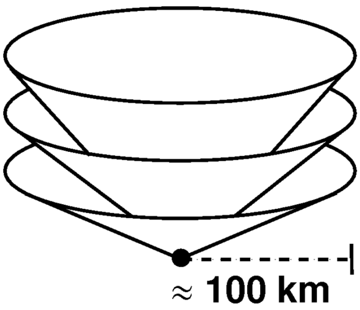
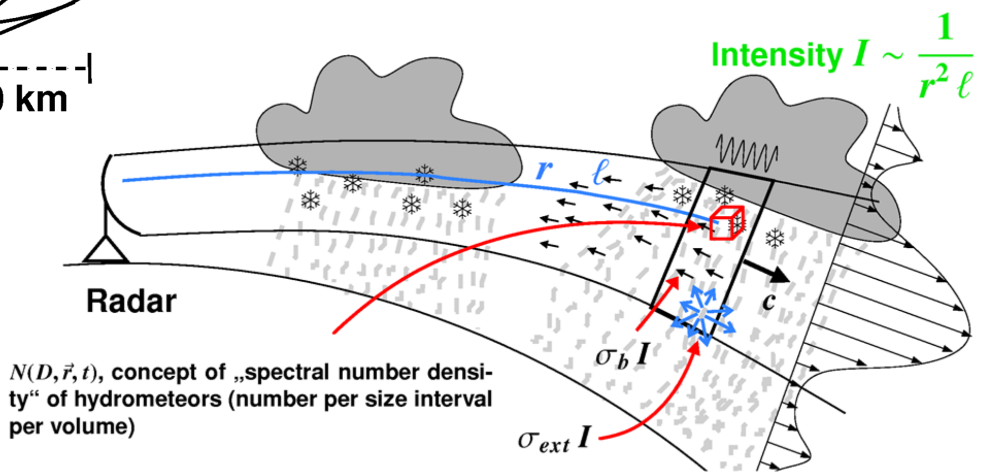
EMVORADO — Efficient Modular VOlume scan RADar Operator

A User's Guide

Ulrich Blahak, Alberto de Lozar

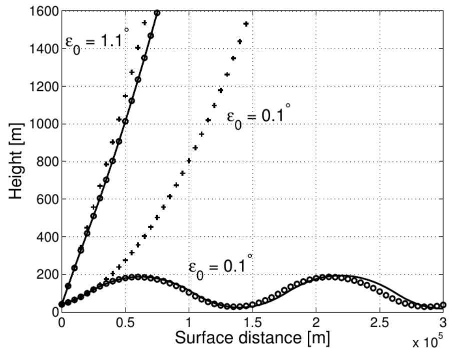
July 21, 2020

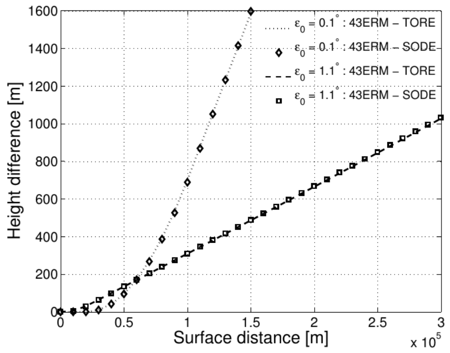




$$Z_e \sim \int_0^\infty \sigma_b(D) N(D) dD \quad \rightarrow$$

$$\text{„Reflectivity“ [dBZ]} = 10 \log_{10} \left(\frac{Z_e}{1\, mm^6 m^{-3}} \right)$$





Contents

1 List of changes

Date	Name	Change
18.9.2019	Ulrich Blahak	Eliminated namelist switch `lvoldata_output_supob` from `/RADARSIM_PARAMS/`. This switch is not needed, because the voldata output of the superobserved fields `'vrsupobs'`, `'vrsupsim'`, `'zrsupobs'`, and `'zrsupsim'` can also be triggered via `voldata_output_list`. Eliminated from Table 4.
18.9.2019	Ulrich Blahak	New namelist parameters `itype_obserr_vr` and `thres_dbz_4_obserr_weight` in `/RADARSIM_PARAMS/` to choose the method to define the observation error for radial wind which is stored in the feedback files (fof) for data assimilation. Inserted into Table 4.
19.9.2019	Ulrich Blahak	New namelist switch `labort_if_problems_obsfiles` in `/RADARSIM_PARAMS/`. Inserted into Table 4.
14.10.2019	Ulrich Blahak	New namelist parameters in `/RADARSIM_PARAMS/` for defining the parameters of the automatic bubbles of the bubble generator. Inserted into Table 4. Updated section 7.2 accordingly.
02.02.2020	Ulrich Blahak	New namelist parameter `levelidlist_for_composite_glob` in `/RADARSIM_PARAMS/` for grib2's levelindex to identify the composites in grib2 records. Inserted into Table 4. Updated section 5.1.6.
14.02.2020	Ulrich Blahak	New namelist parameter `lwrite_ready` in `/RADARSIM_PARAMS/` for writing READY-files after each EMVORADO output time step. Inserted into Table 4. New according section 6.3.
20.04.2020	Ulrich Blahak	Eliminated `thres_dbz_4_obserr_weight`. New namelist parameters `baseval_obserr_vr`, `maxval_obserr_vr`, `ramp_lowdbz_obserr_vr`, `ramp_highdbz_obserr_vr` in `/RADARSIM_PARAMS/`. Inserted into Table 4. According changes in section 6.1.2.
20.07.2020	Ulrich Blahak	"Warm bubble" generator now also implemented in ICON. Updated description of corresponding section(s).

2 Introduction

2.1 Basic information

The **E**fficient **M**odular **VOLume RAD**ar forward **O**perator **EMVORADO** computes synthetic radar volume scan observations of radial wind ($\mathrm{m\,s^{-1}}$) and radar reflectivity (dBZ) on the basis of the simulated prognostic atmospheric fields of NWP-models for a given set of radar stations. From a scientific point of view it is documented in Blahak (2016), Zeng (2013), Jerger (2014), Zeng et al. (2014) and Zeng et al. (2016).

Radar data can be output for different purposes and in different formats. The “raw” volume data are useful for model verification or a postprocessing by methods/software from the radar community, such as compositing or the detection of simulated convective “cell objects”. EMVORADO is able to produce and output it’s own reflectivity composites. It can also process observed volume scans alongside the simulated data to produce so-called “feedback” files as input for DWD’s KENDA data assimilation system, which contain pairs of observed and simulated reflectivities. From the composites of observed and simulated reflectivites EMVORADO offers the option to detect missing convective cells in the hosting model and to provide informations for automatic artificial convection triggers (“warm bubble generator”) to spin up missing convective cells in the model.

Use of the forward operator can be switched on by the top level namelist parameter `luse_radarfwo` in one of the top-level namelist in the hosting model (COSMO: `/RUNCTL/`; ICON: `/run_nml/`), provided that the model has been compiled with a number of additional pre-processor switches, cf. Section 3.

EMVORADO provides options for different degrees of “physical” approximation for certain scattering- and radar measurement processes, to provide the possibility to find the “best” compromise between necessary physical accuracy and computational efficiency for the user’s respective application. These options are described in detail in Zeng et al. (2016) and can be configured in an operator-specific namelist `/RADARSIM_PARAMS/` (COSMO: file `INPUT_RADARSIM`, ICON: file `NAMELIST_EMVORADO`).

The present document provides a short overview on the operator and its different application modes (Section 4), its different namelists (operator-specific and hosting model) in Section 5, data formats for radar observation input (Section 4.3) and operator output (Section 6), code compilation (Section 3), as well as general aspects of operator development (Section 8), including implementation into hosting models in general (Section 8.1) and in COSMO (Section 8.2) and ICON (Section 8.3).

Concerning the code, EMVORADO is written in Fortran 2003. Most modules are not specific to a particular NWP-model but can in principle be coupled to any NWP-model. To connect the model world to the EMVORADO world, there is one model-specific module `radar_interface.f90` which, for example, exchanges time informations, provides interpolation routines (model grid $\Longrightarrow$ geographic coordinates $\Longrightarrow$ radar bins), exchanges MPI communicators and -data types, and so on. Another module `radar_mie_iface_cosmo.f90` to compute grid point values such as reflectivity based on the COSMO- and ICON cloud microphysics schemes is also model specific. More details on the implementation and coupling can be found in Section 8.1.

EMVORADO is designed to handle entire networks of radar stations and allows different wavelengths, scan strategies and output times for different stations. These and the other relevant station metadata (geographic location, height, etc.) may be specified explicitly in the `/RADARSIM_PARAMS/` namelist or taken from actual observation files (if available). With the namelist-based station definition method, even idealized simulations with OSMO and virtual radar stations are possible (e.g. OSSE studies for scientific questions in data assimilation). As mentioned above, these different simulation modes are described in more detail in Sections 4.

With regards to its contents, the operator for reflectivity is composed of two steps.

Step 1: Computation of unattenuated reflectivity and attenutation coefficient (optional for Mie-scattering option, see below) on the model grid based on the simulated hydrometeor fields (depending on the chosen microphysics scheme) as described in Blahak (2016) and Zeng et al. (2016). There are 3 general options, the first is Mie-scattering and second and third are two variants of the so-called Rayleigh approximation for particles small compared to the wavelength. The main difference of the two Rayleigh options is the treatment of melting hydrometeors.

Step 2: Interpolation/aggregation of the reflectivity and attenuation coefficient to the radar bins (polar coordinates range, azimut, elevation), optionally iterative computation of attenuation effects along each ray (only possible with the Mie-option), starting at the radar station, and output of volume scan data to files. Step 1 depends on the radar wavelength (also slightly in the first of the two Rayleigh options), and because different radar stations are allowed to have different wavelengths, it is repeated for each single radar station.

However, if consecutive simulated radars have the same wavelength and other reflectivity-specific settings, some efficiency is gained by just re-using the computed reflectivity from the last radar instead of repeating the computation.

The operator for radial wind is similar, but if some grid point calculations (step 1) are necessary or not depends on the physical configuration. Step 1 is only necessary, if the reflectivity weighted terminal fall velocity of hydrometeors is taken into account in the radial winds. If yes, this velocity is derived along the lines of the grid point reflectivity from the model hydrometeor fields. The reflectivity for weighting the fall velocity is however always according to the Rayleigh approximation, see Zeng et al. (2016). Step 2: Interpolate/aggregate U, V, W to the radar bin positions (polar radar coordinates range, azimut and elevation), compute the radial wind as described in Zeng et al. (2016) and output volume scan data to files. This is similar to step 2 for the reflectivty.

2.2 Remark about online domain nesting capabilities

For models like ICON with the capability for online nesting runs with multiple model domains, EMVORADO is ready to be applied independently to the different model domains. In this case, the namelist parameter `luse_radarfwo` is a list of switches for each model domain. Each domain for which `luse_radarfwo(`k`)=.TRUE.` is called "radar-active" in the following.

An own `/RADARSIM_PARAMS/` for each radar-active model domain is required. There is a mandatory namelist parameter `dom` (domain number, integer) to indicate which namelist belongs to which model domain. In COSMO we always have to set `dom=1`. If this parameter is missing in the `/RADARSIM_PARAMS/` namelist, the run will stop with an appropriate error message.

2.3 Parallelization

The operator code is parallelized as described in Zeng et al. (2016). The details of the parallelization strategy depend on the optionally chosen degree of physical approximation of the radar measurement process, in particular if the beam propagation is modeled by the simple climatological "4/3-earth-model" (sufficient for most applications) or by an actual ray-tracing method based on the simulated actual field of the air refractive index. More details about the underlying physics can be found in Zeng et al. (2014).

Technically, the step 2 for both radial wind and reflectivity is divided into two sub-parts, related to the parallelization strategy. Each compute-PE contains a subset of the radar bins of each station, depending on the horizontal domain decomposition (number of compute PEs in X- and Y-direction) and the position and scan strategies of the radar stations. The polar radar bins of one station might or might not be spread over several compute-PEs. In the first sub-part, the interpolation/aggregation of model grid point values to the subset of radar bins is done by each compute-PE separately in parallel. In the second sub-part, the subsets on different PEs for each station are collected on dedicated output-PEs, one for each station and sorted into full 3D volume scan arrays (range, azimut, elevation). At this point, additional computations which require continuous and sorted data (e.g., optional iterative attenuation computation along each ray, super-observations for the data assimilation feedback files, radar composites of observed and simulated reflectivity) can be perfomed, before the data are output into files.

Normally these output processors are among the compute processors, i.e., if the hosting model runs on 100 PEs and there are 17 radar stations to simulate, the first 17 compute PEs will do the output for the 17 stations in parallel. There can be however less processors than stations, in which case each processor does output for several stations one after another in a round-robin fashion. Even single-processor-runs are possible.

This "synchroneous" strategy has however the effect that the non-output PEs have to wait for the output PEs to finish their computations and data output before the COSMO-model can continue with the next model time step. This leads to a more or less strong load imbalance among the compute-PEs, because not only the output can be costly, but also the abovementioned additional computations before the actual output.

Therefore, there is also an asynchroneous mode in the EMVORADO code where the radar output processors can be extra PEs dedicated exclusively to radar data IO, similar to the already existing asynchroneous grib output of the COSMO-model. This mode can be activated by simply specifying the desired extra number of PEs as the namelist parameter `nprocio_radar` in COSMO (namelist `/RUNCTL/`) or `num_io_procs_radar` in ICON (namelist `/parallel_nml/`). Here, the compute-PEs can continue with their model time step integration while the output-PEs are doing their output tasks in parallel. If more than one model domain is radar-active, the asynchroneous radar-IO-PE's are automatically sub-divided over the number of radar-active domains.

From a technical standpoint, the ray-tracing method requires an intermediate parallelization step. The atmospheric refractive index n (function of T, p and q_v), the wind components and the grid point reflectivity are interpolated to so-called azimuthal slices. These are vertical 2D slices extending radially outwards from each radar station for all discrete azimut angles of the radar scan strategy. For the ray tracing, the complete data of one azimutal slice have to be present on one processor, which is achieved in a special MPI communication. Moreover, the set of slices from all stations is evenly distributed over the compute-PEs to do the ray tracing for the radar bin heights as function of range in parallel and to interpolate the wind components and reflectivity to these bin positions.

If the so-called beam function smoothing option is enabled, which is, for each radar bin value, a weighted integral averaging procedure using a Gaussian kernel over some neighborhood volume, the radar bin values in sub-part 1 of step 2 are in fact the values at some auxiliary Gauss-Legendre-nodes around each radar bin center, but otherwise the same parallelization strategy is employed. The actual integral averaging is done in sub-part 2 on the output-PEs, after the optional attenuation computation.

2.4 Options for reflectivity computation of different physical accuracy and efficiency

The choice of the reflectivity computation method (cf. Section 2.1 above) is governed by a namelist parameter `itype_refl` that appears in different contexts, also as derived type component, in radar-related namelists. A number of sub-parameters govern some intrinsic details of the computation method, mostly for the Mie-option. An in-depth discussion and documentation, especially with regards to melting hydrometeors (radar "bright band") can be found in Chapter 6 of Blahak (2016). The parameters described therein appear below in Sections 5.1.4 and 5.2.

For Mie-scattering, the use of efficient lookup tables replacing the full expensive computations is implemented and strongly advised (cf. Section 2.6 below). With this, Mie-scattering becomes computationally as cheap as the Rayleigh options. The namelist switch `llookup_mie=.TRUE.` activates the use of lookup tables and also appears in different contexts in namelists. The tables themselves are autogenerated by EMVORADO if necessary; the full Mie-routines are included in the code.

A "normal" user should only choose the radar wavelength and the overall type of reflectivity computation. The following options are provided in EMVORADO:

- `itype_refl = 1`: Mie-scattering option assuming spherical particles for all hydrometeors. This is the preferred option in EMVORADO because particle sizes are allowed to be larger than the wavelength. Also, a detailed treatment of melting hydrometeors (cf. Blahak, 2016, Sections 4, 5, 8) allows for halfway realistic bright bands, at the same time allowing to explore the wide range of uncertainty by choosing many options for the refractive index of ice/water/air mixture particles (so-called Effective Medium approximations or EMA's).

  If switching on the lookup table option (`llookup_mie=.TRUE.`), very good efficiency is achieved, because the (autgenerated) tables directly relate reflectivity to the prognostic hydrometeor moments and temperature, and, concerning efficiency, there is no reason to choose the below Rayleigh approximations any more.

- `itype_refl = 3`: Rayleigh-scattering approximation using the Oguchi-formulation of the effective refractive index of melting hydrometeors as described in Blahak (2016), Section 6.2. It contains an analytic formulation for melting hydrometeors which leads to a systematic underestimation of bright bands. Reflectivity is overestimated, if the particle sizes are comparable or larger than the radar wavelength.

- `itype_refl = 4`: "Old" standard method for reflectivity computation from `pp_utilities.f90`. It is also a Rayleigh-scattering approximation, but it contains only a very simplistic treatment of melting hydrometeors, leading to unrealistic and "jumpy" bright bands. Reflectivity is also overestimated, if the particle sizes are comparable or larger than the radar wavelength.

For the other parameters in Table 7, the defaults are "reasonable" and with respect to melting particles, they lead to an "intermediately strong" bright band. To attain "stronger" or "weaker" bright bands (the uncertainty range is 10 dB!), an experienced user might change the parameters for the EMA's based on the detailed informations and extensive figures given in Blahak (2016).

Note that the option `itype_refl = 2` which is described in Section 6.3 of Blahak (2016), has been eliminated from the code in the meantime. It was similar to option 1 with respect to the EMA's for melting particles but replaced the exact Mie-backscattering coefficients by the "D^6" Rayleigh-approximation. This saved computing time with respect to the original backscattering coefficient calculation of single particles, but still required numerical integration over the size distributions to compute the reflectivity. With the advent of the Mie-lookup tables, this option did no longer provide any substantial computational advantage and at the same time its application was restricted to particles small compared to the wavelength and neglected attenuation.

2.5 Enhancement of traditional grid point output of unattenuated reflectivity (COSMO only)

With the coupling of EMVORADO to COSMO, the traditional output of (unattenuated) reflectivity on the model grid (`/GRIBOUT/` namelists), namely the fields `DBZ` (`yvarml`, `yvarpl`, `yvarzl`), `DBZ_850` (`yvarml`), `DBZ_CMAX` (`yvarml`) and `DBZ_CTMAX` (`yvarml`), can now be computed optionally by the new methods of Mie-scattering and Rayleigh-Oguchi-approximation as in step 1 of EMVORADO. The computation method for these values can be configured separately in the `/GRIBOUT/` namelist(s) by a set of new namelist parameters. The previous method from `pp_utilities.f90` is still available as an option. Internally, these grid point values are independent of the ones of EMVORADO step 1 which go into it's step 2, because of independent calls to the respective functions. But again, if the same reflectivity computation settings are chosen here as in step 1 (wavelength and other parameters), efficiency is gained by re-using reflectivity values from the last subroutine call instead of new computations.

For ICON, a similar mechanism for grid point reflectivity is planned, but the namelist- and namelist parameter names will be different.

2.6 Lookup tables for Mie-scattering option

Concerning the Mie-option for reflectivity, considerable speedup can be gained by using lookup tables (one table for each model hydrometeor category plus melting hydrometeors). This option is strongly recommended and can be chosen by namelist switches, both for step 1 computations in EMVORADO and for grib output of grid point reflectivities. It leads to about the same runtime as the Rayleigh-approximations and enables to take attenuation into account for the volume scans in EMVORADO. There is an automated mechanism to handle the generation of lookup-tables (the full Mie-codes are part of EMVORADO), storage in fortran binary files and re-using existing tables. Once the option is chosen, things happen automatically and thread-safe.

The names of the lookup table files are unique and consist of the hydrometeor type names and a "magic number" (10 digit signed integer) which is a hash-value representing the exact configuration parameters of the reflectivity calculation. An example for dry graupel is

"`radar_mietab_lm_drygraupel_-1582690394.bin`".

The user can specify the directories where to write and read these files via namelist (`ydir_mielookup_read` and `ydir_mielookup_write`). At each model start it is checked which different reflectivity configurations (wavelength, details of melting hydrometeors, etc.) are needed among all radar stations and grid point output streams for this model run, and based on the respective hash values the respective files are searched in the given read-directory. If the file is found, it is read from disk and re-used. If it does not exist, the respective table is computed and the file is created in the write-directory. Ideally the read- and write-directories should be defined equal and non-temporary, in which case no manual interaction (copying table files around) is necessary.

If the read- and write-directories are the same and are permanent, the user does not have to care about. This should be the preferred way for the "normal" user. However, in some operational environments it is desired for technical reasons to limit the direct model output exclusively to temporary directories and move it afterwards to where it should be stored permanently. This requires the possibility to read lookup tables from a different directory than where they are written to, but needs manual copying of newly created tables to the read-directory.

2.7 Output of radar composites

2D composites of simulated and observed reflectivities (useful e.g. for spatial model precipitation verification in dBZ-space) on a rotated lat-lon-grid are available for output through EMVORADO. These composites are based on volume scans and are computed in the exact same way for observation and simulation. Of course the observation composite is only available if observations are actually used in the simulation, which is not necessarily the case. All radar geometric effects (cone of silence, asymetric data distribution in space, etc.) are thus taken into account, enabling a fair comparison of model results and observations. However, radar reflectivity is a different moment of the hydrometeor size distribution than precipitation, so that results need not necessarily be consistent to, e.g., a surface-station-based precipitation verification. Composites are based on single elevations of each radar station and take on the maximum values in areas of radar overlap. Several composites for different elevations can be computed during one model run. DWD precipitation scans are possible as well in simulations, observations and composites. A further option is to take the maximum of all elevations and overlaps at each gridpoint, resulting in a kind of "MAX-CAPPI" but only the part with the "view from the top".

