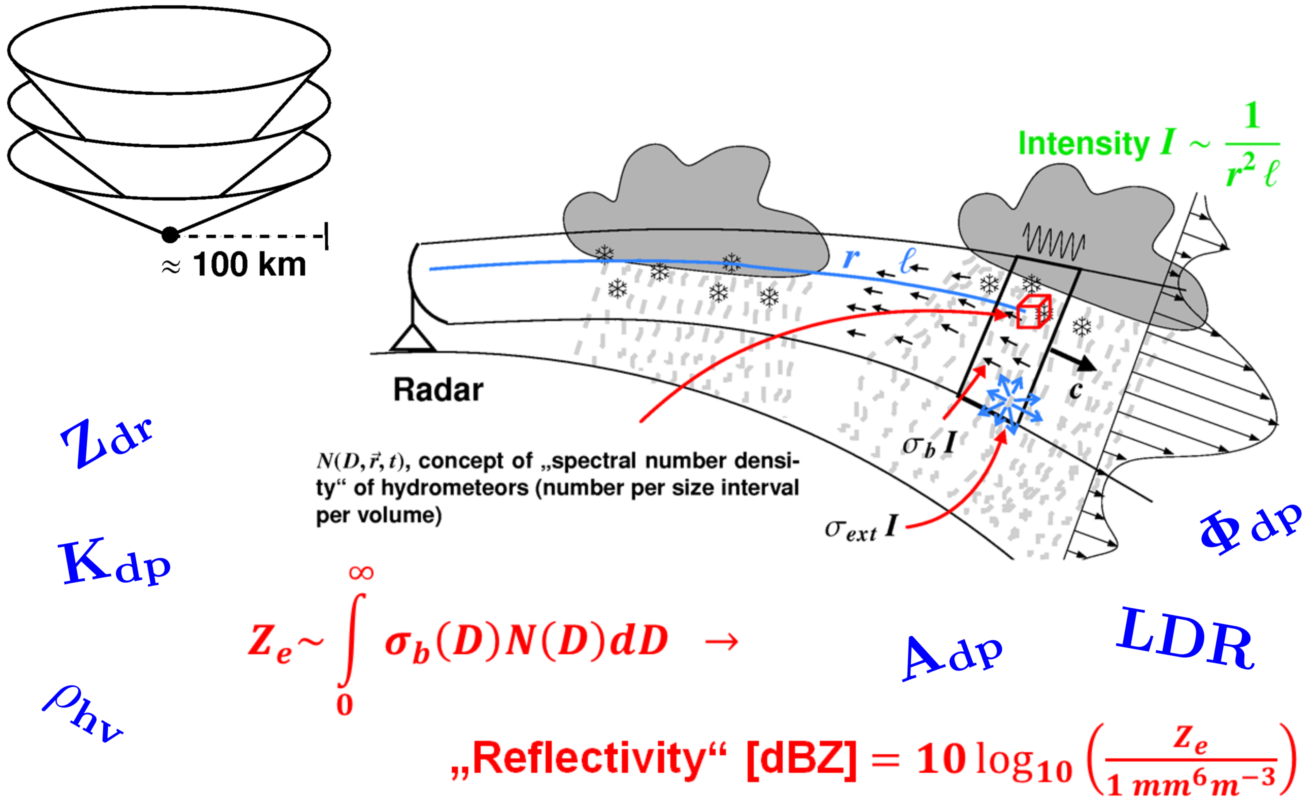
EMVORADO — Efficient Modular VOlume scan RADar Operator

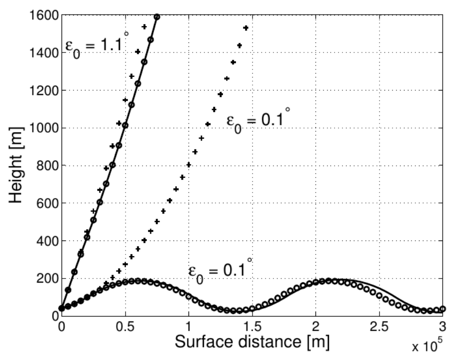
— A User's Guide —

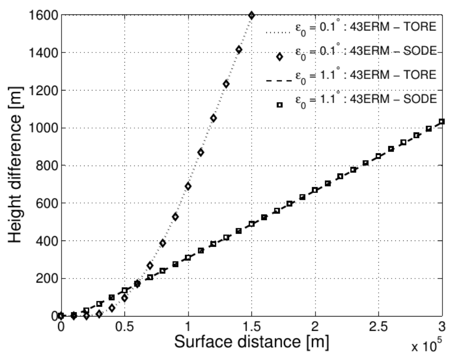
Ulrich Blahak[1,2], Alberto de Lozar[2], Jana Mendrok[2]

February 27, 2025

(Contributors to EMVORADO development see imprint on next page)


≈ 100 km
Radar
Intensity $I \sim \frac{1}{r^2 \ell}$
r
ℓ
c
$\sigma_b I$
$\sigma_{ext} I$
$N(D,\vec{r},t)$, concept of „spectral number density" of hydrometeors (number per size interval per volume)
Z_{dr}
K_{dp}
ρ_{hv}
Φ_{dp}
A_{dp}
LDR
$Z_e \sim \int_0^\infty \sigma_b(D) N(D) dD \rightarrow$
„Reflectivity" [dBZ] $= 10 \log_{10}\left(\frac{Z_e}{1\,mm^6 m^{-3}}\right)$





Created February 27, 2025 using LaTeX and KOMA-Script

Corresponding Author:

Ulrich Blahak
Deutscher Wetterdienst / German Weather Service (DWD)
Frankfurter Str. 135
63067 Offenbach Germany

Email: ulrich.blahak@dwd.de

Contributors to EMVORADO development:

Ulrich Blahak[1,2], Yuefei Zeng[1,2,3], Dorit Jerger[1], Daniel Leuenberger[4], Axel Seifert[1,2], Jana Mendrok[2], Jeffrey Snyder[5], Jacob Carlin[5,6], Alberto de Lozar[2]

[1] Institute for Meteorology and Climate Research (IMK), Karlsruhe Institute of Technology (KIT)
[2] German Weather Service (DWD), Offenbach, Germany
[3] Meteorological Institute Munich (MIM), Ludwig-Maximilians-University Munich (LMU), Germany
[4] MeteoSwiss, Zurich, Switzerland
[5] National Severe Storms Laboratory (NOAA NSSL), Norman, Oklahoma, USA
[6] Cooperative Institute for Severe and High-Impact Weather Research and Operations, University of Oklahoma, Norman, Oklahoma, USA

Where to get this User's Guide?

- COSMO-Webpage:
  `http://www.cosmo-model.org/content/model/documentation/core/emvorado_userguide_dualpol.pdf`
- For ICON-NWP users:
  `git@gitlab.dkrz.de:icon/icon-nwp.git`
  `<icon-repo-dir>/externals/emvorado/DOC/TEX/emvorado_userguide.pdf`
- For EMVORADO standalone offline-framework users:
  `git@gitlab.dkrz.de:dace_projects/emvorado-offline.git`
  `<emvorado-offline-dir>/DOC/TEX/emvorado_userguide.pdf`
- For EMVORADO core developers:
  `git@gitlab.dkrz.de:dace_projects/emvorado-package.git`
  `<emvorado-package-dir>/DOC/TEX/emvorado_userguide.pdf`

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 12.
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 12.
19.9.2019	Ulrich Blahak	New namelist switch `labort_if_problems_obsfiles` in `/RADARSIM_PARAMS/`. Inserted into Table 12.
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 12. 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 12. Updated section 5.1.7.
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 12. 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 12. 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 7 and Subsection 7.4.
18.09.2020	Ulrich Blahak	For ICON only: Change default for output grid of composites from COSMO-DE to COSMO-D2. Update Table 12.
29.03.2021	Ulrich Blahak	New reader for DWD's flavor of ODIM hdf5 radar observation files. Updated Section 4.3
29.03.2021	Ulrich Blahak	New optional output format grib2 for simulated and observed reflectivity volume scans. According updates to Section 6.1.1.
29.03.2021	Ulrich Blahak	New option for multi-moment multi-volume ouput (`voldata_format='cdfin-mulmom'` or `'grib2-mulmom'`). According updates to Section 6.1.1.
29.03.2021	Ulrich Blahak	New namelist parameter `dbz_meta_glob` of type `t_dbzcalc_params`, replacing the previous `itype_refl_glob` and `llookup_mie_glob` and offering the possibility to specify global background values of all type components for all radars.
29.03.2021	Ulrich Blahak	Eliminated `itype_refl_glob` and `llookup_mie_glob`

