbance types which directly specify a T-disturbance.	bub_dT_d = 3.0
bub_heatingrate	R (ntempdist_max)	List of constant heating rate(s) amplitude parameters in [K/s]. Effective for all disturbance types which directly specify a heating rate disturbance.	bub_heatingrate_d = 0.005
lbub_rhconst	L (ntempdist_max)	List of switches to determine, how to treat the relative humidity when specifying T- or heating rate disturbances. If .true., the relative humidity will kept constant during altering the temperature. If .false., the specific humidity q_v will be kept constant.	lbub_rhconst_d = .FALSE.
ladd_bubblenoise_t	L (ntempdist_max)	List of switches to overlay some noise on the disturbances. If set .true., white noise will be applied in a multiplicative/ relative sense with an amplitude parameter specified by bub_dT_bubblenoise below. Effective for some of the disturbance types only.	ladd_bubblenoise_t_d = .FALSE.
bub_dT_bubblenoise	R (ntempdist_max)	List of relative amplitude parameter(s) α_{noise} [-] of the additional noise on disturbance amplitudes in the sense of: $\Delta T_{bub,noisy} = \Delta T_{bub}(1 + \alpha_{noise}\epsilon_{[-1,1]})$ where ΔT_{bub} represents the undisturbed 'bub_dT' amplitude parameter from above and $\epsilon_{[-1,1]}$ denotes white noise in the range $[-1, 1]$.	bub_dT_bubblenoise_d = 0.1
bub_zi_mcnider	R (ntempdist_max)	List of boundary layer height parameter(s) in [m] for 'mcnider'- and 'mcnider-hrd'-type disturbances (McNider and Kopp, JAM, 1990)	bub_zi_mcnider_d = 4000.0
bub_zmax_mcnider	R (ntempdist_max)	List of width scaling parameter(s) in [m] for 'mcnider'- and 'mcnider-hrd'-type disturbances (McNider and Kopp, JAM, 1990)	bub_zmax_mcnider_d = 6000.0
bub_h0_mcnider	R (ntempdist_max)	List of surface heat flux parameter(s) in [W/m^2] for 'mcnider'- and 'mcnider-hrd'-type disturbances (McNider and Kopp, JAM, 1990)	bub_h0_mcnider_d = 600.0

Name	Kind (Dim.)	Description / Remarks	Default
ladd_noise_t	L (1)	If set to .true., apply white noise (in an additive sense) to the T- and W field in the lowest 100 hPa of the atmosphere, so that the T disturbance is: $$T'(z) = \Delta T_{noise} \epsilon_{[-1,1]} \cos\left(\frac{\pi}{2} \frac{p_{surf} - p(z)}{100\,\mathrm{hPa}}\right)$$ for $p_{surf} - p(z) < 100$ hPa. $\epsilon_{[-1,1]}$ represents white noise in the interval $[-1, 1]$. The W-disturbance is analogue, with amplitude parameter ΔW_{noise}. The noise is applied at a time specified by 'hadd_noise' below, and the seeds to initialize the random number generators can be given by 'iseed_noise_t' and 'iseed_noise_w', respectively.	ladd_noise_t_d = .FALSE.
hadd_noise	R (1)	Time in hours since model start for specifying the white noise.	hadd_noise_d = 0.0
dT_noise	R (1)	Amplitute parameter ΔT_{noise} in [K] for T-noise.	dT_noise_d = 0.02
iseed_noise_t	I (1)	Seed for the random number generator for the T-noise. If the seed -999 is given, the system clock is used instead and the noise will be different for every new model run.	iseed_noise_t_d = 606
dW_noise	R (1)	Amplitute parameter ΔW_{noise} in [m/s] for W-noise.	dW_noise_d = 0.02
iseed_noise_w	I (1)	Seed for the random number generator for the W-noise. If the seed -999 is given, the system clock is used instead and the noise will be different for every new model run.	iseed_noise_w_d = 607
hcond_on	R (1)	Time in hours since model start to activate the cloud microphysics and cloud condensation processes. Takes effect, if hcond_on > 0 only. Then, regardless of the initial settings for 'lgsp' (namelist INPUT_PHY) and 'lcond' (namelist DYNCTL), these two parameters will be automatically set to .false. at the model start and will resume their initial namelist parameter settings after the time 'hcond_on'. This is helpful for, e.g., allowing the model to spin up mountain wave flow over a certain time period without triggering clouds and perhaps convection.	hcond_on_d = 0.0

References

Doms, G., 2011: *A Description of the Nonhydrostatic Regional COSMO-Model LM. Part I: Dynamics and Numerics*, Consortium for Smallscale Modeling (COSMO), available online: http://www.cosmo-model.org/content/model/documentation/core/cosmoDyncsNumcs.pdf, 147 pp.

Herzog, H.-J., U. Schubert, G. Vogel and A. Fiedler, 2002a: LLM - the high-resolving nonhydrostatic simulation model in the DWD-project LITFASS. part I: Modelling technique and simulation method, *Technical Report 4*, Consortium for Small Scale Modeling (COSMO), URL `http://www.cosmo-model.org/content/model/documentation/techReports/docs/techReport04.pdf`.

Herzog, H.-J., G. Vogel and U. Schubert, 2002b: LLM – a nonhydrostatic model applied to high-resolution simulations of turbulent fluxes over heterogeneous terrain, *Theor. Appl. Climatol.*, **73**, 67–86, doi:10.1007/s00704–002–0694–4.

Schättler, U., 2012: *A Description of the Nonhydrostatic Regional COSMO-Model. Part VII: User's Guide*, Consortium for Smallscale Modeling (COSMO), available online: http://www.cosmo-model.org/content/model/documentation/core/cosmoUserGuide.pdf, 192 pp.

Weisman, M. L. and J. B. Klemp, 1982: The dependence of numerically simulated convective storms on vertical wind shear and buoyancy, *Mon. Wea. Rev.*, **110**, 504–520.