The 2D composite grid (rotated lat-lon-grid) can be arbitrarily chosen and an own grib-output method is provided by EMVORADO itself, based on special grib sample files provided through EMVORADO. This output can be activated by namelist switch `lcomposite_output = .TRUE.` in `/RADARSIM_PARAMS/`, provided that composites are actually computed by choosing `ldo_composite = .TRUE.`, `nel_composites` $>$ `0` and `eleindlist_for_composites` defined appropriately in `/RADARSIM_PARAMS/`.

In COSMO, the default settings for the composite grid are equal to the model grid, so that the COSMO grib output facilities (`/GRIBOUT/` namelists, parameter `yvarml`, shortnames `DBZCMP_SIM`, `DBZCMP_OBS`) might be used to write the composites to the model output files. However, this only works if no asynchroneous radar IO is done (`nprocio_radar = 0`), because otherwise an involved MPI-communication would destroy the entire advantage of asynchroneous radar IO. Therefore this is not recommended. Instead, one should rely on the extra grib files produced by EMVORADO with `lcomposite_output = .TRUE.`, cf. Section 6.1 below.

3 Compilation aspects

The source code of EMVORADO consists of a collection of independent, not model-specific, Fortran2003 modules ("core") and model-specific interfaces that are part of the NWP models. Currently there are implementations of the core in COSMO and ICON.

The EMVORADO core itself is hosted on a git repository at the German Climate Computing Center (DKRZ) in Hamburg ("cosmo-emvorado-package"), and potential users are welcome to contact the author for access. The way the core is compiled and linked depends on the hosting model.

For COSMO, the current version of the core at the time of creation of the COSMO version is already part of the COSMO source code distribution. It might be updated at any time by a newer EMVORADO core version by simply replacing the core files in the COSMO source by newer files from the git and recompiling.

For ICON, only the ICON-specific interface files are part of the source code distribution. Here, the user needs access to the above git to obtain the EMVORADO core files and make them available to the ICON `configure` and `build` process (see below).

In order to compile the model with the EMVORADO interface and with the EMVORADO modules, the pre-processor flag `-DRADARFWO` (COSMO) respectively `-DHAVE_RADARFWO` (ICON) has to be set for compiling. Further, the following pre-processor flags are implemented in EMVORADO, mostly (but not always) to enable/disable the use of some external libraries, connected with certain operator features. Ideally they should all be enabled, but may require additional libraries:

- `-D__COSMO__`: This standard COSMO flag should be set for the COSMO-implementation and the offline-version of EMVORADO and enables COSMO specific things in EMVORADO code. No extra library is necessary. **Do not use it for ICON!**

- `-DGRIB_API`: Set this to enable output of reflectivity composites in grib2 format by the operator (`ldo_composite=.TRUE.` and associated sub-parameters in namelist `/RADARSIM_PARAMS/`). Requires the `grib_api` or `eccodes` libraries.

- `-DNETCDF`: Set this to enable input of radar observation files in NetCDF format ("cdfin"), output of simulated and observed radar files in "cdfin" format (`voldata_format='cdfin'` in namelist `/RADARSIM_PARAMS/`), and output of NetCDF feedback files for data assimilation. Output files in "cdfin" format are internally gzip-compressed, and this requires library `netcdff` from `netcdf-4`, not `netcdf-3`.

- `-DNUDGING`: Set this to enable the production of NetCDF feedback files. Some modules from the DACE-code are requried for this, but, e.g., for COSMO and ICON these are already contained in the source code distribution.

- `-DWITH_ZLIB`: Set this to enable a gzip-compression of the optional ASCII volume data file output (Section 6.1) using `voldata_format='ascii-gzip'` in namelist `/RADARSIM_PARAMS/`. This compresses the data before writing to disk and saves disk space. It is normally as fast as uncompressed ASCII output (`voldata_format='ascii'`), because the additional time for compression is compensated by faster writing of less data to disk.

- `-DHDF5_RADAR_INPUT`: Set this to enable input of radar observation data in ODIM-HDF5 format. Currently the formats of MeteoSwiss and ARPA-SIMC are implemented. Requires the `hdf5_fortran` and `hdf5hl_fortran` libraries associated with `netcdf-4`.

For COSMO, the preprocessor flags and appropriate additional libraries for linking have to be registered by hand in the `Fopts` configuration file, which is included in the `Makefile`.

For ICON, the `configure` script has been extended by a target `--with-emvorado=<core_dir>`, where `<core_dir>` is the directory where the EMVORADO core files are located. Here is an example for the ICON configure and compile process:

```
$> ./configure --with-fortran=gfortran
               --with-emvorado=/home/myuser/cosmo-emvorado-package/src_links_for_icon/
               --with-netcdf=/usr --with-hdf5=/usr --with-zlib=/usr --with-grib_api=/usr
$> make -j 10
```

By using this mechanism, all of the above preprocessor switches are automatically set in the resulting auto-generated `Makefile`, so that all the above additional libraries (netcdf-4, hdf5, grib_api/eccodes, zlib) are needed.

4 Modes of operation

This section briefly describes the three general operation modes of EMVORADO and how to configure them via namelist parameters. The corresponding `/RADARSIM_PARAMS/` namelist(s) is/are independent of the hosting model. A general description of `/RADARSIM_PARAMS/` will be given below in Sections 5.1.2 to 5.1.4.

4.1 Idealized mode

Purey synthetic radars are simualted within an idealized model simulation (COSMO: Blahak, 2015). The radar station metadata (geographic location, wavelength, scan strategy, etc.) can be defined via namelist parameters in `/RADARSIM_PARAMS/`. One can have up to 50 radar stations in one model run.

For the COSMO world, an example is given in the exemplary runscript `run_ideal_radvop` in the `RUNSCRIPTS` folder of the COSMO distribution, which is a Weisman-Klemp-type "warm-bubble" initialization of a squall-line system, sampled by 4 radar stations. Of course one has to adapt this script to the specific computing platform.

It is suggest to go through the commented namelist `/RADARSIM_PARAMS/` to get a first idea on how to run and configure EMVORADO, along with the namelist documentation in Sections 5.1.2 to 5.1.4.

4.2 Real mode without using observation files

A "normal" real-case model forecast is driven by some external initial and boundary data and up to 50 synthetic radars are simulated, whose metadata are again fully defined via namelist parameters in `/RADARSIM_PARAMS/`.

For the COSMO world, an example is given in the runscript `run_cosmo_de_radvop_noobs` (`RUNSCRIPTS` folder of the COSMO distribution), a COSMO-DE run sampled by the 17 radar stations of the German network. Of course it has to be adapted to the user's specific computer platform.

It is suggest to go through the commented namelist `/RADARSIM_PARAMS/` to get a first idea on how to run and configure EMVORADO, along with the namelist documentation in Sections 5.1.2 to 5.1.4.

4.3 Real mode with observation files and creation of LETKF feedback files

A "normal" real-case model forecast as in the previous section uses real observational data files of up to 50 radar stations (directory `ydirradarin`, namelist parameter in `/RADARSIM_PARAMS/`) to

- define (some of) the station metadata for the radar simulation,
- write NetCDF feedback files for KENDA data assimilation (needs pre-processor flags `-DNUDGING` and `-DNETCDF`),
- produce radar composites from observations and simulations for model verification, or
- output volume scan data of the observations in the exact same format as the simulated volume scans for easier postprocessing (for output in CDFIN-format, pre-processor flag `-DNETCDF` is needed and a netcdf4-library; `-DWITH_ZLIB` is necessary for compressed ASCII output).

For the COSMO world, an example is given in the `RUNSCRIPTS/run_cosmo_de_radvop_obs` (`RUNSCRIPTS` folder of the COSMO distribution), which is a COSMO-DE run ingesting observation files and producing NetCDF feedback files and radar composites. Of course this has to be adapted to the user's specific computing platform.

It is suggest to go through the commented namelist `/RADARSIM_PARAMS/` to get a first idea on how to run and configure EMVORADO, along with the namelist documentation in Sections 5.1.2 to 5.1.4.

Concerning the format of the observation data, the following file types are currently supported in EMVORADO:

- **NetCDF files from DWD radars** which have been converted from original DWD BUFR using the tool `readbufrx2netcdf`. **At DWD, this format is internally called "CDFIN".** It contains all elevations of one radar parameter and at least one observation time per file (multi-volume single-parameter). It is allowed to have more than one observation time per file (although single time files are allowed, too), and the time range may be longer and may start earlier than the model forecast time. CDFIN-files are expected to follow the naming convention:

  `cdfin_<datasetname>_id-XXXXXX_<starttime>_<endtime>_<scantype>`

  - `<datasetname>`: “vr” (radial wind, can also be “vrsim” from a nature run in OSSEs), “qv” / “qvobs” (quality flags for radial wind), “z” / “zrsim” (reflectivity), or “qz” / “qzobs” (quality flags for reflectivity).
  - `XXXXXX`: the 6-digit WMO station ID, e.g., “01038”
  - `starttime`: the start of the time range contained in the file, format `YYYYMMDDHHmmss`
  - `endtime`: the end of the time range contained in the file, format `YYYYMMDDHHmmss`; can be equal to `starttime`
  - `scantype`: keyword for the general scan type, either “volscan” or “precipscan”

  Examples:

  - `cdfin_zr_id-010950_201307281200_201307281255_volscan`
  - `cdfin_zrsim_id-010950_201307281410_201307281435_volscan`
  - `cdfin_vr_id-010950_201307281200_201307281455_volscan`
  - `cdfin_vrsim_id-010950_201307281230_201307281230_volscan`
  - `cdfin_zr_id-010950_201307281200_201307281255_precipscan`

  Some of the station metadata are overtaken from the NetCDF attributes, but most stem from an internal background metadata list in the code (see below). For this, the compilation needs the pre-processor flag `-DNETCDF`. If you are on DWDs HPC environment, you can use the script `get_radbufr_data.sky` (available from the author) to retrieve such files from the DWD “Cirrus” data base. Supported are DWD’s PPI-scans as well as the socalled single-sweep horizon-following “precipitation scans”.

- **ODIM HDF5 files from MeteoSwiss.** These files contain one sweep (elevation) of one parameter of one observation time per file (single-sweep single-parameter) and are expected to follow the naming convention:

  `<M|P>L<stationletter><datetime>XX.<elevation>.<datasetidentifier>.h5`

  - `<stationletter>`: One of “A”, “D”, “L”, “P”, “W” (Albis, La Dole, Monte Lema, Plaine Morte, Weissfluhjoch)
  - `XX`: two “arbitrary” characters/digits (have some internal meaning at MeteoSwiss)
  - `elevation`: 3 digits for the elevation index in the volume scan, e.g. “002”. Normally a Swiss volume scan has 20 elevations, but in principle this is flexible in EMVORADO
  - `datetime`: nominal time (end-time) of the volume scan to which the file belongs, 9 digits in the format `YYJJJhhmm`, where `YY` is the 2-digit year and `JJJ` is the Julian day number and `hhmm` are hour and minute, respectively.
  - `datasetidentifier`: “V” for radial wind, “Z” for reflectivity.

  for example,

  - `MLL1807115250U.016.Z.h5`
  - `MLL1807115250U.012.V.h5`
  - `MLA1807115250U.001.Z.h5`
  - `PLD1331409457U.020.V.h5`

  Note that here the observation time can only be retrieved from the filename, not the date/time given in the HDF5 attributes, because the latter is the end-time of the sweep, not the volume scan. For this, the compilation needs the flag `-DHDF5_RADAR_INPUT` and the HDF5 libraries mentioned in Section 3.

- **ODIM HDF5 files from ARPA-Piemonte and ARPA-SIMC (Italy).** These files contain all elevations of one parameter from one observation time per file (single-volume single-parameter) and are expected to follow the naming convention `odim_<datetime>_<station-index>`

  - `datetime`: nominal time (end-time) of the volume scan, 12 digits in the format `YYYYMMDDhhmm`

  - `station-index`: 2-digit index of the station, e.g. `01`, `02` (the station-id is determined from an attribute in the file)

for example,

  - `odim_201410090010_01`
  - `odim_201410090025_02`

For this, the compilation needs the flag `-DHDF5_RADAR_INPUT` and the HDF5 libraries mentioned in Section 3.

The ODIM HDF5 standard leaves some room for the radar data providers to organize their data in detail, e.g., there is freedom on how to distribute the single elevations of volume scans and the observed parameters of a certain observation time among different files. Therefore each data provider requires an own internal data reader despite ODIM HDF5 standardization.

Many different kinds of metadata for each station are required in the code to be able to simulate the volume scan measurements, e.g., the nominal elevation angles of each PPI scan (scan strategy). Because these informations might not necessarily be available from the observation files, EMVORADO uses a background metadata list in the code to have a full set of metadata for each "known" radar station. Stations are identified by their WMO-ID (given in the data files), and data files and metadata sets are matched accordingly. This means that EMVORADO can only work with observations from radar stations which are part of this background list in the code. Currently, the radars from DWD, MeteoSwiss and ARPA-Piemonte are included. For other stations, respective entries in the background list are necessary in the code.

All observation files have to be collected (or linked) into a single directory, which has to be passed to EMVORADO by the namelist parameter `ydirradarin` in `/RADARSIM_PARAMS/`. For different model domains, `ydirradarin` can be different.

5 Namelist parameters

The namelist parameters related to EMVORADO can be divided into two logical streams.

The first stream concerns the "true" EMVORADO volume scan output stream (steps 1 and 2 of the reflectivity- and radial wind operators) and is given in the namelist `/RADARSIM_PARAMS/` in file `INPUT_RADARSIM` respectively `NAMELIST_EMVORADO`. Associated with this is also the production of feedback files for radar data assimilation and the production of simulated and observed radar composites. "True" observation data might come into play here.

The second stream is the "traditional" output of unattenuated reflectivity on the model grid, which is independent of any "true" observations, but whose computation now optionally may use the same fortran procedures than step 1 of EMVORADO (e.g., the more accurate Mie-scattering option). Again, this output is independent of step 1 of EMVORADO and can be done also if EMVORADO itself (steps 1 and 2) is switched off.

5.1 Namelist parameters to control steps 1 and 2

There is a dedicated namelist `/RADARSIM_PARAMS/` to control steps 1 and 2, which has to be found in the file `INPUT_RADARSIM` / `NAMELIST_EMVORADO`. If the hosting model has the capability for online-nesting, EMVORADO has the capability to be applied independently on each of the model domains. This is the case for ICON (Section 2.2. Here, each model domain needs its own `/RADARSIM_PARAMS/` namelist in the file `INPUT_RADARSIM` / `NAMELIST_EMVORADO`. To indicate which `/RADARSIM_PARAMS/` namelist is for which model domain, there is a mandatory namelist parameter `dom=<idom_model>` (integer), which has to contain the domain number in the hosting model starting from 1.

For COSMO, a model without such capabilities, `dom` has to be set to 1 always.

To indicate for which domain(s) EMVORADO should be applied ("radar-active" domains) and if asynchroneous radar IO should be performed, there are two parameters in top-level namelists of the hosting model, which are described in Section 5.1.1.

As mentioned above, each radar-active model domain needs its own `/RADARSIM_PARAMS/` namelist, identified by its own mandatory `dom` parameter. The parameters of this namelist are documented in these 3 tables:

- Table 4 in Section 5.1.2 for global parameters applied either generally or to all radar stations,
- Table 5 in Section 5.1.3 for the contents of the derived type `radar_meta_type` (detailed metadata of a single radar station), where station i is represented by the list element ("parameter block") `rs_meta(`i`)` of this type.
- Table 6 in Section 5.1.4 for the contents of the derived type `dbzcalc_params` (configuration parameters for the reflectivity computation), which can be indifidually different for each radar station. Again, station i is represented by the list element ("parameter block") `dbz_meta(`i`)` of this type.

Some of the namelist parameters in these tables and internal fields are vectors or arrays with maximum allowed sizes for (some of) their dimensions. These upper limits for the dimensions are deklared in module `data_radar.f90` and can be adapted by the user as needed. The list of parameters, their current settings and explanations is as follows:

`nel_max`	25	max. number of elevations of any station
`nscanstrategies_max`	10	max. number of different nominal scan strategies of any station
`nobstimes_max`	500	max. number of observation times of any station during model forecast
`ngpsm_max`	15	max. number of vertical/horizontal nodes for the Gauss-Legendre quadrature applied for beam function smoothing
`nradsta_max`	50	max. number of radar stations; two different scan strategies for the same station count as two different radar stations!
`ncountry_max`	4	max. number of "countries" (set of defaults for groups of radars with similar properties)
`ndatakind`	4	max. number of different radar observables/moments for input*

`nel_composite_max`	10	max. number of different composites (from different radar elevations) produced during one run
`noutput_fields_max`	25	max. number of different radar fields/quantities/moments for volume data output
`ndoms_max`	5	max. number of model domains supported by EMVORADO

*About `ndatakind`: This is the maximum number of different data fields contained in the observation input files. Currently, only DWD NetCDF radar files are supported, and here we currently only use reflectivity Z_e, radial wind v_r and their respective quality flags q_z and q_v. Therefore this number has been set to 4.

5.1.1 Additional parameters in the hosting model

In the hosting model itself, there are the (domain dependent) master switch `luse_radarfwo` in some top-level namelist (for COSMO: `/RUNCTL/`; for ICON: `/run_nml/`) and the number of additional PEs for asynchroneous radar IO (COCMO: `nprocio_radar` in `/RUNCTL/`; ICON: `num_io_procs_radar` in `/parallel_nml/`). `luse_radarfwo` switches on steps 1 and 2 of EMVORADO. It has no influence on the "traditional" grid point output described in Section 5.2.

Table 3: Additional parameters in some top-level namelist (COSMO: `/RUNCTL/`).

Name	**Type**	**Definition / Purpose / Comments**	**Default**
`luse_radarfwo`	`LOG (ndoms)`	Global switch to include/exclude the computation and output of volume scan data and reflectivity composites (step 2 of EMVORADO). For ICON, the parameter is a vector of switches for each corresponding ICON model domain. For COSMO it is a scalar. From EMVORADO side, there can be at most `ndoms_max` radar-active domains. If more domains are set `.TRUE.` in ICON, the run will stop with an error message. The user may increase `ndoms_max` in the source code (`radar_data.f90`) and recompile. Note: the "traditional" grid point reflectivity output via `/GRIBOUT/`-namelists in the hosting model is independent of this switch.	`.FALSE.`
`nprocio_radar` (COSMO) `num_io_procs_radar` (ICON)	`INT (1)`	Number of additional PEs for asynchroneous radar IO (total sum for all radar-active domains). Note: the "normal" asynchroneous grib IO of the hosting model is independent of it and can be used together.	0

5.1.2 Global namelist parameters in `/RADARSIM_PARAMS/`

Table 4: Global namelist parameters in `/RADARSIM_PARAMS/`. Kind abbreviations: "I" = INTEGER, "R" = REAL/DOUBLE, "C" =CHARACTER, "L" = LOGICAL, "T" = Derived TYPE.