Date	**Name**	**Change**
29.03.2021	Ulrich Blahak	New type components in `t_dbzcalc_params` of previously hardcoded parameters for EMVORADOs melting sheme. Updates of Table 14 in Section 5.1.5.
29.03.2021	Ulrich Blahak	Optimized lookup table generation: optimized order of computations with re-cycling of Mie-results and better parallelization. New associated namelist parameters `itype_mpipar_lookupgen`, `pe_start_lookupgen` and `pe_end_lookupgen`
29.03.2021	Jana Mendrok	**Polarimetric upgrade of EMVORADO:** introduce polarimetric radar moments in EMVORADO: Z_{dr}, A_{dp}, ρ_{hv}, K_{dp}, Φ_{dp} and LDR. New options `dbz_meta_glob%itype_refl=5` (T-matrix for spheroidal particles) and `6` (T-matrix but assuming spherical particles; option for cross-comparison with classical Mie-scattering `dbz_meta_glob%itype_refl=1`). No composites and fof-files yet of these new variables. Also no grib2-ouput due to missing shortNames. Processing of polarimetric variables on top of Z_h and A_h is triggered by `loutdbz=.TRUE.` and `loutpolstd=.TRUE.` or `loutpolall=.TRUE.`.
29.03.2021	Ulrich Blahak	New option for the diagnostic melt fraction as function of particle size and temperature and a new parameter `dbz_meta_glob%itype_Dref_fmelt` to select the option (“2”) or use the default scheme (“1”). The new option is somewhat simplified and uses fixed reference diameters in the meltdegree formula instead of some mean PSD size. This greatly enhances the efficiency of polarimetric T-matrix lookup table generation, but also changes the results in the melting layer. Updates of Table 14 in Section 5.1.5.
29.03.2021	Ulrich Blahak	New possibility to have several output streams (formats, time intervals, file names with flexible file name patterns) for the same volume data. New derived type `t_voldata_ostream` defining new namelist parameter list `voldata_ostream(1:noutstreams_max)`
29.03.2021	Ulrich Blahak	With the introduction of `voldata_ostream(1:noutstreams_max)`, eliminated the old namelist parameters `voldata_format`, `voldata_output_list`, `cdfin_dt` and `cdfin_tref`, which are now components of the derived type `voldata_ostream` and slightly renamed.
29.03.2021	Ulrich Blahak	New namelist parameter `ysubdircomp` to explicitly specify the output subdirectory under `ydirradarout` of composites in grib2. If it starts with a “/”, it is taken as an absolute path.
29.03.2021	Ulrich Blahak	New namelist parameter `composite_file_pattern` to explicitly specify the file pattern of composite files, with the help of (optional) keys.
29.03.2021	Ulrich Blahak	New namelist parameter `ysubdirfof` to explicitly specify the output subdirectory under `ydirradarout` of fof files. If it starts with a “/”, it is taken as an absolute path.
29.03.2021	Ulrich Blahak	New namelist parameter `ydir_ready_write` to explicitly specify the output directory for READY files.
29.03.2021	Ulrich Blahak	New namelist parameter `ready_file_pattern` to explicitly specify the file pattern of READY files, with the help of (optional) keys.
15.04.2021	Ulrich Blahak	New namelist parameter `llookup_interp_mode_dualpol` to switch to a consistent interpolation technique for table lookup among all polarimetric moments (cubic interpolation of linear values). Avoids interpolation artifacts, e.g., values of $\rho_{hv} > 1$.

Date	**Name**	**Change**
15.04.2021	Ulrich Blahak	New namelist parameter `lcalc_dbz_on_radarbins` for new option to switch the order of interpolating model state variables to radarbins and computing polarimetric moments. If `.TRUE.`, first interpolate the model state and then compute radar moments. Only implemented for 4/3-earth beam propagation (`lonline=.FALSE.`).
11.08.2021	Ulrich Blahak	New type component `voldata_ostream%grib2_packingtype` of namelist parameter `voldata_ostream` to explicitly define the grib2 packing type. Up to now, the packing type `grid_ccsds` was hard-coded.
04.12.2021	Ulrich Blahak	In obs data mode (`lreadmeta_from_netcdf=.TRUE.`), new possibility to add further stations in the namelist-driven re-definition step of station metadata. Up to now, it was only possible to change meta-data of existing stations in obs files. To enable correct setting of obs times, set the new `rs_meta(`*i*`)%lobstimes_ovwrt_recalc=.TRUE.` for every new station *i*.
04.12.2021	Ulrich Blahak	In obs data mode, enable to extend/change the obs times for stations present in obs files. For this, new type component `rs_meta(`*i*`)%lobstimes_ovwrt_recalc` (logical) of namelist parameter `rs_meta` to trigger re-calculation of the series of output times for station *i* where `rs_meta%dt_obs` and/or `rs_meta%nobs_times` have been changed by namelist after obs metadata reading. If no obs are available for a certain time, the production of obs composites and the bubble generator are skipped.
04.03.2022	Ulrich Blahak	In obs data mode, enable to process OPERA hdf5 files (file names starting with `T_PA`) of Switzerland and Belgium.
04.03.2022	Ulrich Blahak	In obs data mode, enable to switch off the writing of radial winds and reflectivity to the feedback files for single radar stations. For this, new type components `rs_meta(`*i*`)%lvrad_to_fdbk` and `rs_meta(`*i*`)%ldbzh_to_fdbk` (logicals) of namelist parameter `rs_meta`. And because for some countries the Nyquist velocity alternates between different elevations and/or times, radial winds might or might not be usable for assimilation. The new type component `rs_meta(`*i*`)%vnyq_min_for_vr_active_fdbk` allows to set a threshold Nyquist velocity above which radial winds are set `active` in feedback files. Otherwise `passive`.
19.03.2022	Ulrich Blahak	In obs data mode, enable to process OPERA hdf5 files of France, Denmark, Netherlands, Poland, Czech Republik and Germany.
03.08.2022	Ulrich Blahak	Allow possibility to have different composite grids for the warm bubble generator and the “normal” composite output. New namelist parameters `comp_meta_bub`, `lcomposite_output_bub`, and `composite_file_pattern_bub`.
11.09.2022	Ulrich Blahak	In obs data mode, enable to process OPERA hdf5 files of Slovakia.
23.11.2022	Jana Mendrok	New namelist parameters in `dbz_meta` that allow specification of hydrometeor shape and orientation for dual polarization calculations. Inserted in Table 14.
29.11.2022	Ulrich Blahak	Renamed type `dbzcalc_params` to `t_dbzcalc_params`.
07.12.2022	Ulrich Blahak	Increased `nradsta_max` from 80 to 120. Decreased `ndoms_max` from 5 to 3.

Date	**Name**	**Change**
10.4.2023	Ulrich Blahak	New format for namelist parameters `rs_meta%dt_obs`, `rs_meta%dt_obs_voldata`, `rs_meta%dt_obs_fdbk`, `dt_obs_voldata_glob` and `dt_obs_fdbk_glob`. Instead of specifying just the time increment for the time lists and implying that the times run from model start time to end time, now a time triple has to be given: `from-time, to-time, increment`. However, for backwards compatibility, one can also give only one value in the namelist, which is interpreted as the increment and automatically shifted to the correct position in the triplet. `from-time` and `to-time` are set to -999.9 in this case. If `from-time` is less than -900, the series of times starts (approximately) at model init time and is synchronized to time 0.0 s. If `from-time` is larger than -900, it is interpreted as the exact start time of the series and is not synchronized to time 0.0 s.
07.08.2023	Ulrich Blahak	Increased `nradsta_max` from 120 to 140.
07.08.2023	Ulrich Blahak	KIT hdf5 radar observation files can now be processed: KIT C-band radar Karlsruhe with pseudo WMO-ID 20001 and non-standard hdf5-files (see Section 4.3) with an own special reader. KIT-Cube X-band radar with pseudo WMO-ID 10000 and ODIM-conforming hdf5-files, using the existing OPERA reader (see Section 4.3).
24.08.2023	Ulrich Blahak	New switch `dbz_meta%ldynamic_wetgrowth_gh` (default `.FALSE.`) If `.TRUE.`, switch on new option for dynamic determination of `dbz_meta%Tmeltbegin_g/h` and `dbz_meta%meltdegTmin_g/h` for graupel and hail categories in each grid column, according to the actually present supercooled liquid in relation to the actual mean particle sizes and the wet growth diameters. This improves simulated radar moments in and above the melting layer. More informations in Section 5.1.5.
30.08.2023	Ulrich Blahak	Changed the default T-thresholds related to the freezing point of water from .16 to .15 for consistency to ICON microphysics. This changes all radar moments slightly, but not significantly.
21.05.2024	Ulrich Blahak	New namelist parameters `dbz_meta(`i`)%ext_tune_fac_pure` and `dbz_meta(`i`)%ext_tune_fac_melt` (logical) to tune the attenuation coefficients. This enables to “adjust” the forward simulation to observations where the attenuation has been partly/conservatively corrected. Updates of Table 14 in Section 5.1.5.
24.9.2024	Ulrich Blahak	New namelist switch `labort_if_problems_gribout` in `/RADARSIM_PARAMS/`. Inserted into Table 12.
19.12.2024	Jana Mendrok	Split namelist parameter `supob_ave_width` into separate parameters for reflectivity and radial wind. Updated in Table 12.
20.12.2024	Jana Mendrok	New namelist parameter `rain2mom_mu_incloud` in `/RADARSIM_PARAMS/`. Inserted into Table 12.
21.1.2025	Ulrich Blahak	Update names of F90 source files, because in the last year some really big files have been split into several smaller modules.
28.1.2025	Ulrich Blahak	New namelist variable `fof_file_pattern` to define the file name pattern for feedback (fof) files.