Name	Kind (Dim.)	Description / Remarks	Default
dom	I (1)	Domain number in the hosting model (starting from 1) for which radar data are to be simulated using the settings of the respective `/RADARSIM_PARAMS/` namelist. Mandatory parameter. For COSMO (only one domain) the value 1 has to be given. For ICON, there has to be one `/RADARSIM_PARAMS/` namelist for each radar-active domain.	Mandatory, no default
nradsta_namelist	I (1)	Number of radar stations whose metadata are defined respectively altered via this namelist. The meaning of this number differs for runs with and without using observational data: • For `lreadmeta_from_netcdf=.TRUE.` (using obs data): Number of radar stations to take into account from the list of `rs_meta(`i`)` and `dbz_meta(`i`)` blocks of the namelist (see below): For each station from `i=1...nradsta_namelist`, you can override some metadata in the structures `rs_meta(`i`)` and `dbz_meta(`i`)`, which otherwise are taken from the obs files found in the input directory `ydirradarin` (a few metadata only) or in the background list (most of the metadata). The number of simulated radar stations is determined automatically from the number of files found in the input directory. `nradsta_namelist` is NOT the number of radar stations found in the input directory but rather the number of stations where you want to change some of the metadata! If you define more change-blocks than `nradsta_namelist`, only the first `nradsta_namelist` blocks will be taken into account. **Important:** for a correct match of the below change-blocks with the radar stations from the input files, give the correct WMO `rs_meta(`i`)%station_id` and `rs_meta(`i`)%scanname` for each block below. See also Section 5.1.5. • For `lreadmeta_from_netcdf=.FALSE.` (no obs data used): Number of radar stations to simulate. For each station `i=1...nradsta_namelist`, you can define one structure block `rs_meta(`i`)` and `dbz_meta(`i`)` in the namelist. If you define more blocks than `nradsta_namelist`, only the first `nradsta_namelist` stations will be simulated. **Important:** give a unique station id to each station, which is an integer number > 0 with at most 6 digits. In case of "really existing" radars, use e.g. the WMO station ID.	1

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
icountry	L (1)	Global background value for the country flag. It influences the default for all radar station metadata (scan strategy, beam width, etc.), which in turn act as a background value (effective if not explicitly changed by namelist). This applies to all modes of operation (idealized, real case without observation files, real case with observation files). Currently implemented are: 1 = Germany (DWD) 2 = Switzerland (MeteoSwiss) 3 = Italy (ARPA-SIMC) The actual code for these defaults can be found in the subroutines `get_meta_proto_dwd()`, `get_meta_proto_swiss()` and `get_meta_proto_italy()` in module `radar_obs_meta_list.f90`.	1
rs_meta	T (nradsta_max)	Derived type to hold all the metadata information of one specific radar station. The namelist parameter is a vector of this type, one element / block per station. Parameters for a certain radar have to be specified in derived type notation, e.g. `rs_meta(5)%station_id` for the station id of the 5'th radar. Detailed explanation of the type components can be found in a separate table below. Note that all the components of the type can be specified in the namelist, but not all of them really take effect. Depending on the mode of simulation (see previous section), this can be used to either specify directly all the metadata of the simulated stations (idealized mode and real case mode without observational data), or to alter some of the metadata which have been read from observational files (real case mode with using observational data). Most of the default values depend on the choice of `icountry`. Currently, either values typical for German radars (`icountry=1`) or Swiss radars (`icountry=2`) can be chosen. The default radar position(s) is/are in the center of the model domain. The components of an element of `rs_meta` will be explained in Table 5below.	Depends on `icountry`, see below in Table 5
dbz_meta	T (nradsta_max)	Derived type to hold all the metadata information of the grid point reflectivity calculation for a specific radar station. The namelist parameter is a vector of this type, one element / block per station. Parameters for a certain radar have to be specified in derived type notation, e.g. `dbz_meta(3)%itype_refl` for the type of reflectivity computation of the 3'rd radar. The type components are described shortly in a separate table below and an extensive documentation can be found in Blahak (2016). Its usage in the namelist is similar to that of `rs_meta` above. Most component default values are hardcoded in the source code and are stored in the variable `dbz_namlst_d` of the same derived type. The components of an element of `dbz_meta` will be explained in Table 6 below. One type component (radar wavelenght) will however be automatically overwritten by the value given in the `rs_meta`-structure for each radar station.	see below in Table 6
ldebug_radsim	L (1)	Switch to enable extensive debug messages to stdout. This concerns the **real forward operator** (computation of the polar synthetic radar data) **and not the `DBZ` grid point output**, which is triggered through the `GRIBOUT` namelist(s). There, the debug mode is triggered by setting `ldebug_io=.TRUE.` in `IOCTL`.	.FALSE.

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
lout_geom	L (1)	If `lvoldata_output=.TRUE.`: enable output of geometric informations for each radar bin into volume data files. Filenames contain keywords to identify the following parameters: `'losim'` = simulated geographic longitude [°] `'lasim'` = simulated geographic latitude [°] `'hrsim'` = simulated height of radar bins [m MSL] `'ersim'` = simulated local beam elevation angle [°] `'adsim'` = simulated arc distance from radar site (great circle distance) [m]	.FALSE.
ltestpattern_hydrometeors	L (1)	Switch to enforce an artificial testpattern of hydrometeors in the model domain for technical testing. Is used in the technical testsuite. Useful during development of reflectivity calculation methods for quick test runs with only one timestep.	.FALSE.
loutdbz	L (1)	Master switch to enable simulation and output of radar reflectivity. Also, observation data processing for reflectivity (e.g., for feedback files) is enabled/disabled according to this switch. **If** `lvoldata_output=.TRUE.`, the following volume data sets are available for output (`voldata_output_list`): `'zrsim'` = simulated reflectivity [dBZ]; -999.99=missing value, -99.99=correct 0 `'zrobs'` = observed reflectivity [dBZ] (only if `lreadmeta_from_netcdf=.TRUE.`) **If** `lreadmeta_from_netcdf=.TRUE.`, the following superobservations are available for output (`voldata_output_list`): `'zrsupsim'` = super-observations of simulated reflectivity [dBZ] `'zrsupobs'` = super-observations of observed radial reflectivity [dBZ]	.TRUE.
itype_refl_glob	I (1)	Background value for the type of reflectivity calculation for all radar stations (cf. Section 2.4): 1 = Mie (Blahak, 2016) 3 = Rayleigh-Oguchi (Blahak, 2016) 4 = “Old” Rayleigh from COSMO `pp_utilities.f90` Can be adjusted for each individual station by `dbz_meta(`i`)%itype_refl` (see Table 6).	3
lextdbz	L (1)	For `dbz_meta(`i`)%itype_refl=1`: Take into account extinction (attenuation) along the ray paths. Not possible for the Rayleigh options. If `lvoldata_output=.TRUE.`, the twoway-attenuation coefficients [db/km] are written to volume data files using the keyword “`epsim`” and the path integrated attenuation [dB] to files using “`etsim`”.	.FALSE.
llookup_mie	L (1)	Global background switch to enable the use of efficient lookup tables for Mie scattering. Can be adjusted for each individual station by `dbz_meta(`i`)%llookup_mie` (see Table 6). Only effective if `dbz(`i`)%itype_refl=1`.	`.TRUE.`

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
ydir_mielookup_read	C (250)	For `dbz_meta(`i`)%itype_refl=1` and `dbz_meta(`i`)%llookup_mie=.TRUE.`: directory for reading lookup table files. Different lookup table files are expected for the different hydrometeor types and for specific reflectivity computation configurations. If specific files are not present in the `ydir_mielookup_read`, EMVORADO automatically creates the tables and stores the files in `ydir_mielookup_write`, which takes some computation time. To save this time from run to run, `ydir_mielookup_read` should be equal to `ydir_mielookup_write` and should be a permanent directory.	' '
ydir_mielookup_write	C (250)	For `dbz_meta(`i`)%itype_refl=1` and `dbz_meta(`i`)%llookup_mie=.TRUE.`: directory for storing new Mie lookup tables. Should normally be equal to `ydir_mielookup_read`, but for some operational applications it is not permitted to write output to non-temporary directories, so that read- and write-directories have to be different. In this case, the newly created lookup table files have to be moved to `ydir_mielookup_read` by hand or by operational scripts.	' '
loutradwind	L (1)	Master switch to enable simulation and output of radial wind. Also, observation data processing for radial wind (e.g., for feedback files) is enabled/disabled according to this switch. **If** `lvoldata_output=.TRUE.`, the following volume data sets are available for output (`voldata_output_list`): `'vrsim'` = simulated radial wind [m/s]; -999.99=missing value (only if `loutradwind=.TRUE.`) `'vrobs'` = observed radial wind [m/s]. Dealiasing depends on namelist switch `ldealiase_vr_obs` (only if `lreadmeta_from_netcdf=.TRUE.`) `'vrobserr'` = reflectivity dependent observation error for radial wind (only if `lreadmeta_from_netcdf=.TRUE.`) **If** `lreadmeta_from_netcdf=.TRUE.`, the following superobservations are available for output (`voldata_output_list`): `'vrsupsim'` = super-observations of simulated radial wind [m/s `'vrsupobs'` = super-observations of observed radial wind [m/s] `'vrsubobserr'` = reflectivity dependent observation error for superobe'd radial wind (only if `lreadmeta_from_netcdf=.TRUE.` and `itype_obserr_vr`>0) `'vasim'` = area-wide simulated radial wind field. Does not take into account reflectivity weighting and hydrometeor fallspeed. Internally used as a proxy for dealiasing the observations [m/s] (only if `ldealiase_vr_obs=.TRUE.`)	.TRUE.

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
lweightdbz	L (1)	Take into account reflectivity weighting in case of: • `lfall=.TRUE.`: reflectivity weighted fallspeed of hydrometeors instead of average fallspeed • `lsmooth=.TRUE.`: reflectivity weighted volume averaged radial wind instead of non-weighted average. Side-effect: radial winds for bins with a too small reflectivity (range-dependent detection threshold if `lmds_vr=.TRUE.` or -90 dB if `.FALSE.` are set to missing values -999.99.	.FALSE.
lfall	L (1)	Take into account the fallspeed of hydrometeors in. radial wind simulations. Reflectivity weighting is enabled by `lweightdbz=.TRUE.`, which takes into account the Rayleigh-Oguchi-Approximation, regardless of `itype_refl`	.FALSE.
ldealiase_vr_obs	L (1)	If `lreadmeta_from_netcdf=.TRUE.`, dealiase the observed radial winds bin-wise based on a simulated radial wind estimate from plain model u, v, w (`'vasim'`). This estimate is present at all locations which are not blocked by the orography. The algorithm behind this is very simple, just determine the nearest Nyquist interval from the difference to the a-priori estimate. Has problems if the unambiguous range (Nyquist interval) is very small and multiple velocity foldings occur frequently.	.TRUE.
lfill_vr_backgroundwind	L (1)	Fill "missing" values for simulated radial winds caused by `lsmooth=.TRUE.` or `lmds_vr=.TRUE.` by the same simulated plain radial wind estimate `'vasim'` as for `ldealiase_vr_obs=.TRUE.`. Note that missing data caused by bins blocked by the model orography. This is useful for data assimilation applications where each observed radial wind is expected to have a valid model equivalent.	.FALSE.
lonline	L (1)	Option for "online" beam propagation. If .TRUE., enables actual ray tracing / beam bending computations based on the actual simulated air refractive index field. The sub-switch `lsode` provides the choice of two different ray tracing methods. If .FALSE., the much simpler and more efficient climatological "4/3-earth" model is used. The online-option is more expensive. All options are documented in Zeng et al. (2014)	.FALSE.
lsode	L (1)	If `lonline=.TRUE.`: choice of the "online" beam propagation algorithm: .FALSE. = method TORE from Zeng et al. (2014), based on Snell's law for spherically stratified media including effects of total reflection. .TRUE. = method SODE from Zeng et al. (2014), based on the second-order ordinary differential equation for the beam height as function of range. Both methods are equally accurate and of similar efficiency. SODE is believed to be more robust.	.TRUE.
lcomm_nonblocking_online	L (1)	If `lonline=.TRUE.`: enable a non-blocking communication instead of an `MPI_ALLTOALLV`. May result in considerable performance gains in the additional communication step mentioned in Section 2.3. **However, this depends on the computing platform and ultimately has to be tested by the user.**	.TRUE.

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
lsmooth	L (1)	Master switch to take into account that radar returns are a beam function weighted volume average ("pulse volume"). If .FALSE. the simulated radar returns are represented by their value at the center point of the pulse volume only ("pencil beam" approximation). If .TRUE. a 2D Gauss-Legendre quadrature using a variable number of integration nodes with respect to an azimuth-elevation-sector perpendicular to the beam direction around the center point of the pulse volume is performed, within which about 90 % of the transmitted energy is confined. Up to now, range weighting is not taken into account. The effective Gaussian beam weighting function of an azimutally scanning radar with axisymetric beam pattern (Blahak, 2008) is applied for weihting. The additional computational costs and main memory requirements of this option scale with the product of the chosen numbers of integration nodes in both angular directions. Because hydrometeors vary predominantly in the vertical, elevational smoothing is considered more important than azimutal smoothing. Notable effects are expected mainly for radial wind. Another aspect is the simulation of (partial) beam blocking caused by the (model) orography, but always in light of the fact that the model orography is smoother than the true orography. All in all, Zeng et al. (2016) suggest that at most 9 points in the vertical and 3 points in the horizontal are sufficient to give very accurate results.	.FALSE.
ngpsm_h_glob	I (1)	Number of integration nodes for Gauss-Legendre quadrature in the horizontal (azimuthal) direction. The given number should be odd so that the central point is among the nodes. Normally, at most 1-3 points are sufficient.	-99
ngpsm_v_glob	I (1)	Number of integration nodes for Gauss-Legendre quadrature in the vertical (elevation) direction. The given number should be odd so that the central point is among the nodes. Normally, at most 7-9 points are sufficient.	-99
lmds_z	L (1)	Take into account the limited sensitivity of the radar receiver for the simulated reflectivity. If .TRUE. simulated reflectivities below a range dependent threshold are set to -99.99 dBZ ("correct zero"). The threshold is defined by the inverse square-dependence of the returned energy on range and a reference minimal detectable signal at a reference range (Zeng et al., 2016). Both parameters are part of the radar metadata list for each station (`rs_meta(`*i*`)%mds_Z0` and `rs_meta(`*i*`)%mds_r0`) and meaningful defaults for each station are defined in the code, depending on the country and the WMO station ID (if `lreadmeta_from_netcdf=.TRUE.`).	.FALSE.
lmds_vr	L (1)	Take into account the limited sensitivity of the radar receiver for the simulated radial wind. If .TRUE. radial winds at bins with simulated reflectivities below a range dependent threshold are set to -999.99 m/s ("missing").	.FALSE.
ydirradarout	C (250)	Output directory for volume data, radar composites and feedback files which are generated by EMVORADO. Has to be unique for each different model domain (cf. `dom` parameter and Section 2.2). If empty, the standard model output directory is used, as defined in subroutine `get_model_inputdir()` in module `radar_interface.f90`.	' '

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
lcomm_nonblocking_online	L (1)	Enable a non-blocking communication method instead of a series of blocking `MPI_GATHER` to send radar data from the compute PE's to the radar-IO-PE's. May result in a considerable performance gain of this communication step.	.TRUE.
lvoldata_output	L (1)	Master switch to enable the output of radar volume data, cf. Section 6.1.1. If .TRUE., output files of the volume data are written to the output directory `ydirradarout`. By the different sub-parameters immediately below, the user can select the data format and which radar moments and variables to output. It is also possible to limit the output to certain observation times and certain elevations only.	.TRUE.
voldata_format	C (12)	Character string to indicate the desired output format for the volume data, cf. Section 6.1.1. Can be either `'ascii'`, `'ascii-gzip'`, `'cdfin'` or `'f90-binary'`. **The recommended format is `'cdfin'`.** Besides having one file per output time step, this format also enables to write a series of consecutive output time steps ("time batch") to a single output file per radar moment (setting `cdfin_tref` and `cdfin_dt` appropriately below). While the former limits the number of data files, the latter is required to ingest the data into DWD's POLARA software.	'ascii'
voldata_output_list	C (12) (noutput_fields_max)	List of character strings to indicate the desired volume data output quantities. Per default the list is empty, which indicates to output all available quantities depending on the EMVORADO configuration. The available quantities are described in Section 6.1.1 and depend on the configuration. For example, if `loutradwind=.FALSE.`, there will be no files for `'vrsim'`, `'vrobs'`, `'vrsupsim'`, `'vrsupobs'` or `'vasim'`, even if they have been explicitly listed in the `voldata_output_list`. The files for geographic and geometric informations on the radar bins are only produced if `lout_geom=.TRUE.`.	' '
ind_ele_voldata_glob	I (nel_max)	For `lvoldata_output=.TRUE.`: List of indices of elevations which are written the into volume data files. This enables to write only some of the elevations to to the files to save disk space. Global background list for all radar stations, which can be adjusted for each individual station by `rs_meta(`i`)%ind_ele_voldata` (see Table 6). After namelist reading, the given list will be filtered for each radar station. All invalid indices (< 0 or > rs_meta(i)%nel) will silently be eliminated and only the valid indices will be used. However, if all values are -999, all elevations are used.	-999
obs_times_voldata_glob	R (nobstimes_max)	List of times for which volume data output is desired [s since model start]. Global background list for all stations, may be refined for single stations by `rs_meta(`i`)%obs_times_voldata`. This is useful to tailor the amount of volume data output to the specific user needs, independent from the feedback file output. Takes precedence over `dt_obs_voldata_glob`. If existing `rs_meta(`i`)%obs_times` do not match, no output for station i for these times. Default: missing value to indicate that `dt_obs_voldata_glob` should be evaluated instead.	-999.9

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
dt_obs_voldata_glob	R (1)	Time increment for desired volume data output [s]. Active only if `obs_times_voldata_glob(:)=-999.99`. Is used to build the list of desired output times in equal steps starting at model start time. Default: missing value to indicate that all existing `rs_meta(`i`)%obs_times` should be output.	-999.9
cdfin_tref	R (1)	For 'cdfin' output files: reference time [s] for time batches. `cdfin_tref=` 0.0 means that batches are synchronized to the model start time	0.0
cdfin_dt	R (1)	For 'cdfin' output files: time increment [s] of time batches. The default is a very large time in seconds to indicate that a cdfin-file should contain the whole forecast time span. **Setting it to 0.0 means that a separate file for each time step and radar moment is created, which is mandatory to ingest the data into DWD's POLARA software.**	1E6
lreadmeta_from_netcdf	L (1)	Master switch to trigger the use of real radar observation files. Currently this is implemented for the German, Swiss and Italian radar networks. **Setting the switch to .TRUE. is the pre-requisite for producing any observation- related output: NetCDF feedback files, observed composites, bubble generator.**	.FALSE.
ydirradarin	C (250)	Input directory for observation data files. Can be equal or different for different model domains. If empty, the standard model input directory is used, as defined in subroutine `get_model_inputdir()` in module `radar_interface.f90`.	' '
ydirlistfile	C (250)	Text file (name and path) containing a directory listing of the input directory given in `ydirradarin`. The first line of this file should contain the number of listed files, followed by the file names (without paths), one file per text line. If an empty name is given, the directory listing is automatically created via a Fortran `call system()` expecting a UNIX/Linux operating system. "Normal" users should leave this parameter blank. It might be useful for operational applications to avoid system-calls for some reasons, but requires the pre-generation of this file by hand or shell.	' '
lqc_flag	L (1)	Enable the use of quality flags to filter observations with bad quality. If .TRUE., EMVORADO expects to find input data files for quality flags in the input directory. So far, only quality flags from an old and deprecated quality control for DWD radar data are imlemented, and it is only checked whether the quality is good (flag=0) or bad (flag=1). The dataset names expected in the filenames are `'qv'` for radial wind and `'qz'` for reflectivity, cf. Section 4.3. **Since the availability of already quality-controlled radar data at DWD (2016) this switch should be set to .FALSE. and quality-controlled data should be directly used on input!**	.TRUE.
lcheck_inputrecords	L (1)	Deprecated! Has no function any more and will be removed in a future version of EMVORADO.	.FALSE.

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
lequal_azi_alldatasets	L (1)	Flag to indicate that observation files for the same station and observation time(s) but different radar moments (e.g, for DWD cdfin-files of MeteoSwiss hdf5-files) are believed with confidence to contain the exact same azimuts and elevations. Normally this should be the case, therefore its default is .TRUE. and a series of cross-checks among files for different radar moments are bypassed to save time. If .FALSE., a thorough check for equal azimuts and elevations is performed, and in case of error the respective time step of the respective station is completely ignored. An error message is output, but the EMVORADO and the model run do continue.	.TRUE.
lfdbk_output	L (1)	Trigger output of NetCDF feedback files. Only possible for `lreadmeta_from_netcdf=.TRUE.`.	.FALSE
ind_ele_fdbk_glob	I (nel_max)	For `lfdbk_output=.TRUE.`: List of indices of elevations which are written the into feedback files. This enables to write only some of the elevations to to the files to save disk space. Global background list for all radar stations, which can be adjusted for each individual station by `rs_meta(`i`)%ind_ele_fdbk` (see Table 6). After namelist reading, the given list will be filtered for each radar station. All invalid indices (< 0 or $>$ rs_meta(i)%nel) will silently be eliminated and only the valid indices will be used. However, if all values are -999, all elevations are used.	-999
obs_times_fdbk_glob	R (nobstimes_max)	List of times for which radar data should be written to the feedback files [s since model start]. Global background list for all stations, may be refined for single stations by `rs_meta(`i`)%obs_times_fdbk`. This is useful to tailor the amount of data to the specific needs of the data assimilation system, independent from the volume data output. Takes precedence over time specification via `dt_obs_fdbk_glob`. If existing `rs_meta(`i`)%obs_times` do not match, no output for station i for these times. Default: missing value to indicate that `dt_obs_fdbk_glob` should be evaluated instead.	-999.9
dt_obs_fdbk_glob	R (1)	Time increment for writing radar data to feedback files [s]. Active only if `obs_times_fdbk_glob(:)=-999.99`. Is used to build the list of desired output times in equal steps starting at model start time. Default: missing value to indicate that all existing `rs_meta(`i`)%obs_times` should be output.	-999.9
itype_supobing	I (1)	Type of computation of super-observations for `lreadmeta_from_netcdf=.TRUE.`, mainly intended for the NetCDF feedback files (fof): 0 = no superobing (but possible data thinning, see `thin_step_azi` and `thin_step_range` below) 1 = weighted averaging over a symetric range-azimuth-sector around a superobservation reference point within each PPI (range-azimuth-plane). Distande-to-center depended weights according to Cressman (1959) 2 = median over the same sectors (very slow!)	0
thin_step_azi	I (1)	If no superobing: azimutal step width (array index space) of data thinning for NetCDF feedback files.	1
thin_step_range	I (1)	If no superobing: range bin step width (array index space) of data thinning for NetCDF feedback files.	1
supob_cart_resolution	R (1)	Resolution [m] for the (near-)cartesian grid for super-observations.	20000.0

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
supob_ave_width	R (1)	Width of averaging area for superobing [m].	$\sqrt{2}\times$ supob_cart_resolution_d
supob_minrange_vr	R (1)	Min. range of superobing points for v_r [m].	$0.75\times$ supob_ave_width_d
supob_minrange_z	R (1)	Min. range of superobing points for Z [m].	$0.75\times$ supob_ave_width_d
supob_azi_maxsector_vr	R (1)	Maximal azimut sector (symetrical to its center) for v_r superobing [°].	40.0
supob_nrb	I (1)	Lower threshold for number of radar bins for computing valid superobservations	3
supob_vrw	R (1)	Upper threshold of allowed radial wind standard deviation within a superobservation area [m/s]. Superobservations with larger values are rejected for feedback files.	10.0
supob_rfl	R (1)	Upper threshold of allowed linear radar reflectivity standard deviation within a superobservation area [$\mathrm{mm}^6\,\mathrm{m}^{-3}$]. Superobservations with larger values are rejected for feedback files.	$\sqrt{\text{HUGE(1.0\_dp)}}$
supob_lowthresh_z_obs	R (1)	Before computing the superobservations from the original observed reflectivities, set values smaller than this threshold [dBZ] to this threshold. This influences the impact of no-reflectivity observations in the data assimilation. Only effective if `itype_supobing`>0.	-999.99
supob_lowthresh_z_sim	R (1)	Same for the simulated reflectivities [dBZ]. Only effective if `itype_supobing`>0.	-999.99

Table 4: continued

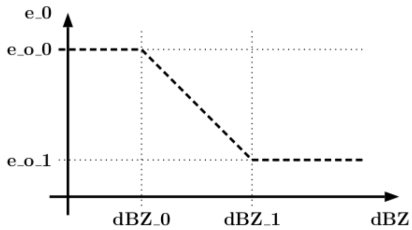
Name	Kind (Dim.)	Description / Remarks	Default
itype_obserr_vr	I (1)	Type of computation of the observation error for radial wind to be stored in the NetCDF feedback files (fof) for data assimilation, for `lreadmeta_from_netcdf=.TRUE.` and `loutradwind=.TRUE.`: 0 = constant value from namelist parameter `baseval_obserr_vr` (if 1.0, to be interpreted as relative error) 1 = define the observation error for radial wind as function of observed reflectivity in form of a linear ramp function as shown in the sketch below. In case of superobservations (`itype_supobing>0`), the reflectivity used for this purpose is also superobserved, but small reflectivities are not raised to `supob_lowthresh_z_obs`). The ramp function can be defined via namelist parameters described below the sketch. Needs `loutdbz=.TRUE.`! 2 = observation error is computed from observed reflectivity before superobbing, also as the below linear ramp. In case of `itype_supobing=1`, • its inverse is an additional weight in computing superobservations of radial wind, • the observation error is itself superobe'd with the same weighting as the radial wind. Needs `loutdbz=.TRUE.`!  The ramp can be definded by the following namelist parameters: dBZ_0 = `ramp_lowdbz_obserr_vr` e_o_0 = `maxval_obserr_vr` dBZ_1 = `ramp_highdbz_obserr_vr` e_o_1 = `baseval_obserr_vr` (if 1.0, then entire ramp to be interpreted as relative value) Note that for reflectivity the observation error is always set to 1.0 and should be interpreted as a relative value.	0
baseval_obserr_vr	R (1)	**If** `itype_obserr_vr=0`**:** general constant value for radial wind observation error **If** `itype_obserr_vr=1/2`**:** obs error for dBZ-values $\geq$ `ramp_highdbz_obserr_vr` (e_o_1 in above sketch). IF baseval_obserr_vr=1.0, the observation error for radial wind is a relative error, which, in the data assimilation software, may be multiplied with observation errors which come from other sources (e.g. Desroziers-statistics).	1.0
maxval_obserr_vr	R (1)	**If** `itype_obserr_vr=1/2`**:** obs error for dBZ-values $<$ `ramp_highdbz_obserr_vr` (e_o_1 in above sketch).	10.0
ramp_lowdbz_obserr_vr	R (1)	**If** `itype_obserr_vr=1/2`**:** lower ramp dBZ-threshold (dBZ_0 in above sketch) for increasing observation error in radial wind.	0.0
ramp_highdbz_obserr_vr	R (1)	**If** `itype_obserr_vr=1/2`**:** upper ramp dBZ-threshold (dBZ_1 in above sketch) for increasing observation error in radial wind.	10.0

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
itype_metric_refl_fdbk	I (1)	Option for the specific metric of reflectivity Z to write into feedback files (fof), for `lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.`: 1 = write ζ in dBZ to feedback files ($\zeta = 10\lg\frac{Z}{1\,\mathrm{mm}^6\mathrm{m}^{-3}}$) 2 = convert to effective LWC $= 0.004\,Z^{0.55}$ (g/m^3), but leave observation error unchanged from `itype_obserr_vr` 3 = convert to effective LWC as for (2) and write a relative observation error for this LWC to fof, such that, when multpilied by a certain ΔdBZ (one-sided standard dev.) in the LETKF, this factor reproduces the weight which the equivalent dBZ-observation would have in the LETKF assuming a constant reflectivity error ΔdBZ: $e_o = e_{o,lim} + 0.5\,a\,\frac{10^{0.1\,(\zeta+5dB)\,b} - 10^{0.1\,(\zeta-5dB)\,b}}{5dB}$ with $a = 0.004$, $b = 0.55$. The additional constant $e_{o,lim}$ is added to prevent overly small observation errors for small LWC. It is the asymptotic value for LWC $\rightarrow$ 0.0 and can be given by namelist parameter `minval_obserr_lwc`.	1
minval_obserr_lwc	R (1)	**If `itype_metric_refl_fdbk=3`:** asymptotic value for LWC $\rightarrow$ 0.0.	5E-4
labort_if_problems_obsfiles	L (1)	If .TRUE., abort the model run if serious problems with required observation files or meta data occur, i.e., no obs files at all, errors in file content, missing variables, missing or wrong station ID or scan strategy, etc. The default is to abort, but in operational runs this decision should be up to the user. If .FALSE., the model run continues but issues respective ERROR and WARNING messages.	.TRUE.
lcomposite_output	L (1)	Only effective if `ldo_composite=.TRUE.` or `ldo_bubbles=.TRUE.`: if `lcomposite_output=.TRUE.` EMVORADO uses it's own grib2-output facilities to write radar reflectivity composites to the output directory `ydirradarout`. See also Sections 2.7 and 5.1.6 for more informations. Needs the local DWD grib sample file `DWD_rotated_ll_7km_G_grib2` from DWD's `grib_api` or `eccodes` distribution (center=EDZW). Otherwise, the composite output relies on the hosting model's output facilities. In the COSMO-model this is via the `/GRIBOUT/` namelist using parameter `yvarml='DBZCMPSIM','DBZCMPOBS'`, but this only works correctly if the composite grid is defined equal to the COSMO-model grid **and is not recommended any more**. **For ICON, `lcomposite_output=.TRUE.` is the only possibility to output the composites.**	.FALSE.

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
ldo_composite	L (1)	If .TRUE. generate one or more radar composite(s) of reflectivity from the simulated and (if available) observed volume scans. A composite is generated by using one elevation of each station and, in areas of horizontal overlap, take the maximum of both (cf. Section 2.7). More than one composite using different elevations can be generated simultaneously. Composites are generated on a rotated lat-lon grid. In case of COSMO, it is equal to the model grid at the moment, but any other rotated lat-lon grid specifications would be possible in principle. The reason to choose the model grid has been the fact that in the past only the COSMO-internal grib-output facilities have been used to write the composites to disk. However, in the meantime there is an own grib2-writer in EMVORADO, which is able to code any arbitrary lat-lon grids. There will be new namelist parameters in the future to specify the composite grid parameters independently from the model grid.	.FALSE.
comp_meta%ni	I (1)	Rotated lat/lon grid for the reflectivity composites: number of grid points in rotated meridional ("East-West") direction.	COSMO = ie_tot ICON = 421
comp_meta%nj	I (1)	Rotated lat/lon grid for the reflectivity composites: number of grid points in rotated zonal ("South-North") direction.	COSMO = ie_tot ICON = 461
comp_meta%pollon	R (1)	Rotated lat/lon grid for the reflectivity composites: geogr. longitude of rotated North-pole [°]. For a non-rotated grid set to -180.0.	COSMO = pollon ICON = -170.0
comp_meta%pollat	R (1)	Rotated lat/lon grid for the reflectivity composites: geogr. latitude of rotated North-pole [°]. For a non-rotated grid set to 90.0.	COSMO = pollat ICON = 40.0
comp_meta%polgam	R (1)	Rotated lat/lon grid for the reflectivity composites: angle between the North poles of two rotated grids [°]. Normally set to 0.0.	COSMO = polgam ICON = 0.0
comp_meta%startlon	R (1)	Rotated lat/lon grid for the reflectivity composites: lower left corner longitude in rotaded coordinates [°].	COSMO = startlon_tot ICON = -5.0
comp_meta%startlat	R (1)	Rotated lat/lon grid for the reflectivity composites: lower left corner latitude in rotaded coordinates [°].	COSMO = startlat_tot ICON = -5.0
comp_meta%dlon	R (1)	Rotated lat/lon grid for the reflectivity composites: angular resolution [°] in rotated meridional direction.	COSMO = dlon ICON = 0.025
comp_meta%dlat	R (1)	Rotated lat/lon grid for the reflectivity composites: angular resolution [°] in rotated zonal direction.	COSMO = dlat ICON = 0.025
nel_composite	I (1)	Number of composites to generate. The elevations for each station to apply for these composites are specified by the global background index list in the namelist parameter `eleindlist_for_composite_glob`. This list can be adjusted for each single station by `rs_meta(`*i*`)%eleindlist_for_composite_glob`. If the list is longer, the first `nel_composite` entries will be used.	2

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
eleindlist_for_composite_glob	I (nel_composite_max)	For `ldo_composite=.TRUE.`: global list of elevation indices to construct the composites. Will serve as the global default list for all stations, which can however be adjusted for each individual station by `rs_meta(`i`)%eleindlist_for_composite` (see Table 6). The first element denotes the elevation index for the first composite, the second for the second composite, and so on. 1... `nel_max` = Take this elevation of all volume scans. If a station has less elevations, it will be clipped to the largest elevation index of this station. 98 = Take the precipitation scans (DWD-radar only). If precipitation scan is missing for certain stations, no data from these radars will appear in the composite. 99 = Take the vertical maximum of all elevations of all radars other: Not allowed, model run will be terminated. For precipitation scans: the elevations for lat/lon computations will not be the true elevations but the nominal elevation (either 0.4°, 0.59°, 0.8°, or 1.3°).	(/ 1, 2 /)
levelidlist_for_composite_glob	I (nel_composite_max)	For `ldo_composite=.TRUE.`: global list of grib2 level identifiers for the composites. The first identifier in the list corresponds to the first composite in the list `eleindlist_for_composite_glob`, the second identifier to the second composite and so on. This identifier is written to the grib2-keys `level` and `scaledValueOfFirstFixedSurface`. Negative values lead to a crash of `grib_api` and are not allowed.	eleindlist_for_composite_glob
lsmooth_composite_bub_glob	L (1)	Not yet in the namelist, up to now hardcoded in `radar_namelist_read.f90`	.FALSE.
nsmoothpoints_for_comp_bub_glob	I (1)	Not yet in the namelist, up to now hardcoded in `radar_namelist_read.f90`	9
nfilt_for_comp_bub_glob	I (1)	Not yet in the namelist, up to now hardcoded in `radar_namelist_read.f90`	1
ldo_bubbles	L (1)	Enable the "warm bubble generator" to trigger missing convective cells in the model by artificial warm bubbles inspired by Weisman and Klemp (1982). The detection of missing convective cells is based on simulated and observed radar composites (cf. Section 2.7). The bubble's type, amplitude, size, shape and duration can be defined by additional namelist parameters below. The exact locations and model times are detected by the bubble generator. More informations can be found in Section 7. Uses the same composite grid (`comp_meta%...`) and elevation index conventions as for `ldo_composite=.TRUE.`. Only effective if `lreadmeta_from_netcdf=.TRUE.`	.FALSE.
eleind_for_composite_bub_glob	I (1)	For `ldo_bubbles=.TRUE.`: elevation index to be used for the composite to detect the need for warm bubbles. Same valid values as for `eleindlist_for_composite_glob`.	-99
lsmooth_composite_bub_glob	L (1)	For `ldo_bubbles=.TRUE.`: If composite is to be smoothed by binomial filter. Not yet in the namelist, up to now hardcoded in `radar_namelist_read.f90`	.FALSE.
nsmoothpoints_for_comp_bub_glob	I (1)	For `ldo_bubbles=.TRUE.`: width of symetric 2D binomial smoother in grid points. Not yet in the namelist, up to now hardcoded in `radar_namelist_read.f90`	9

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
nfilt_for_comp_bub_glob	I (1)	For `ldo_bubbles=.TRUE.`: number of consecutive applications of the smoother. Not yet in the namelist, up to now hardcoded in `radar_namelist_read.f90`	1
dt_bubble_search	R (1)	For `ldo_bubbles=.TRUE.`: time interval from one automatic bubble search to the next [s].	900.0
t_offset_bubble_trigger_async	R (1)	For `ldo_bubbles=.TRUE.` and in case of asynchroneous radar IO for runtime optimization: time delay of (advection corrected) bubble triggering after detection step [s]. In an ideal world, one would set this delay to 0, but this prevents the worker PEs to continue with model integration in parallel to the IO PEs, which detect, among other tasks, the missing cells. If triggering is delayed, the worker PEs can continue for this amout of time until they gather the bubble parameters from the IO PEs and trigger the warm bubbles.	0.0
prob_bubble	R (1)	For `ldo_bubbles=.TRUE.`: probability of triggering a bubble when it is found [0-1].	1.0
lbub_isolated	L (1)	For `ldo_bubbles=.TRUE.`: check that the bubbles are isolated from other objects in observations.	.FALSE.
maxdim_obs	R (1)	For `ldo_bubbles=.TRUE.`: maximum dimension of missing cell that can trigger a bubble (larger objects are not targeted) [m].	75000.0
threshold_obs	R (2)	For `ldo_bubbles=.TRUE.`: thresholds for observed composite that define the minimum dBZ in an object- and the high intensity region [dBZ].	(/25.0, 30.0/)
threshold_mod	R (2)	For `ldo_bubbles=.TRUE.`: thresholds for simulated composite that define the minimum dBZ in an object- and the high intensity region [dBZ].	(/25.0, 30.0/)
areamin_obs	R (2)	For `ldo_bubbles=.TRUE.`: minimum area in observed composite of the object- and high intensity region for detecting an object [m^2]	(/ 25e6, 9e6/)
areamin_mod	R (2)	For `ldo_bubbles=.TRUE.`: minimum area in simulated composite of the object- and high intensity region for detecting an object [m^2]	(/ 25e6, 9e6/)
mult_dist_obs	R (1)	For `ldo_bubbles=.TRUE.`: multiplicative axis lenght factor for the observed object to define the minimal required ellisoidal distance frame required for simulated objects to be defined as “isolated” and trigger bubbles [-].	1.0
mult_dist_mod	R (1)	For `ldo_bubbles=.TRUE.`: multiplicative axis lenght factor for simulated objects to define the minimal required ellisoidal distance frame required forthe simulated objects to be defined as “isolated” and trigger bubbles [-].	1.0
add_dist_obs	R (1)	For `ldo_bubbles=.TRUE.`: additive axis increase for the observed object to define the minimal required ellisoidal distance frame required for simulated objects to be defined as “isolated” and trigger bubbles [m]. Is applied after mult_dist_obs.	1.0
add_dist_mod	R (1)	For `ldo_bubbles=.TRUE.`: additive axis increase for simulated objects to define the minimal required ellisoidal distance frame required forthe simulated objects to be defined as “isolated” and trigger bubbles [-]. Is applied after mult_dist_mod.	1.0
dt_bubble_advect	R (1)	For `ldo_bubbles=.TRUE.`: time scale for downstream advection of automatic bubbles [sec].	300.0

Table 4: continued

Name	Kind (Dim.)	Description / Remarks	Default
zlow_meanwind_bubble_advect	I (1)	For `ldo_bubbles=.TRUE.`: the lower bound of averaging height interval for computing the integral-averaged advection speed for automatic bubbles [m MSL].	3000.0
zup_meanwind_bubble_advect	I (1)	For `ldo_bubbles=.TRUE.`: the lower bound of averaging height interval for computing the integral-averaged advection speed for automatic bubbles [m MSL].	6000.0
bubble_type	C (12)	For `ldo_bubbles=.TRUE.`: type of the bubble (`'cos-hrd'` or `'cos-instant'`)	`'cos-hrd'`
bubble_heatingrate	R (1)	For `ldo_bubbles=.TRUE.`: heating rate for the bubbles of type 'cos-hrd' [K/s]	0.015
bubble_timespan	R (1)	For `ldo_bubbles=.TRUE.`: timespan for heating the bubbles of type 'cos-hrd' [s]	200.0
bubble_dT	R (1)	For `ldo_bubbles=.TRUE.`: temperature disturbance for the bubbles of type 'cos-instant' [K]	3.0
bubble_centz	R (1)	For `ldo_bubbles=.TRUE.`: center height MSL (Z) of the bubbles [m]	2000.0
bubble_radx	R (1)	For `ldo_bubbles=.TRUE.`: horizontal radius (main axis) in X-dir of the bubbles [m]	7500.0
bubble_rady	R (1)	For `ldo_bubbles=.TRUE.`: horizontal radius (main axis) in Y-dir of the bubbles [m]	7500.0
bubble_radz	R (1)	For `ldo_bubbles=.TRUE.`: vertical radius in Z-dir of the bubbles [m]	1400.0
bubble_rotangle	R (1)	For `ldo_bubbles=.TRUE.`: rotation angle of the main axes of bubbles [°]	0.0
bubble_holdrhconst	L (1)	For `ldo_bubbles=.TRUE.`: switch to choose if RH should kept constant during heating or not	.TRUE.
bubble_addnoise	L (1)	For `ldo_bubbles=.TRUE.`: switch to activate some random noise on the bubbles with a relative amplitude of `bubble_dT_noise`	.FALSE.
bubble_dT_noise	R (1)	For `ldo_bubbles=.TRUE.` and in case of `bubble_addnoise_T=.true.`, `bubble_dT_noise` is the relative noise level η, such that $\Delta T_{bubble} = \Delta T_{bubble,0}(1+\eta)$ with $\eta \in [-1, 1]$)	0.1
ldo_bubbles_manual	L (1)	Setting this switch to true enables to manually specify artificial convection triggers in real case simulations. This nees an explicit namelist `/ARTIFCTL/` in a file `INPUT_IDEAL` as described in Blahak (2015) with the relevant convection trigger parameters for atmospheric temperature- and humidity disturbances. Disturbances in the soil and all other namelist parameters of `/ARTIFCTL/` will be ignored.	.FALSE.
lwrite_ready	L (1)	Switch to enable the writing of so-called READY-files, cf. Section 6.3. If .TRUE., files having names like `READY_EMVORADO_20200215123500` are written into the output directory at the end of each EMVORADO output time step. Postprocessing jobs on output files for this timestep may start as soon as the READY-file exists, concurrently to the model run.	.FALSE.

5.1.3 Station metadata in namelist `rs_meta(`i`)` type in `/RADARSIM_PARAMS/`

Table 5: Table of radar station metadata for station i in `/RADARSIM_PARAMS/` in an element `rs_meta(`i`)` of the vector `rs_meta` of derived type `radar_meta_type` (module `radar_data.f90`). Kind abbreviations: "I" = INTEGER, "R" = REAL/DOUBLE, "C" =CHARACTER, "L" = LOGICAL, "T" = Derived TYPE. The defaults depend on `rs_meta(`i`)%icountry` respectively namelist parameter `icountry`. In this table, we assume `rs_meta(`i`)%icountry` = 1 (Germany).