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 ($\mathrm{m\,s^{-1}}$)
- horizontally polarized radar reflectivity Z_h (dBZ)
- differential reflectivity Z_{dr} (dB) sd
- horizontal attenuation coefficient A_h (dB/km)
- specific differential phase shift K_{dp} ($^\circ$/km)
- total differential phase shift Φ_{dp} ($^\circ$)
- cross-correllation coefficient ρ_{hv}
- linear depolarization ratio LDR
- differential attenuation A_{dp} (db/km)

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 to a large degree (Mie- and Rayleigh options, beam bending options, beam smoothing options) in Blahak (2016), Zeng (2013), Jerger (2014), Zeng et al. (2014) and Zeng et al. (2016). The listed polarization parameters have recently been added and are based on the works of Ryzhkov (2001) and Ryzhkov et al. (2011), approximating all hydrometeors as one- or two-layered oblate spheroids and employing the T-matrix method to compute scattering properties. Here it is assumed that the so-called "backscatter-rule" as an approximation of the scattering-properties as funcion of the particle orientation relative to the incoming wave holds, which requires that the particle size is not too large compared to the wavelenght (Ryzhkov, 2001) and that the elevation angle is not too steep (Ryzhkov et al., 2011). It is planned to relax this condition in the near future.

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 (Z_h) 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. For this, the computation spacially averaged super-observations at regular spatial intervals is possible as an option. 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 polarimetric parameters have been integrated into these processes accordingly.

The present document provides an 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 ("core" modules). To connect the model world to the EMVORADO core, there is one model-specific module `radar_interface.f90` which, for example, exchanges time informations, feeds the model state variables to EMVORADO, provides interpolation routines (model grid $\Longrightarrow$ geographic coordinates $\Longrightarrow$ radar bins), exchanges MPI communicators and -data types, and so on. Concerning the coupling to the particle- and size distribution shapes of specific bulk cloud microphyscis schemes, EMVORADO currently implements the 1-moment- and 2-moment bulk microphysics of COSMO and ICON, but in a rather generic way, so that it might be possible to connect also other microphysics schemes. More details on the implementation and coupling can be found in Section 8.1.

There is also a so-called "offline" framework (a simple but fully MPI-parallel stand-alone wrapper around the core), which reads one timestep of model fields from disk and feeds it to the core. The parallelization strategy of the wrapper is derived from that of the COSMO-model and makes use of parts of the COSMO code. For the core, it "fakes" the COSMO environment (data structures, timers, domain decomposition, rotated lat-lon grid) and expects a COSMO-like rotated lat-lon grid for the model state variables on input. Because ICON is able to interpolate it's output variables to such a grid for output, it is possible to use the offline framework also for ICON input data.

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 COSMO 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.

Internally in EMVORADO one radar station is defined as the combination of a WMO station ID and a certain scan strategy (= set of elevation angles). This means for example, that if the same station conducts two alternating scan strategies, these are internally handled as two different radar stations. In that sense, DWD's horizon-following "precipitation scans" which alternate with volume scans are handled as different radar stations.

With regards to its contents, the operator for Z_h, A_h and the polarization parameters is composed of two steps.

Step 1: Computation of unattenuated Z_h, A_h (optional for Mie- and T-matrix scattering, see below) and polarization parameters 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) for Z_h and A_h and Ryzhkov et al. (2011) for the polarization parameters. There are 5 general options. The first is Mie-scattering, the second and third are two variants of the so-called Rayleigh approximation for particles small compared to the wavelength, the fourth is T-matrix for oblate spheroids, and the fifth is T-matrix for spheres. The main difference of the two Rayleigh options is the treatment of melting hydrometeors.

Step 2: Interpolation/aggregation of the Z_h, A_h and polarization parameters to the radar bins (polar coordinates range, azimuth, elevation) with an option for explicit ray tracing, optional iterative computation of attenuation effects along each ray (only possible for Mie- or T-matrix options), 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 scattering-theory-specific settings, some efficiency is gained by just re-using the computed radar parameters from the last radar instead of repeating the computation.

This procedure is the traditional way of computing Z_h and A_h in EMVORAO, which can be very efficient for large radar networks, because it reduces the amount of Z_h and A_h computations to a minimum. However, consistency of the different parameters with respect to the implicit particle size distribution is not guaranteed, if the interpolation to radar bins in step 2 is of type linear (COSMO). For this, a new option has been implemented to interchange the order of interpolation and computation of parameters: first the model state variables (hydrometeors, temperature T, pressure p) are interpolated to the radar bins before computing the scattering parameters on the radar bins. This ensures consistency among the parameters, because they are based on the same interpolated hydrometeor properties. The modified steps are as follows, if the namelist parameter `lcalc_dbz_on_radarbins` is set .TRUE.:

Modified step 1: Interpolation of model state variables to the positions of the radar bins, optionally after explicit ray tracing, then computation of unattenuated Z_h, A_h (optional for Mie- and T-matrix scattering, see below) and polarization parameters on the radar bins based on the interpolated hydrometeor fields and T. In this way, the computation of the radar parameters has to be triggered for the radar bins of each radar station individually and no re-use of computed values from a previous computation is possible. Depending on the number of model grid points and the number of radar stations and their scan strategy, this might be computationally more cheap or more expensive. But in this way it is naturally to take the local elevation angle correctly into account, which might become relevant at higher elevations for oblate spheroids.

Modified step 2: Optional averaging of the Z_h, A_h and polarization parameters over the effective beam weighting function, optional iterative computation of attenuation effects along each ray (only possible for Mie- or T-matrix options), starting at the radar station, and output of volume scan data to files.

Note that this is not necessary for ICON, because in that case the interpolation method in step 2 is Nearest Neighbour, which preserves the consistency. This method leads however to coarser radar signatures in simulated data and has been chosen for technical reasons due to the unstructured nature of the triangular ICON model grid. Other interpolation methods would have been computationally much more expensive because there are no regulatrities which can be exploited to efficiently find the surrounding grid point neighbourhood for interpolation.

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 values 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, azimuth 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 radial wind, Z_h and polarization parameters 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, azimuth, 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 azimuth angles of the radar scan strategy. For the ray tracing, the complete data of one azimuthal 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 scattering computations of different physical accuracy and efficiency

The choice of the reflectivity and dual polarimetric parameter 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- and T-matrix options. 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 5.1.5 and 5.2.

For Mie- and T-matrix 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- and T-matrix scattering become 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 and derived types. The tables themselves are autogenerated by EMVORADO if necessary; the full Mie- and T-matrix 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. With this, Z_h and optionally A_h are simulated. Particle sizes are allowed to be larger than the wavelength, but all particles are assumed spherical. The dual polarization parameters are simply set to their "spherical" neutral values. 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. Only Z_h is simulated. The approximation contains an analytic formulation for melting hydrometeors which leads to a systematic underestimation of bright bands. Z_h 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 for Z_h only, but it contains only a much more simplistic treatment of melting hydrometeors, leading to unrealistic and "jumpy" bright bands. Z_h is also overestimated, if the particle sizes are comparable or larger than the radar wavelength.
- `itype_refl = 5`: Similar to `itype_refl=1` but replacing Mie by T-matrix calculations assuming oblate spheroids following Ryzhkov (2001); Ryzhkov et al. (2011). EMA's and melting model are the same as for Mie. With this, Z_h, Z_{dr}, A_h, ρ_{hv}, K_{dp}, Φ_{dp}, LDR, and A_{dp} are simulated. Further necessary assumptions on size-dependend particle asymmetry and hydrometeor-dependend canting angle distributions are largely taken from Ryzhkov et al. (2011) but subject to future changes.
- `itype_refl = 6`: Similar to `itype_refl=5` but replacing oblate spheroids by ordinary spheres. This option can be used to cross-check T-matrix codes with Mie in the spherical limit.

For the other parameters in Table 14, 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 has been 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 Z_h (COSMO only)

With the coupling of EMVORADO to COSMO, the traditional output of (unattenuated) Z_h 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 EMVORADO methods of Mie-/T-matrix scattering and Rayleigh-Oguchi-approximation as in (unmodified) step 1 of EMVORADO. **No polarization parameters at the moment!** 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 it is not implemented yet.

2.6 Lookup tables for Mie- and T-matrix scattering options

Concerning the Mie- and T-matrix options for Z_h, A_h and polarization parameters, considerable speedup can be gained by using lookup tables (one table for each model hydrometeor category plus melting hydrometeors). T-matrix computations are even more (much more!) demanding than Mie-computations and lookup tables are key for any larger-scale operational application of EMVORADO in data assimilation or model verification. Therefore this option is strongly recommended and can be chosen by namelist switches, both for step 1 computations in EMVORADO and for the 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- and T-matrix 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.