Name rs_meta(i)%	Kind (Dim.)	Description / Remarks	Default
icountry	I (1)	Country flag for this radar. Is initialized by the global namelist parameter `icountry` and can be altered for each individual station i via namelist. Current possible values are: 1 = Germany (DWD) 2 = Switzerland (MeteoSwiss) 3 = Italy (ARPA-SIMC) This choice influences the defaults for the below type components in `rs_meta(`i`). In this table, we have assumed rs_meta(`i`)%icountry = 1`	`icountry`
station_id	I (1)	WMO station ID of the radar station (country code + national ID). This ID should be explicitly given in the namelist for each station block i.	999999
station_name	C (3)	3-character station akronym.	'XXX'
lambda	R (1)	Radar wavelength [m]. For efficiency reasons, it can be advantageous not to choose the very exact wavelength. For example, in a network of C-Band radars, choose all wavelengths to be the same 0.055 m to enable re-use of simulated grid-point reflectivity values from one radar station to the next to save computing time.	0.055
lon	R (1)	Station geographical longitude [deg]	Domain center
lat	R (1)	Station geographical latitude [deg]	Domain center
alt_agl_mod	R (1)	Station height above model orography [m]. Represents the height of the radar antenna above ground.	50.0
alt_msl	R (1)	Station height above MSL [m]. This height will internally used for all height-related computations. If it is set to the value -9999.99, it will automatically be computed as `rs_meta(`i`)%alt_agl_mod` plus model orography height at the station location.	-9999.99
alt_msl_true	R (1)	True station height MSL [m] as read from observation files [m]. Only relevant if `lreadmeta_from_netcdf=.TRUE.`	-9999.99
naz	I (1)	Number of nominal azimuths (radials) within a PPI-elevation	360
az_start	R (1)	Start azimuth of first radial relative to True North [deg]	0.0
az_inc	R (1)	Azimuth increment [deg] from radial to radial	1.0
nra	I (1)	Max. number of range bins occuring in a volume scan (sometimes, higher elevations have less range bins, but here the maximum number of all elevations is required)	124
ra_inc	R (1)	Range increment [m] from bin to bin anlong a radial	1000.0
nel	I (1)	Number of elevations of the PPI volume scan	18

Table 5: continued

Name rs_meta(i)%	Kind (Dim.)	Description / Remarks	Default
el_arr	R (nel_max)	List of nominal elevation angles [deg] of the PPI volume scan. The list should contain `rs_meta(i)%nel` elements. These elevation angles are actually used for all computations and, in case of `lreadmeta_from_netcdf=.TRUE.`, preferred over the "true" elevation angles given in some observation files. One exception are DWD's "precipitation scans" with their horizon-following nominal elevations. Here, just one "fake" elevation out of (0.4, 0.59, 0.8, 1.3) has to be given and `rs_meta(i)%nel` = 1. However, this only works for stations with a valid German station id, and the elevation as function of azimuth depends on `rs_meta(i)%station_id` as defined in the module `radar_elevations_precipscan.incf`.	(/0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 11.0, 13.0, 15.0, 17.0, 19.0, 23.0, 29.0, 37.0/)
scanname	C (10)	This parameter cannot really be set by namelist but is automatically determined from `rs_meta(i)%el_arr`. It appears in the namelist control output file `YUSPECIF_RADAR` (COSMO) respectively `nml.emvorado.log` (ICON) and is part of some of EMVORADO's output file names. It consists of 'PRECIP' (if DWD precipitation scan) or 'PPI' plus a 4-digit number representing the average elevation angle of the scan times 10. Examples are 'PPI0080', 'PRECIP'.	'PPI0119'
nel_default	I (nscanstrategies_max)	Only relevant if `lreadmeta_from_netcdf=.TRUE.`: in this case, each radar observation file has to conform to one of certain allowed scan strategies. If not, the corresponding station is discarded from the simulation. `nel_default` is the number of elevations for each of these allowed default scan strategies. Depends on `rs_meta(i)%icountry` and is predefined accordingly in the module `radar_obs_meta_list.f90`, so no need to specify it explicitly in the namelist.	(/18, 18, 10, 10, 1, 1, 1, 1, 3, 3/)
el_arr_default	R (nel_max, nscanstrategies_max)	Only relevant if `lreadmeta_from_netcdf=.TRUE.`: array of the `rs_meta(i)%nel_default` allowed default scan strategies. Again depends on `rs_meta(i)%icountry` and is predefined accordingly in the module `radar_obs_meta_list.f90`, so no need to specify it explicitly in the namelist.	See function `get_meta_proto_dwd()` in `radar_obs_meta_list.f90`
el_arr_obs	R (nel_max)	Only relevant if `lreadmeta_from_netcdf=.TRUE.`: List of "true" elevations [deg] from observation files, used for cross-checking with `rs_meta(i)%el_arr`. Will be filled automatically during reading of observation files.	(/0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 11.0, 13.0, 15.0, 17.0, 19.0, 23.0, 29.0, 37.0/)
obs_times	R (nobstimes_max)	List of observation times in seconds since model run start. Any values ≥ 0.0 define the desired times for simulating volume scans of this radar station. If no value ≥ 0.0 is given, observation times are automatically computed from the parameters `rs_meta(i)%nobs_times` and `rs_meta(i)%dt_obs`.	-999.9
dt_obs	R (1)	Observation time increment [s]. Only relevant if no `rs_meta(i)%obs_times` $\geq$ 0.0 are given. The list of `rs_meta(i)%obs_times` is computed in intervals of dt_obs starting from 0.0 until `rs_meta(i)%nobs_times` steps.	300.0

Table 5: continued

Name rs_meta(i)%	Kind (Dim.)	Description / Remarks	Default
nobs_times	I (1)	Number of observation times. Only relevant if no `rs_meta(`i`)%obs_times` $\geq$ 0.0 are given. If $<$ 0, the list of `rs_meta(`i`)%obs_times` is computed to fill the entire model simulation time with observation times in regular intervals of `rs_meta(`i`)%dt_obs` starting from 0.0	-999
obs_cdate	C (14) (nobstimes_max)	List of observation times in character representation `'YYYYMMDDhhmmss'`, e.g., '20180527243500'. Automatically determined from `rs_meta(`i`)%obs_times`, so no need to specify in the namelist.	'YYYYMMDDHHMMSS'
ext_nyq	R (nel_max)	Extended Nyquist velocity for each elevation [m/s] including techniques like Dual-PRF. Relevant for dealiasing observed radial winds (`ldealiase_vr_obs=.TRUE.`).	32.5
high_nyq	R (nel_max)	Nyquist velocity for each elevation [m/s] corresponding to the higher of the two PRF in case of Dual-PRF. Has no application in EMVORADO, but is part of some observation files.	32.5
prf	R (nel_max)	PRF for each elevation [1/s], in case of Dual-PRF this is one of the two. Has no application in EMVORADO, but is part of some observation files.	
dualprf_ratio	R (nel_max)	Ratio of the PRFs for the Dual-PRF method for each elevation. Has no application in EMVORADO, but is part of some observation files.	4/3
rngate_len	R (1)	Range gate length [m] of the "raw" radar echoes before range averaging in the signal processor (`rs_meta(`i`)%ra_inc`) to reduce the statistical noise. Has no application in EMVORADO, but is part of some observation files.	125.0
num_gates	I (1)	Number of averaged range gates in the signal processor to reduce the statistical noise. Has no application in EMVORADO, but is part of some observation files.	0
num_pulses	I (1)	Number of integrated pulses ("raw" azimuths) in the signal processor to reduce the statistical noise for one stored azimuth. Has no application in EMVORADO, but is part of some observation files.	0
mds_Z0	R (1)	Relevant if `lmds_vr=.TRUE.` or `lmds_z=.TRUE.`: minimum detectable signal [dBZ] at the reference range `rs_meta(`i`)%mds_r0`. Reference value for the quadratic dependence of actual minimum detectable signal on range. Smaller simulated reflectivities are set to -99.99 dBZ ("correct zero"), the corresponding radial winds to -999.99 m/s ("missing").	-20.0
mds_r0	R (1)	Relevant if `lmds_vr=.TRUE.` or `lmds_z=.TRUE.`: reference range for minimum detectable signal [m].	10000
fdbkfile	C (60)	For `lreadmeta_from_netcdf=.TRUE.` and `lfdbk_output=.TRUE.`: Name of NetCDF feedback file for radial wind and reflectivity. Automatically created from `rs_meta(`i`)%station_id`, `rs_meta(`i`)%scanname` and simulation start time, so no need to specify in the namelist. Feedback files will be created in directory `ydirradarout`.	'nofile_fdbk'
ncdffile	C (60) (nobstimes_max, ndatakind)	Names of radar observation input files for the different variables (`ndatakind`). Automatically filled with the names of files following the recognized patterns for the implemented countries and found in directory `ydirradarin`. No need to specify them in the namelist.	'nofile_obs'

Table 5: continued

Name rs_meta(i)%	Kind (Dim.)	Description / Remarks	Default
ngpsm_v	I (1)	For `lsmooth` =.TRUE.: number of vertical smoothing points for Gauss-Legendre-quadrature. The higher the number, the more memory is needed. At most 9 points are usually sufficient.	9
ngpsm_h	I (1)	For `lsmooth` =.TRUE.: number of horizontal smoothing points for Gauss-Legendre-quadrature. Not as important as vertical smoothing, so at most 3 points are sufficient. Most of the time, even 1 is sufficient.	1
xabscsm_v	R (ngpsm_max)	For `lsmooth` =.TRUE.: list of normalized nodes (`ngpsm_v` values $\in [-1, 1]$) for vertical Gauss-Legendre quadrature [-]. Automatically determined, so no need to set it in the namelist.	0.0
weigsm_v	R (ngpsm_max)	For `lsmooth` =.TRUE.: list of wieghts (`ngpsm_v` values $\in [0, 1]$ whose sum is 2.0) for vertical Gauss-Legendre quadrature [-]. Automatically determined, so no need to set it in the namelist.	1.0
xabscsm_h	R (ngpsm_max)	For `lsmooth` =.TRUE.: list of normalized nodes (`ngpsm_h` values $\in [-1, 1]$) for horizontal Gauss-Legendre quadrature [-]. Automatically determined, so no need to set it in the namelist.	0.0
weigsm_h	R (ngpsm_max)	For `lsmooth` =.TRUE.: list of wieghts (`ngpsm_h` values $\in [0, 1]$ whose sum is 2.0) for horizontal Gauss-Legendre quadrature [-]. Automatically determined, so no need to set it in the namelist.	1.0
Theta3	R (1)	For `lsmooth` =.TRUE.: vertical 3-dB-oneway beam width [deg]. Half width of the one-way beamfunction.	1.0
Phi3	R (1)	For `lsmooth` =.TRUE.: horizontal 3-dB-oneway beam width [deg].	1.0
dalpha	R (1)	For `lsmooth` =.TRUE.: Angular averaging interval [deg] for the azimuthal pulse averaging (`num_pulses`). Relevant for the effective beam weighting function (Blahak, 2008).	1.0
alpha3_eff_0	R (1)	For `lsmooth` =.TRUE.: Effective horizontal 3-dB-oneway beam width [deg] at elevation=0.0°, depending on `phi3` and the ratio `dalpha/phi3`. Will be automatically determined from the lookup table given in Blahak (2008).	1.461
smth_interv_fact	R (1)	Factor to determine the azimutal and elevational integration range for the smoothing over the effective beam weighting function. The ranges are computed by multiplying this factor to the effective 3-dB-oneway beamwidth in azimuthal direction. A value of 1.29 leads to the 90-%-weight-range of the beam function (Blahak, 2008).	1.29

Table 5: continued

Name rs_meta(i)%	Kind (Dim.)	Description / Remarks	Default
eleindlist_for_composite	I (nel_composite_max)	For `ldo_composite=.TRUE.`: individual list of elevation indices to construct the composites. Will be initialized by the global list `eleindlist_for_composite_glob`, but can be adjusted for each station. The first element denotes the elevation index for the first composite, the second for the second composite, and so on. Valid values are: 1... `rs_meta(i)%nel` = Take this elevation of the volume scan. 98 = Take the precipitation scan (DWD-radar only). If no precipitation scan for this radar is in the observations, no data from this radar will appear in the composite 99 = Take the vertical maximum of all elevations of this radar. other: Will be folded into the range [1, `rs_meta(i)%nel`]. For precipitation scans: the elevations for lat/lon computations will not be the true elevations but the nominal elevation (either 0.4°, 0.59°, 0.8°, or 1.3°).	`eleindlist_for_composite_glob`
eleind_for_composite_bub	I (1)	For `ldo_bubbles=.TRUE.`: individual elevation index for this radar station to construct the composite for detecting the need for artificial warm bubbles. Same valid values as for `eleindlist_for_composite`.	1
nel_fdbk	I (1)	For `lfdbk_output=.TRUE.`: actual number of elevations to be written into the feedback file for this station. Is preset by the number of valid elevation indices in the global list `ind_ele_fdbk_glob`, but can be adjusted for each individual station.	# of valid values in `ind_ele_fdbk_glob`
ind_ele_fdbk	I (nel_max)	For `lfdbk_output=.TRUE.`: list of indices of elevations which are written the into feedback file for this station. Is preset by the global list `ind_ele_fdbk_glob`, but can be adjusted for each individual station.	`ind_ele_fdbk_glob`
nel_voldata	I (1)	For `lvoldata_output=.TRUE.`: actual number of elevations to be written into the volume data files for this station. Is preset by the number of valid elevation indices in the global list `ind_ele_voldata_glob`, but can be adjusted for each individual station.	# of valid values in `ind_ele_voldata_glob`
ind_ele_voldata	I (nel_max)	For `lvoldata_output=.TRUE.`: list of indices of elevations which are written the into volume data files for this station. Is preset by the global list `ind_ele_voldata_glob`, but can be adjusted for each individual station.	`ind_ele_voldata_glob`
obs_times_fdbk	R (nobstimes_max)	List of times for which radar data should be written to the feedback files [s since model start] for station i. This is useful to tailor the amount of data to the specific needs of the data assimilation system, independent from the volume data output. Is preset by the global list `obs_times_fdbk_glob`. Should match with existing `rs_meta(i)%obs_times`, otherwise no output for for this time. Takes precedence over time specification via `rs_meta(i)%dt_obs_fdbk` and `rs_meta(i)%nobs_times_fdbk`. If empty or -999.99, `rs_meta(i)%dt_obs_fdbk` will be evalutated instead.	-999.9

Table 5: continued

Name rs_meta(i)%	Kind (Dim.)	Description / Remarks	Default
dt_obs_fdbk	R (1)	Time increment for writing radar data to feedback files [s] for this station. Active only if `rs_meta(i)%obs_times_fdbk(:)=-999.99`. Is used to build the list of desired output times in equal steps starting at model start time for `rs_meta(i)%nobs_times_fdbk` steps. Is preset by the global parameter `dt_obs_fdbk_glob`. If empty or -999.99, all existing `rs_meta(i)%obs_times` will be output.	-999.9
nobs_times_fdbk	I (1)	If `rs_meta(i)%dt_obs_fdbk` [s] is used to specify the list of output times for data into feedback files, this defines the number such time steps since model start to output. If empty or -999, the steps will fill the entire model forecast range.	-999
obs_times_voldata	R (nobstimes_max)	List of times for which for which volume data output is desired [s since model start] for station i. This is useful to tailor the amount of volume data to the specific user needs, independent from the feedback file output. Is preset by the global list `obs_times_voldata_glob`. Should match with existing `rs_meta(i)%obs_times`, otherwise no output for this time. Takes precedence over time specification via `rs_meta(i)%dt_obs_voldata` and `rs_meta(i)%nobs_times_voldata`. If empty or -999.99, `rs_meta(i)%dt_obs_voldata` will be evalutated instead.	-999.9
dt_obs_voldata	R (1)	Time increment for writing radar data to feedback files [s] for this station. Active only if `rs_meta(i)%obs_times_voldata(:)=-999.99`. Is used to build the list of desired output times in equal steps starting at model start time for `rs_meta(i)%nobs_times_voldata` steps. Is preset by the global parameter `dt_obs_voldata_glob`. If empty or -999.99, all existing `rs_meta(i)%obs_times` will be output.	-999.9
nobs_times_voldata	I (1)	If `rs_meta(i)%dt_obs_voldata` [s] is used to specify the list of output times for volume data, this defines the number such time steps since model start to output. If empty or -999, the steps will fill the entire model forecast range.	-999

5.1.4 Reflectivity config in namelist `dbz_meta(`i`)` type in `/RADARSIM_PARAMS/`

This derived type for parameter definitions of step 1 of the reflectivity operator resembles that of Sections 6.3 to 6.5 of Blahak (2016), but the parameters for the deprecated option `itype_refl=2` are inactive and should not be explicitly used. They have been left out in the below table. Also, the components `llookup_mie` and `lhydrom_choice_testing` have not been mentioned there.

Table 6: Reflectivity computation parameters for radar station i in `/RADARSIM_PARAMS/` in an element `dbz_meta(`i`)` of the vector instance `dbz_meta` of derived type `dbzcalc_params` (module `radar_data.f90`). Kind abbreviations: "I" = INTEGER, "R" = REAL/DOUBLE, "C" =CHARACTER, "L" = LOGICAL, "T" = Derived TYPE.

Name dbz_meta(i)%	Kind (Dim.)	Description / Remarks	Default
station_id	I (1)	6-digit WMO station ID of the radar station (country code + national ID). Setting it explicitly has no effect, because the final value will be overtaken from the corresponding radar station metadata block, `rs_meta(`i`)%station_id`.	999999
lambda_radar	R (1)	Radar wavelength [m]. Does not take effect, because the correct radar wavelength is overtaken from the corresponding radar station metadata block, `rs_meta(`i`)%lambda`.	0.055
`itype_refl`	`I`	Type of reflectivity calculation (cf. Section 2.4): 1 = Mie (Blahak, 2016) 3 = Rayleigh-Oguchi (Blahak, 2016) 4 = "Old" Rayleigh from COSMO `pp_utilities.f90`	`3`
`llookup_mie`	`L (1)`	Switch to enable the use of efficient lookup tables for Mie scattering. Only effective if `itype_refl=1`.	`.TRUE.`
`igraupel_type`	`I (1)`	Type of melting graupel particle model for Mie Scattering `dbz_meta(`i`)%itype_refl`=1: 1 = simple spheres, soaked ice-air-water-mixtures 2 = two-layered spheres, ice-air core surrounded by ice-water shell 3 = two-layered spheres, ice-air core surrounded by pure water shell	`1`
`ctype_drysnow_mie`	`C (6)`	String for defining the EMA of the dry snow category. Particles are assumed to be two-layered spheres of ice-air mixtures having different volume ratios in core and shell. The first 3 characters represent the core material according to Table 38 in Section 6.4 of Blahak (2016), the last 3 characters the shell accordingly. Only effective if `dbz_meta(`i`)%itype_refl=1`.	`'masmas'`
`ctype_wetsnow_mie`	`C (12)`	String for defining the EMA of the melting snow category. Particles are assumed to be two-layered spheres of ice-air-water mixtures having different volume ratios in core and shell. The first 6 characters represent the core material according to Table 40 in Section 6.4 of Blahak (2016), the last 6 characters the shell accordingly. Only effective if `dbz_meta(`i`)%itype_refl=1`.	`'mawsasmawsms'`
`ctype_drygraupel_mie`	`C (3)`	String for defining the EMA of the dry graupel category. Particles are assumed to be simple spheres of an ice-air mixture. The 3 characters are accoding to Table 38 in Section 6.4 of Blahak (2016). Only effective if `dbz_meta(`i`)%itype_refl=1`.	`'mis'`

Table 6: continued

Name dbz_meta(i)%	Kind (Dim.)	Description / Remarks	Default
`ctype_wetgraupel_mie`	`C (6)`	String for defining the EMA of the melting graupel category. Depends on `dbz_meta(`i`)%igraupel_type`. If `dbz_meta(`i`)%igraupel_type`=1: Particles are assumed to be simple spheres of an ice-air-water mixture. The 6 characters are accoding to Table 40 in Section 6.4 of Blahak (2016). If `dbz_meta(`i`)%igraupel_type`=2: Particles are assumed to be two-layered spheres of an ice-air core surrounded by an ice-water shell. The first 3 characters represent the core material and the last 3 characters the shell according to Table 41 in Section 6.4 of Blahak (2016). If `dbz_meta(`i`)%igraupel_type`=3: Particles are assumed to be two-layered spheres of an ice-air core surrounded by a pure water shell. Only 3 characters are needed and represent the core material according to Table 38 in Section 6.4 of Blahak (2016). Only effective if `dbz_meta(`i`)%itype_refl=1`.	`'mawsms'`
`ctype_dryhail_mie`	`C (3)`	String for defining the EMA of the dry hail category. Particles are assumed to be simple spheres of an ice-air mixture. The 3 characters are accoding to Table 38 in Section 6.4 of Blahak, 2016. Only effective if `dbz_meta(`i`)%itype_refl=1`.	`'mis'`
`ctype_wethail_mie`	`C (3)`	String for defining the EMA of the melting hail category. Particles are assumed to be simple spheres of an ice-water mixture. The 3 characters are accoding to Table 39 in Section 6.4 of Blahak (2016). Only effective if `dbz_meta(`i`)%itype_refl=1`.	`'mws'`
`lhydrom_choice_testing`	`L (6)`	Vector of switches to enable/disable single hydrometeor types in reflectivity calculations. The order of the switches by index is 1 = cloud water, 2 = rain, 3 = cloud ice, 4 = snow, 5 = graupel, 6 = hail. E.g., if you set `dbz_meta(`i`)%lhydrom_choice_testing= .TRUE., .FALSE., .TRUE., .TRUE., .FALSE., .TRUE.,` rain and graupel will be excluded from the reflectivity calculations. Can be helpful in software development and testing.	all `.TRUE.`
`Tmeltbegin_s`	`R (1)`	Temperature [K], above which snow is assumed wet	`273.16`
`meltdegTmin_s`	`R (1)`	Degree of snow melting at T=273.16 K	`0.0`
`Tmeltbegin_g`	`R (1)`	Temperature [K], above which graupel is assumed wet	`263.16`
`meltdegTmin_g`	`R (1)`	Degree of graupel melting at T=273.16 K	`0.2`
`Tmeltbegin_h`	`R (1)`	Temperature [K], above which hail is assumed wet	`263.16`
`meltdegTmin_h`	`R (1)`	Degree of hail melting at T=273.16 K	`0.2`

5.1.5 Adapting components of derived types `rs_meta(`i`)` and `dbz_meta(`i`)` in real mode with observations

In real case simulations with using observation files (`lreadmeta_from_netcdf` = .TRUE.) it is possible to overwrite any station metadata and reflectivity computation metadata, after the metadata have been read from the observation files and have been matched against the background metadata list in the code. For example, one can artificially move radar stations to different locations, one can change the height of the stations, one can change the default scan strategies (`rs_meta(`i`)%nel_default(`k`)` and `rs_meta(`i`)%el_arr_default(:,`k`)`) to allow for "unusual" scan strategies in observation files, one can define individual settings for the beam smoothing parameters or the reflectivity computations, one can define individual elevation- and time thinning for feedback- and volume data files, and so on.