For the COSMO- and ICON-models and their bulk cloud microphysical schemes with pre-scribed mass-size-relations, particle size distribution (PSD) shapes and other particle-related assumptions, the simulated radar moments like Z_h, Z_{dr}, A_h, K_{dp} and other polarization parameters depend uniquely on the prognostic hydrometeor moments and temperature T. The temperature dependency arises due to the temperature-dependend melting scheme in EMVORADO and to a lesser degree due to the temperature dependency of the refractive indices for ice and water. EMVORADO's lookup tables tabulate, separately for each hydrometeor type and dry/melting state, the radar moments at regularly spaced intervals (table nodes) of the hydrometeor variables and temperature. Once pre-computed, the radar moments for actual model states are computed by fast interpolation of the tabulated values and summed up over all hydrometeor types. The tables are as low dimensional as possible. In the simplest case of a one-moment scheme there are two dimensions, mass density and T. But even for the two-moment scheme of COSMO and ICON and a melting hydrometeor type there are only 3 dimensions, namely the ratio of mass- to number density (mean size), T and a proxy for the melting state.

The names of the lookup table files are unique and consist of the hydrometeor type names, the scattering theory name, an identifier for the cloud microphysics scheme with which they are consistent, a "magic number" (10 digit signed integer) which is a unique hash-value representing the exact configuration parameters of the reflectivity/scattering calculation (c.f. Sections 5.1.5 and 5.2), and the internal version number. Examples of files containing Mie- and T-matrix tables for dry graupel are

```
radar_tmatrixtab_2mom_drygraupel_1545262779_version009.bin
radar_mietab_2mom_drygraupel_0583900982_version009.bin
```

The hash value depens uniqely on the configuration parameters for the reflectivity/polarization calculation. At simulation start, EMVORADO checks for each hydrometeor type if a table file with the respective hash-value is present on disk. If yes, the existing table is read from disk and used. If not, the generation of the table is triggered, which takes some additional runtime, and the result is stored in a new table file on disk for subsequent model runs.

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 (COSMO only at the moment) 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.

Motivated by the very long time it may take to generate T-matrix tables, the table generation is parallelized (MPI) over the processors of the model run. The way how this task is parallelized in detail can be configured by the namelist switches `itype_mpipar_lookupgen`, `pe_start_lookupgen` and `pe_end_lookupgen`. Their defaults are a reasonable choice for most platforms.

2.7 Output of radar composites

2D composites of simulated and observed Z_h (useful e.g. for spatial model verification in dBZ-space or for the detection of missing cells for automatic warm bubbles, see Sections 5.1.7 and 7) on a rotated lat-lon-grid are available for output through EMVORADO. Note that there are no composites for other scattering parameters or radial wind, though.

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, asymmetric 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 grids (rotated lat-lon-grids) can be arbitrarily defined. There can be different composite grid definitions for the "normal" ouptut and for the automatic warm bubbles. The choice of a coarser grid for the missing cell search considerably speeds up the warm bubble generator.

To enable the "normal" composite computation and output, set the master switch `lcomposite_output = .TRUE.` in `/RADARSIM_PARAMS/`. An own grib-output method is provided by EMVORADO itself, based on special grib sample files provided through EMVORADO. Several composites for different radar elevations can be computed and output simultaneously. Their number is given by `nel_composites` $>$ `0` and the respective elevation indices by `eleindlist_for_composites_glob` in `/RADARSIM_PARAMS/`. `eleindlist_for_composites_glob` is a global setting for all radar stations, but it can be adjusted for station i by the derived type parameter `rs_meta(`i`)%eleindlist_for_composite`.

If the warm bubble generator is active (`ldo_bubbles=.TRUE.`), the respecitve composite can also be output for reference. For this, set `lcomposite_output_bub = .TRUE.` in `/RADARSIM_PARAMS/`.

In COSMO, the default settings for the composite grids 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.` respectively `lcomposite_output_bub = .TRUE.`, cf. Section 6.1 below.

For ICON, the default composite grid for the "normal" output and for the automatic warm bubbles is that of the COSMO-D2 configuration.

In any case, the composite grids may be adjusted to the User's needs by a set of namelist parameters in the derived types `comp_meta` respectively `comp_meta_bub`, see Section 5.1.7 and Table 12.

3 Compilation aspects of the different frameworks

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 as well as in the COSMO-derived offline framework.

The EMVORADO core itself is hosted at a git repository[1] at the German Climate Computing Center (DKRZ) in Hamburg. The offline framework is hosted on another git repository[2]. Potential users are welcome to contact the author for access, which may require some sort of legal agreement.

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[3] in certain branches (contact the author about more informations). It is updated from time to time by a newer EMVORADO core versions by simply replacing the core files in the COSMO source by newer files from the core git and recompiling.

For ICON only the ICON-specific interface files are part of the source code distribution[4] and are located in the subdirectory (`./src/data_assimilation/interfaces/` of the ICON source tree. The EMVORADO core files are contained in the ICON submodule `‘‘emvorado-for-icon’’`. This submodule is itself the master branch of a third git repository[5] and every ICON user with access to the ICON repository has access to it. It is loaded automatically with the other ICON submodules when doing the usual `git submodule update --init --recursive` after cloning the ICON repo.

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 volume scans and composites in grib2 format. Requires DWD’s `grib_api` or `eccodes` distributions because of some needed local definitions (center=EDZW), with support for ccscs-compression (`aec` library).
- `-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 DWD, 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` process has been extended by a target `--enable-emvorado`. This target has to be added to the calls of the configure wrapper script for your specific platform, or to the shell variable `EXTRA_CONFIG_ARGS` in the wrapper. Here is an example for configuring and compiling ICON:

```
$> ./config/dwd/linuxWS.gcc --enable-emvorado
$> make -j 10
```

[1] git@gitlab.dkrz.de:dace_projects/emvorado-package.git
[2] git@gitlab.dkrz.de:dace_projects/emvorado-offline.git
[3] git@github.com:COSMO-ORG/cosmo.git
[4] git@gitlab.dkrz.de:icon/icon-nwp.git
[5] git@gitlab.dkrz.de:dwd-sw/emvorado-for-icon.git

By using this mechanism, all of the above preprocessor switches except `-D__COSMO__` 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.

For the standalone offline framework in the git repository of footnote 2 on page 16, we provide a number of exemplary Makefiles. These Makefiles require an automatic dependency checker called "makedepf90". This very old yet useful program of Erik Edelmann might not be available anymore on the internet, therefore we include it in the repository. You have to compile it prior to executing any of the below Makefiles:

```
$> cd <emvorado-offline-dir>/externals/makedepf90
$> ./configure
$> make -j4
```

See the `<emvorado-offline-dir>/externals/makedepf90/README` for further information and credits.

The exemplary Makefiles can be found in subdirectories for a variety of different platforms:

- Linux workstation, gfortran, openMPI:
  `<emvorado-offline-dir>/build_gfortran/Makefile.gfortran.openMPI.makedepf90`
  `$> cd ./build_gfortran $> make -f Makefile.gfortran.openMPI.makedepf90 -j8`

- Linux workstation, gfortran, serial compile:
  `<emvorado-offline-dir>/build_gfortran/Makefile.gfortran.serial.makedepf90`
  `$> cd ./build_gfortran $> make -f Makefile.gfortran.serial.makedepf90 -j8`

- DWD Linux gateway cluster to CRAY-XC40, intel Fortran compiler, MPI:
  `<emvorado-offline-dir>/build_lce/Makefile.lce.ifort.MPI.makedepf90`
  `$> cd ./build_lce $> make -f Makefile.lce.ifort.makedepf90 -j8`

- DWD CRAY-XC40, cray-compiler, hybrid MPI and OpenMP:
  ```
  <emvorado-offline-dir>/build_xce/Makefile.xce.cray.MPI.makedepf90
  $> cd ./build_xce
  $> make -f Makefile.xce.cray.makedepf90 -j8
  ```

- DWD Linux gateway cluster to NEC SX Aurora, choice of intel or gfortran, MPI, consult this README:
  ```
  <emvorado-offline-dir>/build_rcl/README.compile.rcl
  $> cd ./build_rcl
  $>make_rcl_intel.sh -j8  # accepts all options of make
  ```
  or
  ```
  $>make_rcl_x86-gnu.sh -j8  # accepts all options of make
  ```

- DWD NEC SC Aurora vector-parallel supercomputer, NEC compiler, hybrid MPI and OpenMP, requires two binaries for a hybrid vector-engine/vector-host execution. This wrapper script contains the steps to compile on the NEC SX Aurora:
  ```
  <emvorado-offline-dir>/build_NEC/make_VH_VE.sh
  $> cd ./build_NEC
  $> make_VH_VE.sh -j8  # accepts all options of make
  ```

If all things go right, the result is an executable whose name contains the actual branch name (e.g., `release/dualpol-beta-0.9`) and an identifier for platform and compiler.

To actually run the executable, there are a number of exemplary runscripts for the different platforms and compilers in the repository. Please consult the `README` in the top level directory of the offline version's repository.

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 or the standalone offline framework. For COSMO, there are exemplary runscripts for each of the 3 modes in the source code distribution, see below. For the offline framework, consult the `README` in the top-level directory of the offline version's repository and there are many exemplary commented runscripts as well.

A general description of `/RADARSIM_PARAMS/` will be given below in Sections 5.1.3 to 5.1.5.

4.1 Idealized mode

Purely synthetic radars are simulated within an idealized model simulation (COSMO: Blahak, 2015; ICON: Prill et al., 2020). The radar station metadata (geographic location, wavelength, scan strategy, etc.) can be defined via namelist parameters in `/RADARSIM_PARAMS/`. One can have up to 140 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 5.1.3 to 5.1.5.

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 140 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 5.1.3 to 5.1.5.

Hint: DWD radars provide a special "precipitation scan" in addition to the volume scan every 5 minutes. This scan consists of one antenna revolution with variable elevation angle just above the horizon. It is possbile to simulate also this type of scan, not only PPI scans with constant elevation, if a simulated radar station has a German station template `rs_meta%icountry=1` and a valid DWD `rs_meta%station_id`. Setting the number of elevations to 1 and choosing 0.8° as the elevation angle will trigger the simulation of the precipitation scan for the respective station.

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 140 radar stations (directory `ydirradarin` and namelist parameters 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 5.1.3 to 5.1.5.

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 (Z_h and v_r only) 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`

  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 DWD.** These files contain one sweep (elevation) per per station per observation time. Depending on the version number in the file name, the files may contain one parameter (single-sweep single-moment) or all parameters (single-sweep multi-moment). **Polarization parameters other than Z_h from DWD radars are only available in this format!**

  The **single-moment single-sweep** files are expected to follow the following naming convention:

`ras<vers>-<scandef>_sweeph5onem_<datasetname>_<eleindex>-<datetime>-<stationname>-<stationid>-hd5`

  - `<vers>`: Two-digit version number, "07" (postprocessed from POLARA), "11" (raw files from the radar sites).
  - `<scandef>`: "pcpng01" (DWD's horizon-following precipitation scan) or "vol5minng01" (5-minute volume scan)
  - `<datasetname>`: identifier of the data set, e.g., "dbzh", "rhohv", "vradh", etc.
  - `<eleindex>`: Two-digit elevation index, "00" = precipitation scan; volume scan starting from "01"

- `<datetime>`: Exact time reference for the PPI, 16-digit date string `YYYYMMDDhhmmss00`
- `<stationname>`: 3-character station name, e.g., "neu" for Neuheilenbach
- `<stationid>`: WMO station ID (5 digits for German stations)

for example,

- `rasXX-pcpng01_sweeph5onem_dbzh_00-2020020300053400-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_00-2020020300055800-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_01-2020020300062100-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_02-2020020300064400-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_03-2020020300070800-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_04-2020020300073100-neu-10557-hd5`

The **multi-moment single-sweep** files are expected to follow the following naming convention:

`ras<vers>-<scandef>_sweeph5allm_any_<eleindex>-<datetime>-<stationname>-<stationid>-hd5`

- `<vers>`: Two-digit version number, "07" (postprocessed from POLARA), "11" (raw files from the radar sites).
- `<scandef>`: "pcpng01" (DWD's horizon-following precipitation scan) or "vol5minng01" (5-minute volume scan)
- `<eleindex>`: Two-digit elevation index, "00" = precipitation scan; volume scan starting from "01"
- `<datetime>`: Exact time reference for the PPI, 16-digit date string `YYYYMMDDhhmmss00`
- `<stationname>`: 3-character station name, e.g., "neu" for Neuheilenbach
- `<stationid>`: WMO station ID (5 digits for German stations)

for example,

- `rasXX-pcpng01_sweeph5onem_dbzh_00-2020020300053400-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_00-2020020300055800-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_01-2020020300062100-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_02-2020020300064400-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_03-2020020300070800-neu-10557-hd5`
- `rasXX-vol5minng01_sweeph5onem_dbzh_04-2020020300073100-neu-10557-hd5`

Besides reflectivty and radial wind, polarisation parameters can be read and used from these files. This is an advantage compared to the above CDFIN files of DWD.

- **ODIM HDF5 files from MeteoSwiss.** These files contain one sweep (elevation) of one parameter of one observation time per file **(single-moment single-sweep)**. EMVORADO processes only Z_h and v_r at the moment. The files 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`

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

  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.

- **ODIM HDF5 files from ARPA-Piemonte and ARPA-SIMC (Italy).** These files contain all elevations of all parameters from one observation time per file **(multi-moment multi-sweep)**. EMVORADO processes only Z_h and v_r at the moment. The files 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.

- **ODIM HDF5 files from the OPERA data hub at DWD.** These files contain European radar data in individually different compositions **(single-/multi-moment single-/-multi-sweep)**. EMVORADO processes only Z_h and v_r at the moment. The files are expected to follow the naming convention:

  `T_PA<S><E><NR>_C_<COUN>_<YYYYMMDDhhmmss>.<EXT>`

  - `<S>`: Scan type identifier, 1 character
  - `<E>`: Elevation identifier, 1 character
  - `<NR>`: Station number, 2 digits
  - `<COUN>`: Generating center (=country) identifier, 4 characters (e.g. "EDZW", "LSSW", "EBUM", etc.)
  - `<YYYYMMDDhhmmss>`: Date string, usually the start time of the scan, but sometimes the nominal time (e.g. for Switzerland the beginning of the 5-min interval, and for Czech Republic the true end time)
  - `<EXT>`:Ffile extention, eihter "hdf" or "h5"

  for example,

  - `T_PAGA53_C_EDZW_20220206100301.hdf`
  - `T_PAHZ41_C_EBUM_20220206100020.hdf`
  - `T_PAZB40_C_LFPW_20220206100050.h5`
  - `T_PAHZ60_C_OKPR_20220206100913.hdf`

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

Currently, the radars from the following countries are implemented: Germany, Switzerland, Belgium, France, Denmark, Netherlands, Poland, Czech Republik, and Slovakia.

See Tables 2, 3, 4, 5, and 6 for more details on the radar data of each of these countries.

Additionally, reflectivty and radial wind data from the **X-band research radar of the Karlsruhe KIT-Cube (pseudo WMO-ID 10000)** can also be ingested in this context. For this, the original data files have to be renamed:

  – Reflectivity files `2023071213400900dBZ.vol.h5` to "GZ99" files `T_PAGZ99_KITC_20230712134009.h5`
  – Radial wind files `2023071213400900V.vol.h5` to "HZ99" files `T_PAHZ99_KITC_20230712134009.h5`

with the provider `KITC`.

- **Non-standard HDF5 files from the KIT C-Band radar in Karlsruhe, pseudo WMO-ID 20001.** These files contain all sweeps (elevations) of all parameters of one observation time per file **(multi-moment multi-sweep)**. EMVORADO processes Z_h, v_r and polarisation parameters. The files are expected to follow the naming convention:

  `scan-sidpol-<RAN>km-<NE>_20001_<YYYYMMDDhhmmss>_00.h5`

  – `<RAN>`: Maximum range in km
  – `<NE>`: Number of elevations in the volume scan
  – `<YYYYMMDDhhmmss>`: Date string, true start time of the scan (first elevation)

  for example,

  – `scan-sidpol-120km-14_20001_20230712123001_00.h5`
  – `scan-sidpol-120km-14_20001_20230712123503_00.h5`

  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, ARPA-Piemonte, Belgium, France, Denmark, Netherlands, Poland, Czech Republik, and Slovakia are included, as well as two research radars from KIT Karlsruhe. For other stations, respective entries in the background list would have to be added to the code (`radar_obs_meta_list.f90`).

Note that in the EMVORADO code two scan strategies of the same station (e.g., simultaneously having DWD volume and precipitation scans in one run) are treated internally as two different stations. An additional scan identifier (`rs_meta%scanname`, see Table 13 in Section 5.1.4) is used for discrimination in addition to the WMO-ID `rs_meta%station_id`.

There is also the possibility to overwrite all the station metadata from observation files by namelist. This is described in more detail in Section 5.1.6 and enables, e.g., to correct erroneous metadata from the observation files or even adding new stations to simulate synthetic data without having observation equivalents.

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.

Table 2: Some details about the implemented OPERA radar data in ODIM hdf5 format of Switzerland and Germany.