This can be helpful for many things. For example, developers can set up specialized test cases, or the data amount of EMVORADO output can be reduced individually for operational applications.

If `lreadmeta_from_netcdf` = .TRUE., `nradsta_namelist` has a different meaning as for `lreadmeta_from_netcdf` = .FALSE. Instead of the simulated number of radar stations, it is the number of stations for which the user wants to overwrite any of the metadata.

For example, if `nradsta_namelist` is set to 3, the user wants to change 3 radar stations, and consequently the metadata blocks `rs_meta(1)`, `dbz_meta(1)`, `rs_meta(2)`, `dbz_meta(2)` and `rs_meta(3)`, `dbz_meta(3)` are recognized in the namelist to define the desired changes. If more blocks are present, only the ones with i=1...3 take effect.

The matching between a block index i and the actual radar station is achieved via the `rs_meta(`i`)%station_id` and `rs_meta(`i`)%scanname`. For this, each `rs_meta(`i`)` block in the namelist has to contain these informations in addition to the desired changed radar parameters, otherwise it cannot be correctly matched. The scanname identifies a specific scan strategy of a radar station as described in Table 5. Internally, EMVORADO treats two different scan strategies of the same radar as two different radars! For example, if one wants to adapt the radar wavelength and the station height of station 10908 and scan strategy `PPI0080`, the namelist entries would be

```
nradsta_namelist = 1

rs_meta(1)%station_id = 10908,
rs_meta(1)%scanname   = 'PPI0080',
rs_meta(1)%lambda     = 0.003,
rs_meta(1)%alt_msl    = 1516.0,
```

A `dbz_meta(`i`)` block is also matched by `rs_meta(`i`)%station_id` and `rs_meta(`i`)%scanname`. E.g., if in addition to the above the temperature threshold for beginning of graupel melting is to be changed for stations 10908 and 10950, the correct total block in the namelist is:

```
nradsta_namelist = 2

rs_meta(1)%station_id = 10908,
rs_meta(1)%scanname   = 'PPI0080',
rs_meta(1)%lambda     = 0.003,
rs_meta(1)%alt_msl    = 1516.0,
dbz_meta(1)%Tmeltbegin_g = 265.16,

rs_meta(2)%station_id = 10950,
rs_meta(2)%scanname   = 'PPI0080',
dbz_meta(2)%Tmeltbegin_g = 265.16,
```

`rs_meta(`i`)` and `dbz_meta(`i`)` are "paired" entities, both denoting the same station and scan strategy.

`rs_meta(`i`)%station_id` and `rs_meta(`i`)%scanname` are the only metadata that cannot be changed via namelist. Regarding the `rs_meta(`i`)%scanname`, if the actual scan strategy is modified (`rs_meta(`i`)%el_arr` only, do not change `rs_meta(`i`)%nel`!), the `rs_meta(`i`)%scanname` might no longer be consistent.

The radar wavelength is special. It is contained in both `rs_meta(`i`)%lambda` and `dbz_meta(`i`)%lambda_radar`, but the latter is simply overtaken from the former after all namelist- and metadata reading. Therefore, if the radar wavelength is to be changed via namelist, it has to be done via `rs_meta(`i`)%lambda`, not `dbz_meta(`i`)%lambda_radar`. There is also a `dbz_meta(`i`)%station_id`, but this is also simply overtaken from `rs_meta(`i`)%station_id` after all namelist- and metadata reading.

5.1.6 A remark about output of radar composites

As mentioned earlier in Section 2.7, observed and simulated reflectivity composites on an arbitrary rotated lat-lon grid can be produced in EMVORADO at the end of step 2, if `ldo_composite = .TRUE.`, `nel_composites` > 0 and `eleindlist_for_composites` defined appropriately in the namelist `/RADARSIM_PARAMS/`. Moreover, EMVORADO has its own grib2 output method for these composites, which is active if `lcomposite_output=.TRUE.` and which writes all simulated composites of an observation time to one grib2-file (likewise for the observation composites). To distinguish the different composites, the "level" and "scaledValueOfFirstFixedSurface" keys in the grib2-header of each composite are used as identifiers and are set equal to its index in the list of elevations `eleindlist_for_composites` by default. To give the user some more flexibility to label the composites individually, the namelist parameter `levelidlist_for_composite_glob` (list of integers) allows to replace this index by any positive number as level identifier.

A separate composite is the basis for automatic warm bubbles (`ldo_bubbles=.TRUE.`, cf. Section 7), which does also show up in the composite grib file(s). To distinguish it from the other composites, the given "level" is 0.

The grid for the composites is a rotated lat/lon grid similar to the model grid of COSMO and may be arbitrarily defined by namelist parameters in `/RADARSIM_PARAMS/` (components of the derived type `comp_meta`, see Table 4):

- `comp_meta%ni`
- `comp_meta%nj`
- `comp_meta%pollon`
- `comp_meta%pollat`
- `comp_meta%polgam`
- `comp_meta%startlon`
- `comp_meta%startlat`
- `comp_meta%dlon`
- `comp_meta%dlat`

For COSMO, the defaults for these parameters are directly overtaken from the model grid, i.e., if nothing is specified in the namelist, the composites are created on the model grid. For ICON, the defaults resemble the COSMO-DE grid.

Again for COSMO, if the composites are on the model grid and if EMVORADO is in synchroneous output mode (`nprocio_radar == 0` in COSMO `/RUNCTL/`), they are in principle also available for output via the "normal" COSMO grib output stream (grib1 or grib2) from the `yvarml`-Parameter of each `/GRIBOUT/` namelist, through the shortnames "`DBZCMP_SIM`" and "`DBZCMP_OBS`".

But as also mentioned earlier, this output method for composites is not recommended any more, because it is incompatible with the asynchroneous radar IO option and in principle does not allow the composite grid to be different from the model grid; "`DBZCMP_SIM`" and "`DBZCMP_OBS`" will contain only -999.99 values in these cases. Then, the separate grib2-output via EMVORADO (`lcomposite_output = .TRUE.`) is the only way of getting the correct composites, cf. Section 6.1 below. It should always be preferred.

In ICON, the composites can only be output through EMVORADO itself (`lcomposite_output = .TRUE.`.

5.2 Namelist parameters to control "traditional" grid point reflectivity output

To control the reflectivity computation on the model grid for the standard COSMO / ICON output fields `DBZ` (COSMO: `yvarml`, `yvarpl`, `yvarzl` in the `/GRIBOUT/` namelist(s)), `DBZ_850` (COSMO: `yvarml`), `DBZ_CMAX` (COSMO: `yvarml`) and `DBZ_CTMAX` (COSMO: `yvarml`) , there is a new derived type `dbz` in each output namelist (COSMO: `/GRIBOUT/`). This derived type is formally the same as `dbz_meta(`*i*`)` in `/RADARSIM_PARAMS/` (step 1 of the reflectivity operator). It largely resembles that of Sections 6.3 to 6.5 of Blahak (2016), but the parameters for the deprecated option `itype_refl=2` are inactive and should not be explicitly used; also, the components `llookup_mie` and `lhydrom_choice_testing` have not been mentioned there.

A "normal" user should only change the radar wavelength and the overall type of reflectivity computation, i.e., the parameters `dbz%lambda_radar` and `dbz%itype_refl`.

Each element of this derived type can be specified in the namelist, e.g., `dbz%itype_refl = 1`. All available elements are listed in Table 7. While formally the same as `dbz_meta(`*i*`)`, its effects are completely independent of it, as previously mentioned in Section 2.5. `luse_radarfwo` has no effect on it. This also means that for the grid point output, the directories where to read and write the lookup tables have to be specified independently fromm `/RADARSIM_PARAMS/`. For COSMO, `ydir_mielookup_read` and `ydir_mielookup_write` for this context are part of namelist `IOCTL`, see the COSMO User's Guide (Schättler et al., 2019). As mentioned in Section 2.6, normally these two directories should be equal.

In contrast to `dbz_meta(`*i*`)`, the radar wavelength `dbz%lambda_radar` is effective here, because it is not tied to a specific radar station as in `rs_meta(`*i*`)` — `dbz_meta(`*i*`)` — pairs (cf. Section 5.1.5).

Table 7: Derived type instance `dbz` to configure the grid point reflectivity output in COSMO grib files. `dbz` is of type `dbzcalc_params` from `radar_data.f90` and has the following components:

Name	Type	Definition / Purpose / Comments	Default
`dbz%lambda_radar`	`R (1)`	Radar wavelength [m]	`0.055`
`dbz%itype_refl`	`I (1)`	Type of reflectivity calculation (cf. Section 2.4): 1 = Mie (Blahak, 2016) 3 = Rayleigh-Oguchi (Blahak, 2016) 4 = "Old" Rayleigh from COSMO `pp_utilities.f90`	`4`
`dbz%igraupel_type`	`I (1)`	Type of melting graupel particle model for Mie Scattering `dbz%itype_refl`=1: 1 = simple spheres, soaked ice-air-water-mixtures 2 = two-layered spheres, ice-air core surrounded by ice-water shell 3 = two-layered spheres, ice-air core surrounded by pure water shell	`1`
`dbz%llookup_mie`	`L (1)`	Switch to enable the use of efficient lookup tables for Mie scattering. Only effective if `dbz%itype_refl=1`.	`.TRUE.`
`dbz%lhydrom_choice_testing`	`(6)`	Vector of switches to enable/disable single hydrometeor types in reflectivity calculations. The order of the switches by index is 1 = cloud water, 2 = rain, 3 = cloud ice, 4 = snow, 5 = graupel, 6 = hail. E.g., if you set `dbz%lhydrom_choice_testing= .TRUE., .FALSE., .TRUE., .TRUE., .FALSE., .TRUE.,` rain and graupel will be excluded from the reflectivity calculations. Can be helpful in software development and testing.	all `.TRUE.`
`dbz%Tmeltbegin_s`	`R (1)`	Temperature [K], above which snow is assumed wet	`273.16`
`dbz%meltdegTmin_s`	`R (1)`	Degree of snow melting at T=273.16 K	`0.0`
`dbz%Tmeltbegin_g`	`R (1)`	Temperature [K], above which graupel is assumed wet	`263.16`

Table 7: continued

Name	Type	Definition / Purpose / Comments	Default
dbz%meltdegTmin_g	R (1)	Degree of graupel melting at T=273.16 K	0.2
dbz%Tmeltbegin_h	R (1)	Temperature [K], above which hail is assumed wet	263.16
dbz%meltdegTmin_h	R (1)	Degree of hail melting at T=273.16 K	0.2
dbz%ctype_drysnow_mie	C (6)	String for defining the EMA of the dry snow category. Particles are assumed to be two-layered spheres of ice-air mixtures having different volume ratios in core and shell. The first 3 characters represent the core material according to Table 38 in Section 6.4 of Blahak (2016), the last 3 characters the shell accordingly. Only effective if `dbz%itype_refl=1`.	'masmas'
dbz%ctype_wetsnow_mie	C (12)	String for defining the EMA of the melting snow category. Particles are assumed to be two-layered spheres of ice-air-water mixtures having different volume ratios in core and shell. The first 6 characters represent the core material according to Table 40 in Section 6.4 of Blahak (2016), the last 6 characters the shell accordingly. Only effective if `dbz%itype_refl=1`.	'mawsasmawsms'
dbz%ctype_drygraupel_mie	C (3)	String for defining the EMA of the dry graupel category. Particles are assumed to be simple spheres of an ice-air mixture. The 3 characters are accoding to Table 38 in Section 6.4 of Blahak (2016). Only effective if `dbz%itype_refl=1`.	'mis'
dbz%ctype_wetgraupel_mie	C (6)	String for defining the EMA of the melting graupel category. Depends on `dbz%igraupel_type`. If `dbz%igraupel_type`=1: Particles are assumed to be simple spheres of an ice-air-water mixture. The 6 characters are accoding to Table 40 in Section 6.4 of Blahak (2016). If `dbz%igraupel_type`=2: Particles are assumed to be two-layered spheres of an ice-air core surrounded by an ice-water shell. The first 3 characters represent the core material and the last 3 characters the shell according to Table 41 in Section 6.4 of Blahak (2016). If `dbz%igraupel_type`=3: Particles are assumed to be two-layered spheres of an ice-air core surrounded by a pure water shell. Only 3 characters are needed and represent the core material according to Table 38 in Section 6.4 of Blahak (2016). Only effective if `dbz%itype_refl=1`.	'mawsms'
dbz%ctype_dryhail_mie	C (3)	String for defining the EMA of the dry hail category. Particles are assumed to be simple spheres of an ice-air mixture. The 3 characters are accoding to Table 38 in Section 6.4 of Blahak, 2016. Only effective if `dbz%itype_refl=1`.	'mis'
dbz%ctype_wethail_mie	C (3)	String for defining the EMA of the melting hail category. Particles are assumed to be simple spheres of an ice-water mixture. The 3 characters are accoding to Table 39 in Section 6.4 of Blahak (2016). Only effective if `dbz%itype_refl=1`.	'mws'

6 Output of EMVORADO

6.1 Formats

There are various possible outputs of simulated and observed radar data in EMVORADO. All output options can be enabled/disabled via `/RADARSIM_PARAMS/` namelist. However, some of them are only available depending on the pre-processor flags (cf. Section 3), the EMVORADO operation mode (cf. Section 4) and the EMVORADO configuration:

6.1.1 Volume scan data

Volume scan data may be output in the different formats listed below. The filenames contain keywords (called `datasetname` below) to signify the following possible output quantities (availability depends on EMVORADOs configuration):

Keyword	**Parameter**	**Dependencies (“if ...”)**
`'losim'`	simulated geographic longitude [°]	`lout_geom=.TRUE.`
`'lasim'`	simulated geographic latitude [°]	`lout_geom=.TRUE.`
`'hrsim'`	simulated height of radar bins [m MSL]	`lout_geom=.TRUE.`
`'ersim'`	simulated local beam elevation angle [°]	`lout_geom=.TRUE.`
`'adsim'`	simulated arc distance from radar site (great circle distance) [m]	`lout_geom=.TRUE.`
`'zrsim'`	simulated radar reflectivity [dBZ]; -999.99=missing value, -99.99=correct 0	`loutdbz=.TRUE.`
`'vrsim'`	simulated radial wind [m/s]; -999.99=missing value	`loutradwind=.TRUE.`
`'zrobs'`	observed radar reflectivity in dBZ	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.`
`'vrobs'`	observed radial wind. Dealiasing depends on namelist switch `ldealiase_vr_obs`	`lreadmeta_from_netcdf=.TRUE.` and `loutradwind=.TRUE.`
`'vrobserr'`	reflectivity dependent observation error for radial wind.	`lreadmeta_from_netcdf=.TRUE.` and `loutradwind=.TRUE.`
`'qzobs'`	quality flags for observed reflectivity [-]	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.`
`'qvobs'`	quality flags for observed radial wind [-]	`lreadmeta_from_netcdf=.TRUE.` and `loutradwind=.TRUE.`
`'zetsim'`	path integrated attenuation [dB]	`lextdbz=.TRUE.` and `rs_meta(`i`)%itype_refl=1`
`'zepsim'`	twoway-attenuation coefficient [db/km]	`lextdbz=.TRUE.` and `rs_meta(`i`)%itype_refl=1`
`'vrsupsim'`	super-observations of simulated radial wind [m/s] for development purposes	`lreadmeta_from_netcdf=.TRUE.`
`'vrsupobs'`	super-observations of observed radial wind [m/s] for development purposes	`lreadmeta_from_netcdf=.TRUE.`
`'vrsupobserr'`	reflectivity dependent observation error for superobe’d radial wind.	`lreadmeta_from_netcdf=.TRUE.` and `loutradwind=.TRUE.` and `itype_obserr_vr>0`
`'zrsupsim'`	super-observations of simulated reflectivity [dBZ] for development purposes	`lreadmeta_from_netcdf=.TRUE.`
`'zrsupobs'`	super-observations of observed reflectivity [dBZ] for development purposes	`lreadmeta_from_netcdf=.TRUE.`
`'losupsim'`	geographic longitude of super-observation reference points [°]	`lreadmeta_from_netcdf=.TRUE.`
`'lasupsim'`	geographic latitude of super-observation reference points [°]	`lreadmeta_from_netcdf=.TRUE.`

Keyword	Parameter	Dependencies ("if ...")
'vasim'	area-wide simulated radial wind field, does not take into account reflectivity weighting and hydrometeor fallspeed, internally used as a proxy for dealiasing the observations [m/s]	`lreadmeta_from_netcdf=.TRUE.` and `ldealiase_vr_obs=.TRUE.`

Following data formats are supported by choosing the namelist parameter `voldata_format`:

'ascii': simple ASCII output of 3D volume scans according to range, azimut and elevation. There is one file per output time, parameter, station and scan strategy. Does not need any additional libraries, but produces a large amout of data. The file name convention:

`<datasetname>_id-XXXXXX_<scan-id>_YYYYMMDDHHmmss_DDhhmmss_polar.dat`

- `<datasetname>`: can be for example "zrsim" or "zrobs" for simulated and observed reflectivity, respectively.
- `XXXXXX`: the 6-digit WMO station ID, e.g., "01038"
- `<scan-id>`: a string of variable length denoting the scan strategy, e.g. "PPI0080". It consists of a string denoting the general type (here PPI-type volume scan) followed by 4 digits denoting the average fixed angle in one-tenth degrees. Here this is 8.0 degrees denoting the average of all nominal elevation angles. Currently supported are "PPI" scans and DWD's "PRECIP" scans.
- `YYYYMMDDHHmmss`: the model run start time
- `DDhhmmss`: the model forecast time for which the data set is valid

for example

- `zrsim_id-010873_PPI0080_20170710120000_00030000_polar.dat`
- `vrsim_id-010873_PPI0080_20170710120000_00030000_polar.dat`
- `zrobs_id-010873_PRECIP_20170710120000_00030000_polar.dat`
- `lasim_id-010832_PPI0080_20170710120000_00000000_polar.dat`
- `losim_id-010832_PPI0080_20170710120000_00000000_polar.dat`
- `hrsim_id-010832_PPI0080_20170710120000_00000000_polar.dat`

The files consist of:

- one header line starting with '`# ASCII`' describing the content and the relevant parameters of the model run (`inidate_model`, `forecasttime_model` etc.) and the EMVORADO setup,
- a second header line with 3 whitespace-separated integers denoting the number of ranges, azimuts and elevations, (`nra`, `naz`, `nel`) followed by '`|`' and the white-space separated list of the `nel` elevation angles
- one long data column of `nra` ×`naz` ×`nel` floating point numbers, where the first index `nra` varies first, then `naz` and last `nel`.

Example:

```
# ASCII Simul. radar reflectivity [dBz] parameter=zrsim time=20130728143000 ...
 180  360   10 | 0.50 1.50 2.50 3.50 4.50 5.50 8.00 12.00 17.00 25.00
-7.56745E+00
-5.98657E+00
-4.85857E+00
-4.79550E+00
-6.76644E+00
-8.52418E+00
-4.23030E+00
-1.72290E+00
-3.00870E+00
-4.89106E+00
```

...

'ascii-gzip': compressed ASCII (zlib), same content as ASCII, but compression in memory before writing to disk. Requires zlib. Same filename convention than ASCII, but the suffix is `.dat.gz`,

'cdfin': Internally zlib-compressed netcdf-4 files similar (but not exactly equal) to the CDFIN-files from `readbufrx2netcdf` (Section 4.3). The differences are the netcdf4-compression, different radar parameter keywords in the filename (e.g., `'zrsim'` instead of `'z'`) and the internal names of some variables, e.g., reflectivity is called `'reflectivity'` and not `'MHORREO'`.

There is one file per parameter, station and scan strategy. The time range of the file can be configured in namelist `/RADARSIM_PARAMS/` with the parameters `cdfin_tref` and `cdfin_dt`, e.g., one output time per file only or hourly files.

Note that if more than one output time per file is chosen, any "old" CDFIN-files in the output directory should be deleted before starting the model run. Otherwise, the new data will be appended to the old files of same name instead of replacing these old files!

The file name convention is very similar to the CDFIN input files:

cdfin_<datasetname>_id-XXXXXX_<starttime>_<endtime>_<scantype>

- <datasetname>: can be for example "zrsim" or "zrobs" for simulated and observed reflectivity, respectively.
- `XXXXXX`: the 6-digit WMO station ID, e.g., "01038"
- `starttime`: the start of the time range contained in the file, format `YYYYMMDDHHmmss`
- `endtime`: the end of the time range contained in the file, format `YYYYMMDDHHmmss`; can be equal to `starttime`
- `scantype`: keyword for the scan type, either "volscan" or "precipscan"

for example

- `cdfin_zrsim_id-010132_201707101500_201707101500_volscan`
- `cdfin_vrsim_id-010132_201707101500_201707101500_volscan`
- `cdfin_zrobs_id-010132_201707101500_201707101500_precip`
- `cdfin_lasim_id-010132_201707101200_201707101200_volscan`
- `cdfin_losim_id-010132_201707101200_201707101200_volscan`
- `cdfin_hrsim_id-010132_201707101200_201707101200_volscan`

This needs compilation with the additional pre-processor flag `-DNETCDF` and the netcdf-4 library (not netcdf-3!).

The files for simulated reflectivity (`'zrsim'`) and radial wind (`'vrsim'`) may serve as "observations" (simulated truth) in an OSSE run. Simply put these files into the input directory instead of the CDFIN-files mentioned in Section 4.3 and do not take into account quality flags (set `lqcflag=.FALSE.` in namelist `/RADARSIM_PARAMS/`).