	Switzerland LSSW	**Germany EDZW**
File type	multi-moment/single-sweep	single-moment/single-sweep
Reflectivity file pattern	T_PAGA41_C_LSSW_20220104100000.hdf T_PAGB41_C_LSSW_20220104100000.hdf . . .	T_PAGA52_C_EDZW_20220206100301.hdf T_PAGB52_C_EDZW_20220206100230.hdf T_PAGC52_C_EDZW_20220206100207.hdf . . . T_PAGA52_C_EDZW_20220206100301.hdf T_PAGB52_C_EDZW_20220206100230.hdf T_PAGC52_C_EDZW_20220206100207.hdf . . .
Radial wind file pattern	T_PAGA41_C_LSSW_20220104100000.hdf T_PAGB41_C_LSSW_20220104100000.hdf . . .	T_PAHA52_C_EDZW_20220206100301.hdf T_PAHB52_C_EDZW_20220206100230.hdf T_PAHC52_C_EDZW_20220206100207.hdf . . . T_PAHA52_C_EDZW_20220206100301.hdf T_PAHB52_C_EDZW_20220206100230.hdf T_PAHC52_C_EDZW_20220206100207.hdf . . .
Stations (NR = WMOID = Name)	41 = 6661 = Albis 42 = 6699 = La Dole 43 = 6768 = Monte Lema 51 = 6726 = Plaine Morte 52 = 6776 = Weissfluhgipfel	40 = 10103 = deasb 43 = 10169 = deros 44 = 10339 = dehnr 51 = 10410 = deess 52 = 10440 = defld 53 = 10356 = deumd 55 = 10132 = deboo 61 = 10629 = deoft 62 = 10557 = deneu 63 = 10488 = dedrs 65 = 10392 = depro 71 = 10605 = denhb 72 = 10832 = detur 73 = 10780 = deeis 75 = 10873 = deisn 81 = 10908 = defbg 84 = 10950 = demem
Wavelength	C-Band (we take 0.055 m, but the true wavelengths are slightly different in reality)	C-Band (we take 0.055 m, but the true wavelengths are slightly different in reality)

Elevations (<E>)	A = -0.2°
	B = 0.4°
	C = 1.0°
	D = 1.6°
	E = 2.5°
		A = 0.5°
		B = 1.5°
		C = 2.5°
		D = 3.5°
		E = 4.5°
		F = 5.5°
		G = 8.0°
		H = 12.0°
		I = 17.0°
		J = 25.0°

Table 2: continued

	Switzerland LSSW	**Germany EDZW**

Remarks	Scan identifier: G
	
	Time stamp is the beginning of the nominal 5 min interval.
	
	Radial wind not usable for assimilation because of too low Nyquist interval. There is a possibility to deactivate radial wind in the feedback files in general or based on a Nyquist threshold.
	
	The following settings of these parameters are the default for Swiss radars, so that radial winds are ignored for feedback files, regardless if the radial winds are active otherwise (loutradwind=.true. and Swiss radial wind files present in input direcory):
	
	`rs_meta(i)%lvrad_to_fdbk = .FALSE.`
	`rs_meta(i)%vnyq_min_for_vr_active_fdbk = 25.0`
		Scan identifier G = files for reflectivity
		
		Scan identifier H = files for radial wind
		
		Only the volume scans, no precip scans in OPERA.

Table 3: Some details about the implemented OPERA radar data in ODIM hdf5 format of Belgium and France.

	Belgium EBUM	**France LFPW**
File type	single-moment/multi-sweep	multi-moment/single-sweep
Reflectivity file pattern	T_PAGX41_C_EBUM_20220206100019.hdf . . . T_PAGX42_C_EBUM_20220206100410.hdf . . .	T_PAZA37_C_LFPW_20220206100019.h5 T_PAZB37_C_LFPW_20220206100051.h5 . . . T_PAZA40_C_LFPW_20220206100022.h5 T_PAZB40_C_LFPW_20220206100050.h5 . . .
Radial wind file pattern	T_PAHZ41_C_EBUM_20220206100020.hdf . . . T_PAHZ42_C_EBUM_20220206100424.hdf . . .	T_PAZA37_C_LFPW_20220206100019.h5 T_PAZB37_C_LFPW_20220206100051.h5 . . . T_PAZA40_C_LFPW_20220206100022.h5 T_PAZB40_C_LFPW_20220206100050.h5 . . .

Table 3: continued

	Belgium EBUM	**France LFPW**
Stations (NR = WMOID = Name)	41 = 6477 = bewid 42 = 6410 = bejab	37 = 7760 = fraja 40 = 7005 = frabb 41 = 7510 = frbor 42 = 7255 = frbou 44 = 7436 = frgre 45 = 7027 = frcae 47 = 7180 = frnan 50 = 7629 = frtou 51 = 7145 = frtra 52 = 7168 = frtro 53 = 7471 = frlep 54 = 7223 = frtre 56 = 7108 = frpla 58 = 7381 = frniz 62 = 7637 = frmcl 63 = 7083 = frave 64 = 7336 = frche 65 = 7274 = frbla 66 = 7606 = frmom 67 = 7291 = frmtc
Wavelength	C-Band (we take 0.055 m, but the true wavelengths are slightly different in reality)	C-Band (we take 0.055 m, but the true wavelengths are slightly different in reality)

Elevations (<E>)	**Station 41:** 0.5°, 1.2°, 2.1°, 3.4°, 4.8°, 6.5°, 9.0°, 13.0°, 25.0°
	Station 42: 0.3°, 0.9°, 1.5°, 2.2°, 2.9°, 3.8°, 4.8°, 6.5°, 9.0°, 13.0°, 25.0°
		A = ≈ 6°- 90°
		B = ≈ 3°- 8°
		C = ≈ 0.8°- 5°
		D = ≈ 1.0°- 2.5°
		E = ≈ 0.5°- 4.5°
		F = ≈ 0.4°- 0.8°

Table 3: continued

	Belgium EBUM	**France LFPW**
Remarks	Scan identifier: GX = files for reflectivity HZ = files for radial wind Scan strategies are constant over time. Radial wind should be usable due to sufficient Nyquist velocity (53.5 m/s)	Scan identifier = Z 5 or 6 elevations (depending on station) with periodically alternating elevation angles. Each station has a different set of possible elevations (between 8 and 12) for this alteration. However, at least the lowest 3 elevations for a station stay the same in time. At some stations there is sometimes a 90 degress bird-bath scan. This scan is ignored in EMVORADO, which reduces the maximum number of elevations of any radar to 11. In the background meta data list, each radar station has its individual set of the 8-11 possible elevation angles as default scan strategy (without the bird-bath scan). This will be the internal set of elevations in the operator and will be output. Missing elevations for a certain time will be filled with missing data (-999.99). Up to now no filtering for not-present elevations in the outputs (fdbk, voldata), but this could be implemented perhaps based on rs_meta(¡i¿)should pose any problems. Radial wind is usable. Nyquist velocities are around 60 m/s. The radial winds are put to the feedback files by default, if `loutradwind=.TRUE.` Default: rs_meta(i)%lvrad_to_fdbk = .TRUE.

Table 4: Some details about the implemented OPERA radar data in ODIM hdf5 format of Denmark and Netherlands.

	Denmark EKMI	**Netherlands EHDB**
File type	multi-moment/multi-sweep	multi-moment/multi-sweep
Reflectivity file pattern	T_PAZZ41_C_EKMI_20220206100000.h5 T_PAZZ41_C_EKMI_20220206100500.h5 ... T_PAZZ42_C_EKMI_20220206100000.h5 T_PAZZ42_C_EKMI_20220206100500.h5 ...	T_PAGZ51_C_EHDB_20220206100004.h5 T_PAGZ51_C_EHDB_20220206100504.h5 ... T_PAGZ52_C_EHDB_20220206100004.h5 T_PAGZ52_C_EHDB_20220206100504.h5 ...
Radial wind file pattern	T_PAZZ41_C_EKMI_20220206100000.h5 T_PAZZ41_C_EKMI_20220206100500.h5 ... T_PAZZ42_C_EKMI_20220206100000.h5 T_PAZZ42_C_EKMI_20220206100500.h5 ...	T_PAGZ51_C_EHDB_20220206100004.h5 T_PAGZ51_C_EHDB_20220206100504.h5 ... T_PAGZ52_C_EHDB_20220206100004.h5 T_PAGZ52_C_EHDB_20220206100504.h5 ...

Table 4: continued

	Denmark EKMI	**Netherlands EHDB**
Stations (NR = WMOID = Name)	41 = 6173 = Stevns 42 = 6096 = Römö 43 = 6034 = Sindal 44 = 6194 = Bornholm 45 = 6103 = Virring	51 = 6234 = nldhl (Den Helder) 52 = 6356 = nlhrw (Herwijnen)
Wavelength	C-Band (we take 0.055 m, but the true wavelengths are slightly different in reality)	C-Band (we take 0.055 m, but the true wavelengths are slightly different in reality)
Elevations (<E>)	Alternating elevations from 5 min to 5 min and slightly different for each station. This is the super-set: 0.5°, 0.7°, 1.0°, 1.5°, 2.4°, 4.5°, 4.8°, 8.5°, 10.0°, 13.0°, 15.0° For certain times and stations, some of them might be missing. The sometimes appearing 8.4° is treated as 8.5°.	Usable set of elevations: 0.3°, 0.8°, 1.2°, 2.0°, 2.8° The 0.3° are measured 3 times, but only once with a range increment of 223.5 m. And only this data set is accepted by EMVOADO.
Remarks	Scan and elevation identifiers = ZZ Time stamp is the beginning of the nominal 5' min interval. Alternating scans from 5' to 5' interval with different optimizations (all radar moments are always delivered): At minutes 00, 10, 20, etc.: reflectivity-optimized with longer range (240 km) and smaller Nyquist velocity (8.3 m/s) At minutes 05, 15, 25, etc.: radial-wind-optimized with shorter range (120 km) and larger Nyquist velocity (47.2 m/s) Radial wind is thus usable for every second time interval. To deactivate radial winds of the not usable times for data assimilation, one could set rs_meta(i)%lvrad_to_fdbk = .TRUE. but rs_meta(i)%vnyq_min_for_vr_active_fdbk = 25.0° This is the default in EMVORADO for Danish radars. Station 45 has a birdbath scan (89°) which is ignored by EMVORADO on input.	Scan and elevation identifiers = GZ 2 Radars, but only 5 low elevations where we have constant range increment of 223.5 m. All other elevations use individually different range increments and are thus not to handle technically at the moment. DBZH and VRADH are both usable. Nyquist velocity is 32, 48 or 80 m/s

Table 5: Some details about the implemented OPERA radar data in ODIM hdf5 format of Poland and Czech Republic.

	Poland SOWR	**Czech Republic OKPR**
File type	single-moment/multi-sweep	single-moment/multi-sweep

Table 5: continued

	Poland SOWR	**Czech Republic OKPR**
Reflectivity file pattern	T_PAGZ41_C_SOWR_20220206100005.h5 T_PAGZ41_C_SOWR_20220206101005.h5 . . . T_PAGZ42_C_SOWR_20220206101006.h5 T_PAGZ42_C_SOWR_20220206102006.h5 . . .	T_PAGZ50_C_OKPR_20220206100415.hdf T_PAGZ50_C_OKPR_20220206100914.hdf . . . T_PAGZ60_C_OKPR_20220206100413.hdf T_PAGZ60_C_OKPR_20220206100913.hdf . . .
Radial wind file pattern	T_PAHZ41_C_SOWR_20220206100341.h5 T_PAHZ41_C_SOWR_20220206101342.h5 . . . T_PAHZ42_C_SOWR_20220206100346.h5 T_PAHZ42_C_SOWR_20220206101347.h5 . . .	T_PAHZ50_C_OKPR_20220206100415.hdf T_PAHZ50_C_OKPR_20220206100914.hdf . . . T_PAHZ60_C_OKPR_20220206100413.hdf T_PAHZ60_C_OKPR_20220206100913.hdf . . .
Stations (NR = WMOID = Name)	41 = 12374 = Legionowo 42 = 12514 = Ramza 43 = 12544 = Pastewnik 44 = 12579 = Rzeszow 45 = 12331 = Poznan 46 = 12220 = Swidwin 47 = 12151 = Gdansk 48 = 12568 = Brzuchania	50 = 11718 = czska 60 = 11480 = czbrd
Wavelength	C-Band (we take 0.055 m, but the true wavelengths are slightly different in reality)	C-Band (we take 0.055 m, but the true wavelengths are slightly different in reality)
Elevations (<E>)	**Stations 41-47:** 0.5°, 1.4°, 2.4°, 3.4°, 5.3°, 7.7°, 10.6°, 14.1°, 18.5°, 23.8° **Station 48:** 0.5°, 1.4°, 2.1°, 3.4°, 5.5°, 7.7°, 10.6°, 14.3°, 18.6°, 23.8°	Set of accepted elevations: 0.1°, 0.5°, 0.9°, 1.3°, 1.7°, 2.2°, 3.2°, 4.5°, 6.3°
Remarks	10 min scans only! Might be changed in the future! Scan and elevation identifiers = GZ Up do now only 10' volume scans, but within the 10' there are two types of scans on different elevation and resolution sets, one for DBZH (GZ) starting at minute 0 and another for VRADH (HZ) starting at minute 3. We take the ones for DBZH ONLY, because for the VRADH DATA we have no corresponding reflectivities for quality control. Default: rs_meta(i)%lvrad_to_fdbk = .FALSE.	Scan and elevation identifiers: GZ (reflectivity) and HZ (radial winds) Time in filename is end time of scan! There are three more higher elevations, but these have a smaller range increment and are thus not usable at the moment. Radial winds not usable for data assimilation, because Nyquist velocity is only 7.25 m/s. Default: rs_meta(i)%lvrad_to_fdbk = .FALSE.

Table 6: Some details about the implemented OPERA radar data in ODIM hdf5 format of Slovakia.

	Slovakia LZIB	
File type	single-moment/multi-sweep	

Table 6: continued

	Slovakia LZIB	
Reflectivity file pattern	T_PAGZ41_C_LZIB_20220206100500.hdf T_PAGZ41_C_LZIB_20220206101000.hdf . . . T_PAGZ60_C_LZIB_20220206100500.hdf T_PAGZ60_C_LZIB_20220206101000.hdf . . .	
Radial wind file pattern	T_PAHZ41_C_LZIB_20220206100500.hdf T_PAHZ41_C_LZIB_20220206101000.hdf . . . T_PAHZ60_C_LZIB_20220206100500.hdf T_PAHZ60_C_LZIB_20220206101000.hdf . . .	
Stations (NR = WMOID = Name)	41 = 11812 = skjav 51 = 11958 = skkoj 60 = 11887 = skkub	
Wavelength	C-Band (we take 0.055 m, but the true wavelengths are slightly different in reality)	
Elevations (<E>)	0.0°, 0.5°, 1.0°, 1.5°, 2.0°, 2.7°, 3.4°, 4.4°, 7.0°, 11.4,°, 18.3°, 26.7°	
Remarks	Scan and elevation identifiers: GZ (reflectivity) and HZ (radial winds) Time stamp is the beginning of the nominal 5' min interval. Radial wind is usable. Nyquist velocities are 40.1, 47.6 and 64.2 m/s. The radial winds are put to the feedback files by default, if `loutradwind=.TRUE.` Default: rs_meta(i)%lvrad_to_fdbk = .TRUE.	

5 Namelist parameters

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

The first compartment concerns the "true" EMVORADO volume scan simulation compartment (steps 1 and 2 of the reflectivity- and radial wind operators) and is given in the namelist `/RADARSIM_PARAMS/` in file `INPUT_RADARSIM`. 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 compartment is the "traditional" output of unattenuated Z_h on the model grid as part of the hosting model, which is independent of any "true" observations and uses the output facilities of the hosting model. For example, COSMO and ICON provide a simple reflectivity diagnostic on the model grid based on a very basic Rayleigh-Debye-Approximation, which is independent of EMVORADO. For COSMO, the computation of Z_h for this model output optionally may use the same fortran procedures than step 1 of EMVORADO (e.g., the more accurate Mie- or T-matrix scattering options). For ICON this option is in preparation. 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. When run online in COSMO or when run in the offline-framework, this namelist has to be found in the file `INPUT_RADARSIM`. When run online in ICON, the namelist has to be in file `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, and for the offline-framework, `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 12 in Section 5.1.3 for global parameters applied either generally or to all radar stations,
- Table 13 in Section 5.1.4 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 14 in Section 5.1.5 for the contents of the derived type `t_dbzcalc_params` (configuration parameters for the reflectivity computation), which can be individually 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 declared 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`	140	max. number of radar stations; two different scan strategies for the same station count as two different radar stations!
`ncountry_max`	10	max. number of “countries” (set of defaults for groups of radars with similar properties)
`ndatakind`	10	max. number of different radar observables/moments for input[6]
`nel_composite_max`	10	max. number of different composites (from different radar elevations) produced during one run
`noutput_fields_max`	50	max. number of different radar fields/quantities/moments for volume data output
`ndoms_max`	3	max. number of model domains supported by EMVORADO

The character strings in these tables might have different maximum lengths, and some of them are declared 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:

`cmaxlen`	300	Length of character strings for filenames, directories and error messages
`cobsflen`	300	Length of character strings for observation filenames in `radar_meta_type`
`cvarlen`	10	Length of character strings for names of radar variables in `radar_meta_type`

5.1.1 Additional parameters in the hosting models in case of online-coupling

When used in online-mode in COSMO or ICON, in the hosting model 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 9: 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

[6]About `ndatakind`: This is the maximum number of different data fields contained in the observation input files. Currently, DWD NetCDF-and hdf5 radar files as well as hdf5-files from OPERA data hub at DWD from some European countries are supported, and besides reflectivity Z_e, radial wind v_r and their respective quality flags q_z and q_v, some countries offer a suite of 5 polarization parameters. The Opera standard allows in principle also a quality flag product. Therefore this number has been set to 10.

5.1.2 Additional namelists in the offline framework

When run in the offline-framework, EMVORADO requires additional informations, e.g., on the model grid, that in online mode is passed down directly from the hosting model. This is controlled by the offline-framework's namelists `/REFL_OFFLINE/` and `/GRID_IN/`, to be provided together in file `INPUT_DBZSIM`. Tables 10 and 11 report on their respective parameters and available options.

Table 10: Parameters of namelist `/REFL_OFFLINE/` for offline-mode runs of EMVORADO. Kind abbreviations: "I" = INTEGER, "R" = REAL/DOUBLE, "C" =CHARACTER, "L" = LOGICAL, "T" = Derived TYPE.