The file content is more or less self-explaining by inspecting the files using tools like `ncdump` or `ncview`. The non-trivial file properties are:

- The unlimited dimension is called `'record'`, where one record denotes a PPI-scan. If one volume scan has 12 elevations, there are 12 records per time step. If 3 time steps are in the file, there will be 36 records.
- The same header information as in the `'ascii'` files is contained in the global attribute `'Data_description'`.
- The data vector `ppi_azimuth(records)` contains the nominal start azimuth of each record.
- The data vector `ppi_elevation(records)` contains the nominal elevation of each record.
- The matrix `ray_azimuth(records, n_azimuth)` contains the azimuth value for each ray.
- The matrix `ray_elevation(records, n_azimuth)` contains the elevation value for each ray.

– The global attribute `Data_description` contains the same header line as the ASCII-files, starting with '`# ASCII`' and describing the content and the relevant parameters of the model run (`inidate_model`, `forecasttime_model` etc.) and the EMVORADO setup.

This is the recommended format!

'f90-binary': Fortran90 binary file, same content as ASCII-files, but faster output as a fortran binary byte stream. Files can be converted to ASCII in postprocessing by the Fortran90 program `bin2ascii_cosmofields3D.f90` available from the author, or based on this, own readers can be created. The file name convention is the same as for ASCII-files, except that the suffix is `.bin`:

`<datasetname>_id-XXXXXX_<scan-id>_YYYYMMDDHHmmss_DDhhmmss_polar.bin`

It is possible by namelist paramters in `/RADARSIM_PARAMS/` to thin out elevations as well as observation times from volume scan data files and to restrict the output to a subset of radar observables in order to reduce the output amount for one's special application.

6.1.2 NetCDF feedback files for the KENDA data assimilation system

There is one file per station, per scan strategy and per model forecast, collecting pairs of observations and simulations of all radar observables. This is a generic file format that has been specified for various observation systems (Rhodin, 2012), and EMVORADO implements its radar specific realization. The naming convention is

`fof_radar_id-XXXXXX_<scan-id>_<model-starttime>`

- `XXXXXX`: the 6-digit WMO station ID, e.g., "01038"
- `<scan-id>`: a string of variable length denoting the scan strategy, same as for the ASCII format above.
- `<model-starttime>`: the start time of the model run, format `YYYYMMDDHHmmss`

for example

- `fof_radar_id-010605_PPI0080_20170710120000.nc`
- `fof_radar_id-010908_PPI0080_20170710120000.nc`
- `fof_radar_id-010629_PRECIP_20170710120000.nc`

This needs compilation with the additional pre-processor flags `-DNUDGING` and `-DNETCDF` and linking with netcdf-3 or netcdf-4 library.

The data written to the files can be thinned out in various ways (only certain elevations, only every n'th values, only certain observation times, etc.) guided by namelist parameters in `/RADARSIM_PARAMS/`. There is also an option for so-called super-observations, that is, 2D-averaging within azimut-range-boxes around the points of a quasi-kartesian horizontal grid on the PPI-planes.

The files contain among other meta data the assigned observation error for each datum. Normally the data assimilation software, to which these files are input, assigns own estimates of this observation error afterwards, so that the values in the files might be not relevant. However, for radial wind there are the options `itype_obserr_vr=1/2` which assign observation errors as function of observed reflectivities in form of a ramp function which increases errors towards smaller reflectivities below a reflectivity threshold `ramp_highdbz_obserr_vr`. This is meant to be a kind of quality control to reduce the impact of radial winds from weak echoes (insects, PBL, residual clutter, etc.) in data assimilation. This has been found beneficial in the COSMO model with its specific model biases during nightly very stable clear-sky conditions.

By default, this reduction function is formulated in relative terms, i.e., for larger reflectivities than the threshold the observation error is 1.0, and towards lower reflectivities it grows as a linear ramp at a namelist-specified rate and starting point. In this way, it can be subsequently multiplied in the data assimilation software to the there estimated/defined local absolute observation errors, so that, e.g., Desroziers-estimates can be combined with the weak-echo error increase.

For consistency, the observation error for reflectivity is also set to 1.0 (relative error), same with the radial wind error in case of `itype_obserr_vr=0`.

Note that any "old" radar feedback files in the output directory should be deleted before starting the model run. Otherwise, the new data will be appended to the old files of same name instead of replacing these old files!

6.1.3 Radar composites

(cf. Sections 2.7 and 5.1)

2D composites of simulated and observed (if available) reflectivity of all radar stations are computed by an oversampling-based aggregation technique on a regular rotated lat-lon-grid, based on certain single elevations of each radar station and taking the maximum values in areas of overlap. This is highly configurable with respect to the composite grid and the considered elevation of each radar station. Moreover, there can be several (up to 10) different composites computed in each model run (all on the same grid!). Not only the elevations of PPI-scans can be chosen for each station and composite, but also the DWD precipitation scans are possible. An overall maximum composite over all PPI elevations of each station is possible as well (but very expensive computationally). Composite generation can be switched on by namelist switch `ldo_composite=.TRUE.` and associated sub-parameters in namelist `/RADARSIM_PARAMS/`.

Because the composite grid is currently equal to the COSMO model grid (hardcoded), these composites can either be written as grib-files through the standard COSMO grib output stream (`lfff`-files), see Section 5.1, or as own files produced by EMVORADO (`lcomposite_output = .TRUE.` in `/RADARSIM_PARAMS/`). Only the latter option works in case of asynchroneous radar IO and, in principle, allows the composite grid to be different from the model grid, so it is the recommended option. The EMVORADO composite files follow the file name convention

`dbzcmp_<type>_<model-starttime>_<model-validtime>.grb2`

- `type`: "obs" or "sim" (observations or simulations)
- `model-starttime`: the start time of the model run, format `YYYYMMDDHHmmss`
- `model-validtime`: the actual validity time of the composite, format `YYYYMMDDHHmmss`

The files contain one grib2-record per composite (up to 10).

This needs compilation with the additional pre-processor flag `-DGRIB_API` and DWD's `grib-api` or `eccodes` library distributions with the local DWD grib sample file `DWD_rotated_ll_7km_G_grib2` in subfolder `samples.edzw` of the `ECCODES_SAMPLES_PATH` (DWD = "EDZW").

Note that any "old" composite files in the output directory should be deleted before starting the model run. Otherwise, the new data will be appended to the old files of same name instead of replacing these old files!

6.2 Special values

FOR ALL FORMATS AND RADAR OBSERVABLES: "-999.99" respectively. "-9.99990E+02" are missing values.

In simulated volume scan data, this can happen for radar bins outside the model domain, above the model top or below the model orography (simulated beam blockage). For observed volume scan data, this can also happen because of some clutter filters to remove ground clutter or "blanked" azimut sectors because of known external error sources (e.g., obstacles near the antenna, microwave interference). In composites "-999.99" is also set in areas outside the measuring ranges of the radar stations.

For reflectivity, there is also the special value "-99.99", which denotes a "correct 0". Why? Because reflectivity values ζ are given in the logarithmic dBZ scale, which means

$$\zeta \;=\; 10\log_{10}\left(\frac{\mathrm{Z}}{1\,\mathrm{mm}^6/\mathrm{m}^3}\right)$$

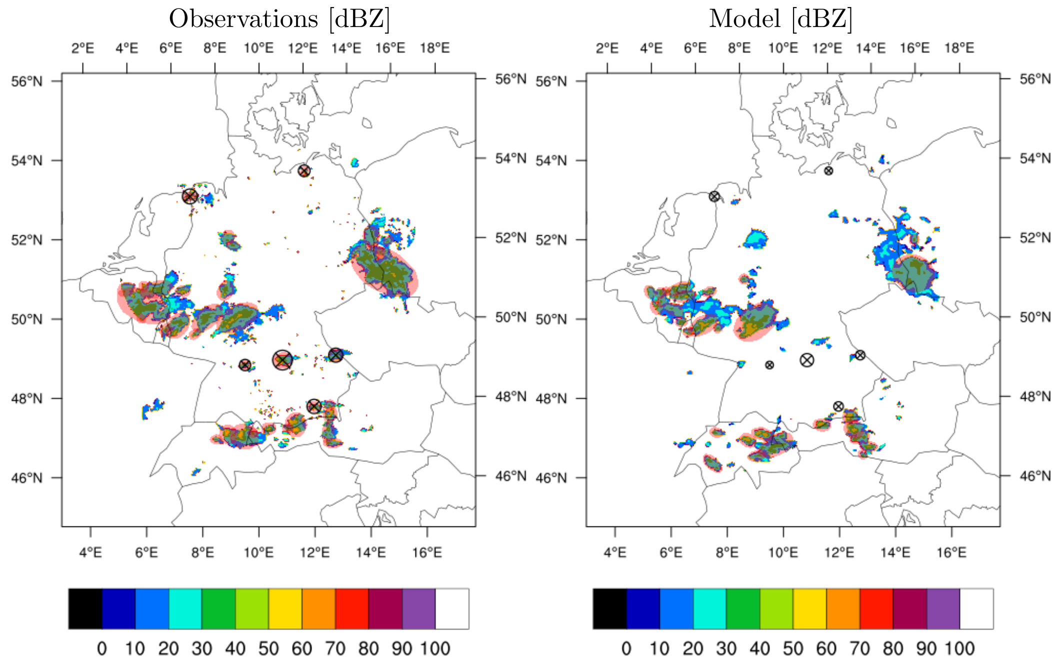


Figure 1: Example for detected elliptical isolated precipitation objects (red semi-transparent ellipses) in observed (left) and simulated (right) composites of reflectivity (colors, dBZ). If observed objects are missing in the model and if these are not “too big” and not “too small” and are isolated from other observed objects (encircled black crosses), artificial “warm bubbles” are added to the model's temperature field in the PBL to trigger the missing convective cells.

where Z is the equivalent radar reflectivity factor, usually given in units of $\mathrm{mm}^6/\mathrm{m}^3$. $Z = 0$ corresponds to $\zeta = -\infty$. Beause $-\infty$ is not good on computers, EMVORADO cuts the values at -90 dBZ ($= 10^{-9}\,\mathrm{mm}^6/\mathrm{m}^3$) and sets smaller values ($< 10^{-9}\,\mathrm{mm}^6/\mathrm{m}^3$) unconditionally to -99.99 dBZ without further ado (except the missing values, which keep their -999.99). Such small values are usually well below the sensitivity range of any radar on the market and can safely be treated as 0.

6.3 READY-files

EMVORADO may optionally write so-called READY-files after each of it's output time steps. These READY-files are useful in an operational context to indicate by their presence, that all files for a certain time have been successfully written to disk and certain post-processing operations may start concurrently to the ongoing model run.

To enable the writing of such READY-files set the namelist switch `lwrite_ready=.TRUE.` in `/RADARSIM_PARAMS/`. The READY-files are named according to the scheme

`READY_EMVORADO_<model-validtime>`

- `model-validtime`: the actual validity time of the composite, format `YYYYMMDDHHmmss`

7 The ”warm bubble generator”

7.1 General description

Simulations of case studies with convection-allowing grid spacings (2 km) are able to produce realistic convective dynamics when the atmospheric profiles of humidity, temperature and wind match the observed conditions. However, these simulations are often not able to capture the process that triggers the convective dynamics, because relevant processes might be active below the model resolution. Convection in idealized or “forced” real case studies is typically initiated by localized perturbations in the temperature and humidity profiles, so-called “warm bubbles”, which are artificially introduced at the beginning of the simulations (Weisman and Klemp, 1982) or any other appropriate time.

Similarly as with such case studies we observe that convection-allowing NWP produces realistic convective dynamics, but may miss the convective trigger in many occasions. Missing the trigger deteriorates not only the prediction but also the assimilation. For example, if the trigger is missed in all ensemble members, the LETKF (and many other ensemble assimilation techniques) cannot recover the convective dynamics in the analysis. This limitation of ensemble methods is caused by the strong non-linearity of convection, so that a convective cell cannot be reconstructed from non-convective members. As a result, NWP might miss large and long-lived convective cells, even when we are certain of their existence from the radar signal.

We propose to use “automatic” warm bubbles to initiate convective cells that are missed in model runs but their existence is certain from radar observations. For data assimilation these are the cycled “first guess” forecasts. While in case studies the researchers manually decide where to introduce warm bubbles, this strategy is not feasible for an operational NWP with, e.g., a 40-member ensemble. We have designed instead an automatic detection/triggering algorithm that decides where to initiate warm bubbles in the model. The detection/triggering algorithm is based on the comparison of radar observation composites with their simulated model counterparts.

The detection/triggering algorithm runs in each ensemble model run, independently of other members. Warm bubbles are triggered in all ensemble members because we expect that bubbles produce realistic convection only in members with the right pre-convective environment. Those more realistic members are closer to the radar observations after the introduction of the bubbles and therefore carry more weight in the subsequent assimilation analysis. The introduction of warm bubbles has thus the potential not only to recover missed convective cells, but also to improve the atmospheric state in the assimilation cycle. The warm bubble analysis is performed in regular time intervals, typically every 10 to 15 minutes. This time span allows for the full early development of convective cells, so that warm bubbles may not be triggered twice if the first was successful.

While similar in the general concept to the traditional Latent Heat Nudging (LHN) method, there are some significant differences:

- LHN adjusts precipitation, not radar reflectivity.
- LHN does this continuously in every model timestep and in the whole domain, but applies rather small temperature and moisture increments continuously.
- LHN is thermodynamically symmetric, because it can also suppress excess precipitation by negative increments.
- The bubble generator can only create missing convective cells in simulation, it cannot destroy wrong cells.
- The bubble generator applies large increments in a short time and waits for the model to react until the next analysis 10 to 15 minutes later.
- In LHN, the increments are directly proportional to the precipitation rate difference (obs-model), whereas the properties of new bubbles (amplitude, size) are pre-selected by the user and do not depend on any reflectivity differences.
- While LHN may be applied to all precipitation events in general, the bubble generator is especially tailored to intense, longlived, isolated and relatively small convective cells, such as rotating supercells. Such events are known to be problematic in LHN.

Conceptually, the bubble generator and LHN may be used together, but this requires further testing and tuning.

7.2 Detecting missing cells in EMVORADO

The bubble generator is active if `ldo_bubbles=.TRUE.` and `lreadmeta_from_netcdf=.TRUE.` in `/RADARSIM_PARAMS/`. The latter namelist also defines the governing parameters for the cell object detection algorithm, which is described below, as well as the properties of the “warm bubbles”, see Table 4. The bubble parameters have similar meaning as corresponding parameters in the COSMO idealized framework (Blahak, 2015). Table 9 shows a typical configuration for a 2-km-scale model. These parameters are equal for all automatic bubbles and are automatically transferred to the hosting model along with position- and time information. See Sections 7.3 and 7.4 for further processing in COSMO and ICON.

The current detection algorithm typically works on the 2D composite reflectivity from radar scans with 0.5° elevation angle (`eleind_for_composite_bub_glob=1` or `rs_meta(`i`)%eleind_for_composite_bub=1`), interpolated to the COSMO-model grid. The compositing method has been described in Section 2.7. For the German radar network, this composite covers all Germany and part of the neighboring countries as shown in Fig. 1. The cell-detection algorithm searches for continuous regions above the threshold Th_1, checking for East-West/North-South- and diagonal connected pixels. We impose two conditions for a continuous region to be defined as a convective feature: it encompasses at least the area A_1, and at least the area A_2 is above a higher threshold Th_2 (cell cores). Once a convective feature is detected we use principal component analysis to find the best ellipse that matches the region. The ellipses are then enlarged by multiplying the axis length by a factor m_{en}, and/or by adding some distance m_{add} to the axis. This option has been introduced to avoid bubbles being triggered too close to existing developing convection. A typical set of parameters is summarized in Table 9 and connected to their respective namelist parameters in `/RADARSIM_PARAMS/`. In this example, the parameters for model and observations are set equal and chosen in a way to detect intense small-scale convective cells. We have also considered the possibility that observation parameters are more restrictive than those for the model, so that we can broadly speak of convective cells in observations and convective features in the model. This depends on the model's ability to simulate very high reflectivities and will differ from model to model.

The triggering algorithm aims to initiate convection in regions where convective cells are observed but there are no convective features in the model. With this idea, the algorithm searches for ellipses identified by the detection algorithm in the observations that do not overlap with ellipses in the model. We also impose that the observation ellipses are small (large axis smaller than some length, e.g. 75 km) The last two conditions were introduced for the few occasions in which the model misses large convective systems, because we think that the assimilation algorithm (LETKF) is more appropriate to deal with them than the warm bubbles.

Warm bubbles are introduced at the location of observed convective cells with no model counterpart, as proposed by the triggering algorithm. The bubbles of type `'cos-instant'` instantaneously increases the temperature, while the bubbles of type `'cos-hrd'` apply a certain heating rate $\dot{T}$ over a certain time interval Δt_{heat}. Optionally, the relative humidity is increased to keep it constant during heating. This is done in a region centered on the ellipse center in the horizontal and at a low height, e.g., $H_{bub} = 2\,\mathrm{km}$ above ground level. The heated region has a fixed ellipsoid shape with certain radii $r_{x,y}$ (e.g., 10 km) for both horizontal main axes, and a radius r_Z (e.g., 2 km) for the vertical axis. The maximum temperature disturbance ΔT (e.g., 3.0 K) is at the center and it decreases towards the ellipsoids borders following a cosine function (Weisman and Klemp, 1982). We have observed that these perturbations are effective in triggering convection while larger perturbations are equally effective but generate too many pressure waves above the tropopause.

The above configuration of parameters for the warm bubble algorithm is rather intrusive, as it produces around five warm bubbles in convective situations every time that the algorithm is called. This aggressive combination is thus appropriate to use warm bubbles as small-scale inflation method. Other tests with more conservative approaches (1 bubbles per call using a less restrictive criteria for convective features in the model) showed that even when warms bubbles were able to recover some convective cells that were missed in the reference runs, the resulting changes in FSS scores were small. We believe that the small changes in FSS may be explained by the fact that this verification metric is mostly determined by large structures that are mostly unaffected by the warm bubbles.

Automatic bubbles might be optionally advected downstream for a certain amount of time Δt_{advect}, to compensate for the effect that it takes time for the bubble to rise above the boundary layer into the tropospheric free flow. The advection velocity is computed as the local average windspeed between two heights H_{lower} and H_{upper}.

Optionally, white noise of a relative level α_{noise} (between 0 and 1) might be superimposed on the bubbles to break their rotational symmetry a bit.

7.3 Implementation in COSMO

Generally, the properties of artificial convection triggers can be defined in two ways in COSMO: automatically via EMVORADO bubble generator or manually via namelist parameters defined in the

Table 9: Typical parameters to configure the warm bubble generator in a 2-km-scale model. Cf. Table 4.

	Unit	Model	Observ.	Param. in `/RADARSIM_PARAMS/`
Detection parameter				
Th_1	dBZ	25	25	threshold_mod(1), threshold_obs(1)
Th_2	dBZ	30	30	threshold_mod(2), threshold_obs(2)
A_1	m^2	135E6	135E6	areamin_mod(1), areamin_obs(1)
A_2	m^2	35E6	35E6	areamin_mod(2), areamin_obs(2)
m_{en}	-	1.0	1.0	mult_dist_mod, mult_dist_obs
m_{add}	m	10000	10000	add_dist_mod, add_dist_obs
Bubble parameter				
Type	"	'cos-hrd'	-	bubble_type ('cos-instant' or 'cos-hrd')
$r_{X,Y}$	m	10000	-	bubble_radx, bubble_rady
r_Z	m	2000	-	bubble_radz
H_{bub}	m	2000	-	bubble_centz
δT	K	3.0	-	bubble_dT ('cos-instant')
$\dot{T}$	K/s	200.0	-	bubble_heatingrate ('cos-hrd')
Δt_{heat}	s	0.04	-	bubble_timespan
If to hold RH constant	-	.TRUE.	-	bubble_holdrhconst
Main axis rotation	°	0.0	-	bubble_rotangle
If to add noise	-	.FALSE.	-	bubble_addnoise
α_{noise}	-	0.1	-	bubble_dT_noise
Δt_{advect}	s	300.0	-	dt_bubble_advect
H_{lower}	m	3000.0	-	zlow_meanwind_bubble_advect
H_{upper}	m	6000.0	-	zup_meanwind_bubble_advect

`/ARTIFCTL/` namelist. Details are described in the COSMO documentation for idealized simulations Blahak (2015), although the part for the artificial convection triggers might also be applied in real-case simulations. Convection triggers might be local disturbances of the atmospheric and/or soil initial state (T, moisture), or local (in space and time) heating/moistening rate disturbances in atmosphere and/or soil.

In general, the parameters for the convection triggers in `/ARTIFCTL/` are lists for up to `ntempdist_max` disturbances, the first element defines the first bubble, the second element the second and so on, and there is a master switch list `ltempdist` for each disturbance. For example, if `ltempdist` =.TRUE., FALSE., ...), only the first bubble in all the parameter lists will be activated. The type for each disturbance is defined using a certain name, e.g., `'cos'` ($\cos^2$ instantaneous bubble), `'cos-hrd'` ($\cos^2$ heating rate) or `'cos-soil'` ($\cos^2$ disturbance in the soil), `'hotspot-soil'`. `'cos-instant'` equals `'cos'`, but is coded internally as a 1-timestep heatingrate for technical reasons. If the COSMO binary is compiled with EMVORADO, these disturbances can be also used in real cases by setting `ldo_bubbles_manual=.TRUE.` in `/RADARSIM_PARAMS/`.

The bubble informations from the automatic bubble generator are inserted into the above `/ARTIFCTL/` disturbance lists starting at the position i of the first `ltempdist`(i) =.FALSE. element, i.e., after any "manual" bubbles. Thus, manual and automatic bubbles may be combined.

For automatic bubbles, only the types `'cos-instant'` or `'cos-hrd'` can be chosen in the EMVORADO namelist. Other disturbance types available in `/ARTIFCTL/` would not make sense in this context. The bubble properties coming from EMVORADO are equal for all automatic bubbles and are automatically filled into the above /ARTIFCTL/ lists at the appropriate model time(s). This information is evaluated in each model time step by the COSMO procedure set_artif_heatrate_dist() to superimpose disturbances at the desired locations and times.

7.4 Implementation in ICON

While COSMO's flexible framework for idealized test cases allowed to use it's part for idealized convection triggers also in real-case simulations, this is currently not possible in ICON. Here, an own trigger procedure set_artif_heatrate_dist() from module mo_emvorado_warmbubbles.f90 is evaluated immediately after the microphysics part of the time stepping, alongside the Latent Heat Nudging with it's T and RH increments.

Only bubbles of types 'cos-instant' or 'cos-hrd' are implemented.

Random noise on the bubbles is not yet implemented in ICON, so that the EMVORADO parameters bubble_addnoise and bubble_dT_noise have no effect.

8 For developers

8.1 Implementing EMVORADO into hosting NWP models

EMVORADO itself is a collection of Fortran2003 modules, and each module name starts with the keyword `radar_`. There is also a Fortran90 `inlcude`-file named `radar_elevations_precipscan.incf` which contains the nominal elevation values as function of azimuth index for the horizon-following "precipitation scans" of DWD for each of the German radar stations (station ID's). The code in this file is the core of a `SELECT CASE (rs_meta(`*I*`)%station_id)` statement and is `#include`'d into subroutine `get_elarr_precipscan()` of `radar_obs_meta_list.f90`. The code for this file has been created using the script `format_precipscan_f90` of U. Blahak and is based on the INPUT text file `elevations_precipscan.txt`, which has been provided by DWD's radar applications unit.

Important for the implementation/coupling of EMVORADO in a numerical NWP model are

- several initialization routines from `radar_interface.f90` which are called once during model initialization from the numerical model.
- the generic organizational subroutine `organize_radar()` in module `radar_src.f90`. This is the top-layer interface for the radar simulation in each model timestep and is directly called once for further initializations ('init' stage) and in the model timeloop ('compute' stage).
- `radar_mie_iface_cosmo.f90` and `radar_mie_meltdegree.f90`: interface procedures to compute grid point values (reflectivity, hydrometeor fall speed), at the moment only for the COSMO- and ICON cloud microphysics schemes, taking into account melting hydrometeors. These interface procedures are associated with step 1 of EMVORADO.
- `radar_namelist_read.f90` contains the subroutine `input_radarnamelist()` to read the `/RADARSIM_PARAMS/` namelist(s).
- `radar_src.f90` also contains the generic interface routines for step 2 of the operator, which are directly called from `organize_radar()`.
  Further, this module includes the code for computing `superobservations` and for output of `volume data`, `feedback files` and `composites` (grib2).
- `radar_obs_meta_read.f90` contains the code for reading the radar station meta data from observation files, if any are used.
- `radar_obs_meta_list.f90` contains the background meta data lists for each known "country" (`icountry`) and radar station.
- `radar_obs_data_read.f90` contains the code for `reading observational data`.
- There are model-specific procedures (interpolation to/from model grid, time housekeeping, parallelization, etc.) in the module `radar_interface.f90`. This module is a two-way connection and generally differs from model to model: On the one hand, it provides the specific code for some generic model-related procedures used in `radar_src.f90` associated with the model grid (interpolation), the time-housekeeping, the profiling ("timing") and the MPI-parallelization. It may use specific routines from the model itself and connects EMVORADO with the model fields and some global model parameters. On the other hand, it provides an operator-specific initialization routine and some parameters to be called/used in the hosting model.
- For ICON, the `radar_interface.f90` module has been split in two modules, named `radar_interface.f90` and `radar_mpi_init_icon.f90`.
- In `radar_data_namelist.f90`, the name and path for the EMVORADO namelist file can be adapted from `INPUT_RADARSIM / NAMELIST_EMVORADO` to differing naming conventions in other models. Similarly, the name and path to the namelist control output (`YUSPECIF_RADAR / nml.emvorado.log`) can be adapted.

The actual implementation of the calls to the top-level procedures of these modules depends on the hosting numerical model.

8.2 Implementation documentation for COSMO

This section describes how the general implementation aspects described in the last section 8.1 are actually implemented in the COSMO-model. Here:

- Calls to several initialization routines from `radar_interface.f90` from the main program `lmorg`. These are described below in Section 8.2.1.

- Calls of the generic organizational subroutine `organize_radar()` from module `radar_src.f90` from `lmorg` for the 'init' and 'compute' stages, also described below in Section 8.2.1.

- Specific calls to interface routines from `radar_mie_iface_cosmo.f90` and `radar_mie_meltdegree.f90` or step 1 of the operator described below in Section 8.2.2.

- Traditional gridpoint output using the same interface routines to reflectivity and hydrometeor fallspeed than for step 1 of the operator (Section 8.2.4).

Along with the calling sequences, a rough description of the specific tasks of the interface routines is also given in the next sections.

8.2.1 The top-level interface to COSMO

At model initialization stage, two EMVORADO-specific sections are added for initializing its MPI-parallelization, the optional asynchroneous radar-IO, reading the `/RADARSIM_PARAMS/` namelist, and connecting the prognostic model fields with correscponding pointers inside EMVORADO.

The computing- and output-stages (steps 1 and 2) are performed in every timestep by a call to `organize_radar('compute')` after the update of the model variables by physics and dynamics. EMVORADO uses timestep "nnow" of the model fields, consistent with the "normal" grib output.

In the following, we give a schematic of the calling sequence of the top-layer interface to EMVORADO in the main program `lmorg` for the initialization stage and the time-loop, taking into account the optional asynchroneous radar-IO if `nprocio_radar` $>$ `0` is chosen in namelist `/RUNCTL/`. The blue color highlights the additional EMVORADO-related code blocks, which are enclosed by `#ifdef RADARFWO` in the COSMO source code. Black indicates "normal" COSMO code for better orientation:

CALL organize_setup

- split **icomm_compute** from **icomm_world** and define **lcompute_pe**, **icomm_cart** (**CALL init_procgrid** from environment.f90)
- split **icomm_computeio** from **icomm_world**, so that **icomm_computeio** = **icomm_compute** + **icomm_asynio**
- if **nprocio_radar** > **0**, additional PEs for asynchroneous radar-IO are allocated at the end of **icomm_world**. These are not part of **icomm_computeio**.

...

CALL organize_dynamics('input')

...

CALL organize_physics('input')

...

CALL get_model_config_for_radar

- because of grid point reflectivity output

IF *luse_radarfwo* **THEN**

  CALL prep_domains_radar
  CALL prep_domains_radar_nml
  CALL init_radar_mpi

  - First initialization step of EMVORADO: internal MPI
  - If asynchroneous radar-IO (**nprocio_radar** > **0**):
    - split **icomm_radario** from **icomm_world**, so that **icomm_radario** = **icomm_world** - **icomm_compute** - **icomm_asynio**. This is the communicator for the extra radar-IO-PEs. Sets **lradario_pe** = **.TRUE.** on **icomm_radario**.
    - split **icomm_radar** from **icomm_world**, so that **icomm_radar** = **icomm_world** - **icomm_asynio**. This is the common communicator of the compute-PEs and the radar-IO-Pes and is used for data exchange between the two. Also, sets **lradar_pe** = **.TRUE.** on **icomm_radar**.
    - as a result, **icomm_radar** = **icomm_compute** + **icomm_radario** and **lradar_pe** = **lcompute_pe or lradario_pe**.
  - If synchroneous radar-IO (**nprocio_radar** = **0**):
    - set **icomm_radar** = **icomm_compute** and **icomm_radario** = **icomm_compute**, sets **lradar_pe** = **.TRUE.** and **lradario_pe** = **.TRUE.** on **icomm_compute**

ELSE

  CALL init_radar_mpi_light
  lradar_pe = .FALSE.
  lradario_pe = .FALSE.

  - Necessary because of possible grid-point reflectivity output

END IF

CALL organize_data('input')

- read namelists for IO, also for lartif_data

...

CALL organize_data('init')

- setup dbz_meta-structure for grid point dBZ-output,
- pre-compute needed MIE-lookup tables by **CALL init_lookup_mie**,
- **CALL mpe_io_init**: split **icomm_compute** from **icomm_computeio** instead of **icomm_-world**

...

IF *lcompute_pe* **THEN**

- allocate model fields
- compute constant fields (metrical `terms`, srcformrlat, srcformrlon)
- read initial data

END IF

...

IF *lradar_pe* **THEN**

IF *luse_radarfwo* **THEN**

CALL organize_radar('init')

- Second initialization step of EMVORADO: namelist and pointers to the COSMO model fields
  - read and distribute radar namelist `/RADARSIM_PARAMS/` to all PEs in **icomm_radar**,
  - this requires reading of the header information from all radar observation input files if namelist parameter lread_meta_from_netcdf = .TRUE.,
  - check radar namelist settings and compute additional parameters,
  - control output of radar namelist to file `YUSPECIF_RADAR`,
  - **CALL get_model_config_for_radar**,
  - setup radar-composite metadata,
  - pre-compute needed MIE lookup tables in parallel over all PEs in icomm_radar,
  - **CALL get_model_hydrometeors**,
  - **CALL get_model_variables**,
  - **CALL alloc_aux_model_variables**.

CALL crosscheck_domains_radar_nml

ELSE

CALL get_model_config_for_radar

- This is necessary for the separate grid-point reflectivity output via `/GRIBOUT/` namelist(s) in case of `luse_radarfwo = .FALSE.`, i.e. if the full EMVORADO is not used.

END IF

END IF

...

IF *lcompute_pe* **THEN**

timeloop: DO *ntstep=1,nstop*

- Integrate model for one timestep

...

CALL organize_radar('compute')

- Steps 1 and 2 of EMVORADO for each radar station at each individual output time.
- If **nprocio_radar** $>$ **0**:
  - sends the simulated radar data from the compute-PEs to the radar-IO-PEs via **icomm_radar** and exits.
- If **nprocio_radar** $=$ **0**:
  - collects (re-distributes) the simulated radar data on one compute-PE per station
  - reads radar observations if needed
  - computes radar composites if desired
  - detects the locations for artificial warm bubbles if desired
  - outputs radar volume data and/or feedback files for data assimilation

...

IF *output-time* **THEN**

IF *lasyn_io* **THEN**

- Send model field data to the asynchroneous model output PEs (non-blocking) and continue

ELSE

- Do the output of model fields on PE 0

END IF

END IF

END DO timeloop

ELSE IF *lradario_pe* **and** *nprocio_radar > 0* **THEN**

timeloop_2: DO *ntstep=1,nstop*

CALL organize_radar('compute')

- This is a call on pure radar output PEs, so **CALL organize_radar** just waits to receive simulated radar data from the compute-PEs via `icomm_radar`, one output-PE per station.
- Receives from the corresponding **CALL organize_radar** in the model timeloop above, therefore the exact same time stepping is needed.
- Does not use any model fields, just needs to know about the actual model time,
- reads radar observations if needed,
- computes radar composites if desired,
- detects the locations for artificial warm bubbles if desired,
- outputs radar volume data and/or feedback files for data assimilation to `ydirradarout`.

END DO timeloop_2

ELSE These are the asynchroneous model output PEs

- Listen to receive and output model data records from the compute-PEs and wait for the next output messages,
- if model time is finished, stop listening.

END IF

8.2.2 Implementation of step 1: grid point values of reflectivity and hydrometeor terminal fallspeed

The corresponding subroutines `calc_dbz_vec_modelgrid()`, `calc_fallspeed_vec_modelgrid()` and `init_lookup_mie()` from `radar_mie_iface_cosmo.f90` are called from step 2-routines from within EMVORADO, when reflectivity and/or hydrometeor fallspeed on the model grid is needed to be interpolated to the radar bins.

8.2.3 Implementation of step 2: volume scans of reflectivity and radial velocity

All corresponding subroutines regarding interpolation from model grid to radar bins are contained in radar_src.f90 and are called from the top-level EMVORADO routine organize_radar() during the 'compute' stage, depending on the general setup of EMVORADO. organize_radar() is called from the main program as described in Section 8.2.1. radar_src.f90 also contains the procedures for the different kinds of radar data output,

If observation files are used, the code for reading the station meta data and the actual radar observables from the files is in radar_obs_meta_read.f90 and radar_obs_data_read.f90. radar_obs_meta_list.f90 holds procedures for initial initialization of the meta data type rs_meta(i) depending on icountry, as well as background meta data lists for each known radar station for cross-checking.

8.2.4 Implementation of "traditional" grid point reflectivity output

The subroutines calc_dbz_vec(), calc_fallspeed_vec_modelgrid() and init_lookup_mie() are also called at other places in COSMO in case of "traditional" grid point reflectivity and fallspeed output via /GRIBOUT/ namelist. This is needed if at least one of the shortnames DBZ (yvarml, yvarpl, or yvarzl), DBZ_850 (yvarml), DBZ_CMAX (yvarml) and DBZ_CTMAX (yvarml) has been specified in the /GRIBOUT/ namelists.

- init_lookup_mie() in organize_data.f90, section 'start', to prepare Mie-lookup tables if required by the choice dbz%itype_refl=1 in namelist /GRIBOUT/
- calc_dbz_vec_modelgrid() in calc_tracks.f90
- calc_dbz_vec_modelgrid() and calc_fallspeed_vec_modelgrid() in src_output.f90

By using these new subroutines at these points, the grid point reflectivity output has been extended by options for the new Mie- and Rayleight-Oguchi-methods dbz%itype_refl=1 and dbz%itype_refl=3 from EMVORADO, as already mentioned in Section 2. For backwards compatibility, the previous method from pp_utilities.f90 is available as option dbz%itype_refl=4.

For developers, hydrometeor fallspeed is available via the shortname DUMMY_1 and the Mie two-way attenuation coefficient (dbz%itype_refl=1) via the shortname DUMMY_2.

8.3 Implementation documentation for ICON

TODO

8.4 Recipe to implement new namelist parameters into /RADARSIM_PARAMS/

- Add declaration statement to module radar_data_namelist.f90.
- Add corresponding component to declaration of derived type glob_nml_type in radar_data_namelist.f90.
- Add corresponding code line to subroutines store_domain_radar_nml and switch_to_domain_radar_nml in radar_data_namelist.f90. Existing lines for other namelist parameters may serve as an orientation.
- Does your parameter have to be different for different model domains? If yes, add a corresponding check to subroutine crosscheck_domains_radar_nml in radar_data_namelist.f90.
- Add default value, some cross-checks (if needed) and global MPI-distribution to subroutine input_radarnamelist() in module radar_namelist_read.f90. Note that the namelist is read from file several times during the process, because the default of some parameters depends on the actual setting of other parameters. But if this is not the case for your new parameter, simply add it to the first namelist reading pass.
- Add control output for file YUSPECIF_RADAR / nml.emvorado.log near the end of subroutine input_radarsim.f90.

8.5 Recipe to implement new type components into `rs_meta`

The namelist parameter type instance `rs_meta` is of derived type `radar_meta_type`, which is declared in module `radar_data.f90`.

- Add new component to declaration of derived type `radar_meta_type` in `radar_data.f90`.
- Add corresponding copy line to subroutines `rsm_multitime2onetime` and `rsm_onetime2multitime` from `radar_data.f90`. If your new component is an array where any of the dimensions is `nobstimes_max`, only the first element along this dimension is retained.
- In module `radar_parallel_utilities.f90`, extent the derived MPI-type `mpi_radar_meta_typ` in subroutine `def_mpi_radar_meta_type` by the size of your new component. The existing code shows you how to do this: There are different blocks for the different basic Fortran90 types, and you have to add the size (number of elements, not number of bytes!) of your new component (might be a scalar with size 1 or an array) to the ` ``blocklengths'' ` - counter before the corresponding `CALL MPI_TYPE_EXTENT()` statement.
- If your new component is an array with `nobstimes_max` as any of the dimensions, use the local INPUT variable `nobstimes_max_loc` in the ` ``blocklengths'' ` - statement instead.
- In module `radar_namelist_read.f90`, add a control output line for file `YUSPECIF_RADAR` / `nml.emvorado.log` in the subroutine `ctrl_output_rsmeta_nuspec_fwo()`.

8.6 Recipe to implement new type components into `dbz_meta`

The namelist parameter type instance `dbz_meta` is of derived type `dbzcalc_params`, which is declared in module `radar_data.f90`.

- Add new component to declaration of derived type `dbzcalc_params` in `radar_data.f90`.
- Add corresponding background default value to the declaration block of
  `TYPE(dbzcalc_params), PARAMETER :: dbz_namlst_d`
  in module `radar_data.f90`.
- In module `radar_parallel_utilities.f90`, extent the derived MPI-type `mpi_dbzcalc_params_typ` in subroutine `def_mpi_dbzcalc_params_type` by the size of your new component. The existing code shows you how to do this: There are different blocks for the different basic Fortran90 types, and you have to add the size (number of elements, not number of bytes!) of your new component (might be a scalar with size 1 or an array) to the ` ``blocklengths'' ` - counter before the corresponding `CALL MPI_TYPE_EXTENT()` statement.
- In module `radar_namelist_read.f90`, add a control output line for file `YUSPECIF_RADAR` / `nml.emvorado.log` in the subroutine `ctrl_output_dbzmeta_nuspec_fwo()`.

8.7 Recipe to implement a new "country"

In the context of EMVORADO, a "country" denotes a set of meta data defaults for a group of similar radars. This set serves as a background default for the station meta data `rs_meta` and, as mentioned earlier in Table 4, may be chosen either globally by the parameter `icountry` or individually for each station by `rs_meta(i)%icountry`. The current choice of countries is described in Table 4.

In order to add a new option to `icountry`, the following steps are necessary:

- Add a background metadata list with allowed default scan strategies in `radar_obs_meta_list.f90`:
  - `get_meta_proto_<newcountry>()`
  - `get_meta_network_<newcountry>()`
  - `get_meta_network_all()`
- Add a new metadata reader in the subroutine `read_meta_info_all()` in `radar_obs_meta_read.f90`:
  - `get_metadata_from_<newcountry>()`
- Add a new data reader in the subroutine `read_obs_rad()` in `radar_obs_data_read.f90`:

  - `read_field_obs_<newcountry>()`
- Add new declarations in `radar_data_namelist.f90`:
  - `nradsta_<newcountry>`
  - `rs_meta_<newcountry>_proto`
  - `rs_meta_<newcountry>(nradsta_<newcountry>)`

References

Blahak, U., 2008: An approximation to the effective beam weighting function for scanning meteorological radars with axisymmetric antenna pattern, *J. Atmos. Ocean. Tech.*, **25**, 1182–1196.

Blahak, U., 2015: *Simulating Idealized Cases with the COSMO model*, Consortium for Smallscale Modeling (COSMO), URL `http://www.cosmo-model.org/content/model/documentation/core/artif_docu.pdf`.

Blahak, U., 2016: RADAR_MIE_LM and RADAR_MIELIB — calculation of radar reflectivity from model output, *Technical Report 28*, Consortium for Small Scale Modeling (COSMO), URL `http://www.cosmo-model.org/content/model/documentation/techReports/docs/techReport28.pdf`.

Cressman, G. P., 1959: An operational objective analysis system, *Mon. Wea. Rev.*, **87**, 367–374.

Jerger, D., 2014: *Radar Forward Operator for Verification of Cloud Resolving Simulations within the COSMO-Model*, Dissertation, IMK-TRO, Department of Physics, Karlsruhe Institute of Technology, URL `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000038411`.

Rhodin, A., 2012: *Feedback File Definition*, Consortium for Smallscale Modeling (COSMO), URL `http://www.cosmo-model.org/content/model/documentation/core/CosmoFeedbackFileDefinition.pdf`.

Schättler, U., G. Doms and C. Schraff, 2019: *A Description of the Nonhydrostatic Regional COSMO-Model. Part VII: User's Guide*, Consortium for Smallscale Modeling (COSMO), URL `http://www.cosmo-model.org/content/model/documentation/core/cosmo_userguide_5.05.pdf`.

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.

Zeng, Y., 2013: *Efficient Radar Forward Operator for Operational Data Assimilation within the COSMO-model*, Dissertation, IMK-TRO, Department of Physics, Karlsruhe Institute of Technology, URL `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000036921`.

Zeng, Y., U. Blahak and D. Jerger, 2016: An efficient modular volume-scanning radar forward operator for NWP models: description and coupling to the COSMO model, *Quart. J. Roy. Met. Soc.*, **142**, 3234–3256, URL `http://onlinelibrary.wiley.com/doi/10.1002/qj.2904/abstract`.

Zeng, Y., U. Blahak, M. Neuper and D. Epperlein, 2014: Radar beam tracing methods based on atmospheric refractive index, *J. Atmos. Ocean. Tech.*, **31**, 2650–2670.