Name	**Kind (Dim.)**	**Description / Remarks**	**Default**
inputdir	C (`cmaxlen`)	Name of folder where the model input data (grid description file(s), model state fields, analysis increments) are located.	' '
yinput_format	C (20)	Identifier of format of the model input data files. Allowed: 'ascii' (ASCII), 'apix', 'grib', 'grib2' (GRIB 1+2), 'ncdf', 'ncdf-cosmo' (NETCDF from COSMO), 'ncdf-icon', 'ncdf-hdcp2' (recent and early-phase NETCDF from ICON).	' '
moddata_filename	C (`cmaxlen`)	Name of model data file that contains the model state fields. **Note:** For GRIB format input, filename cannot be identical to `modgrid_filename` (instead, a symlink might be used, though).	' '
model_starttime	C (14)	Start time of the model run as YYYYMMDDhh. Required for proper time reference in EMVORADO and it's outputs.	'2003032100'
forecast_time	C (8)	Forecast time since model start time as DDhhmmss. Required for proper actual time in EMVORADO and it's outputs.	'00000000'
model_name	C (10)	Name of the NWP model, from where the model input data are coming. Required to choose the model-dependent default set of hydrometeor class microphysics settings. Allowed: 'icon', 'cosmo'.	'undefined'
itype_gscp_fwo	I (1)	3-digit identifier of the grid scale cloud and precipitation microphysics. The first digit determines which moment scheme (1- or 2-moments microphysics) is assumed, the second digit specifies the number of hydrometeor classes handled by the scheme (where 2= cloud liquid and rain, 3= 2+cloud ice, 4= 3+snow, 5= 4+graupel, 6= 5+hail). The third digit specifies sub-types of the schemes (0= standard schemes of COSMO respectively ICON). Allowed: 120, 130, 140, 150, 151 (modified snow), 250, 260.	140
linput_q_densities	L (1)	Flag whether model data contains hydrometeor concentrations as densities rho_x (.TRUE.) or mass-specific values q_x (.FALSE.), with $rho_x = \rho q_x$	.FALSE.
outputdir	C (`cmaxlen`)	Name of the root folder for EMVORADO's output. This emulates the NWP model's standard output directory, which in online mode would be taken from the hosting model. It is the anchor point for any relative output paths given in namelist `/RADARSIM_PARAMS/`.	' '
ldebug_refloffline	L (1)	Flag whether to write out some offline-operator specific debug information, e.g. about domain decompositions	.FALSE.

Table 11: Parameters of namelist `/GRID_IN/` for offline mode runs of EMVORADO. Kind abbreviations: "I" = INTEGER, "R" = REAL/DOUBLE, "C" =CHARACTER, "L" = LOGICAL, "T" = Derived TYPE.

Name	**Kind (Dim.)**	**Description / Remarks**	**Default**
modgrid_filename	C (`cmaxlen`)	Name of model data file that contains the model grid, specifically the vertical grid, information. **Note:** For GRIB format input, filename cannot be identical to `moddata_filename` (instead, a symlink might be used, though).	' '
levtyp	C (20)	Type of vertical levels of input data. Allowed: 'eta' (model levels), 'z' (constant height levels), 'p' (constant pressure levels).	'eta'. 'z' and 'p' are not yet fully implemented and currently not functional.
dlon	R (1)	Grid spacing in longitude (ie, x-) direction in degrees.	0.008992893
dlat	R (1)	Grid spacing in latitude (ie, y-) direction in degrees.	0.008992893
startlon_tot	R (1)	Longitude of the lower left (southwest) corner.	-0.5
startlat_tot	R (1)	Latitude of the lower left (southwest) corner.	-0.5
pollon	R (1)	Longitude of the rotated grid north pole.	-180.0
pollat	R (1)	Latitude of the rotated grid north pole.	90.0
ie_tot	I (1)	Number of grid points in longitude (ie x- or i-) direction.	100
je_tot	I (1)	Number of grid points in latitude (ie y- or j-) direction.	100
ke_tot	I (1)	Number of grid points in vertical (ie z- or k-) direction.	50
is_sub_tot	I (1)	The start grid point index in i- (ie, x- or lon-) direction when using a cut-out of the original model domain in the operator (to, e.g. save memory and runtime).	-HUGE(1)
js_sub_tot	I (1)	The start grid point index in j- (ie, y- or lat-) direction when using a cut-out of the original model domain in the operator (to, e.g. save memory and runtime).	-HUGE(1)
ie_sub_tot	I (1)	The last grid point index in i- (ie, x- or lon-) direction when using a cut-out of the original model domain in the operator (to, e.g. save memory and runtime).	-HUGE(1)
je_sub_tot	I (1)	The last grid point index in j- (ie, y- or lat-) direction when using a cut-out of the original model domain in the operator (to, e.g. save memory and runtime).	-HUGE(1)
luv_staggered	L (1)	Flag whether U and V are on a horizontally staggered grid.	.TRUE.
luv_rotated	L (1)	Flag whether U and V are defined relative to the rotated lat-lon grid (.TRUE.) or to the geographic directions (.FALSE.).	.TRUE.
nboundlines	I (1)	Number of grid points overlapping between patches in domain-decomposed parallel processing. Must be ≥ 0	3
nprocx	I (1)	Number of patches (=processors) in x- (ie, i- or lon-) direction, into which the domain is decomposed for parallel processing. The total number of processors for the job is `nprocx` $\times$`nprocy` +`nprocio_radar`.	1
nprocy	I (1)	Number of patches (=processors) in y- (ie, j- or lat-) direction, into which the domain is decomposed for parallel processing. The total number of processors for the job is `nprocx` $\times$`nprocy` +`nprocio_radar`.	1

Table 11: continued

Name	**Kind (Dim.)**	**Description / Remarks**	**Default**
nprocio_radar	I (1)	Number of parallel-computing nodes exclusively dedicated to I/O. The total number of processors for the job is `nprocx` $\times$`nprocy` +`nprocio_radar`.	0

5.1.3 Global namelist parameters in `/RADARSIM_PARAMS/`

Table 12: 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 5.1.6. • 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 12: 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 13below.	Depends on `icountry`, see below in Table 13
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 14 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 14
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, in COSMO, is triggered through the `GRIBOUT` namelist(s). There, the debug mode is triggered by setting `ldebug_io=.TRUE.` in `IOCTL`.	.FALSE.

Table 12: 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 and polarization parameters. Also, observation data processing for reflectivity and some polarization parameters (e.g., for feedback files) is enabled/disabled according to this switch. **If `lvoldata_output=.TRUE.`**, a number of reflectivity- and polarization output variables becomes available for output as volume data (cf. Tab. 15), feedback files or radar composites. **If `lreadmeta_from_netcdf=.TRUE.`**, the corresponding superobservations are also available for output, see also Tab. 15.	.TRUE.
lcalc_dbz_on_radarbins	L (1)	If `.TRUE.` radar moments such as Z_h or polarimetric parameters are computed on radar bins after the model state variables have been interpolated to radar bins in step 1 above. Otherwise, the radar moments are computed on the model grid and are then interpolated to radar bins (default method). **NOTE: this method is not implemented for online beam propagation yet (`lonline=.TRUE.`)!**	.FALSE.
dbz_meta_glob	T (1)	Background value of type `t_dbzcalc_params` for configuring the computation of reflectivity and polarimetric parameters for all radar stations (cf. Section 2.4). The most important type component is `dbz_meta_glob%itype_refl`: 1 = Mie (Blahak, 2016) 3 = Rayleigh-Oguchi (Blahak, 2016) 4 = “Old” Rayleigh from COSMO `pp_utilities.f90` 5 = T-matrix computations assuming oblate spheroids (Ryzhkov et al., 2011) 6 = T-matrix computations assuming spheres (used for cross checking with Mie) The other components are described in Tab. 14, e.g., `dbz_meta_glob%llookup_mie` for switching on the use of efficient lookup tables for Mie- and T-matrix calculations. Every component of the type can be adjusted for each individual station by `dbz_meta(`*i*`)%<component>`.	Same defaults as in Tab. 14
loutpolstd	L (1)	If `loutdbz=.TRUE.` and `dbz_meta(`*i*`)%itype_refl=5,6`, switch to trigger output of polarimetric ”standard” parameters Z_{dr}, ρ_{hv}, K_{dp} and Φ_{dp}, and A_{dp}.	.FALSE.

Table 12: continued

Name	**Kind (Dim.)**	**Description / Remarks**	**Default**
loutpolall	L (1)	If `loutdbz=.TRUE.` and `dbz_meta(`i`)%itype_refl=5,6`, switch to trigger in addition to `loutpolstd` set the output of *LDR*.	.FALSE.
itype_mpipar_lookupgen	I (1)	Relevant if `dbz_meta_glob%llookup_mie=.TRUE.` or any `dbz_meta(`i`)%llookup_mie=.TRUE.`: Flag to choose the parallelization strategy for lookup table generation: 1 = parallelization over the different required `dbz_meta`-sets and hydrometeor types 2 = parallelization over table elements. **Recommended, because better load balance as option 1.**	2
pe_start_lookupgen	I (1)	Relevant for lookup table generation: start ID of the processors which compute the lookup table. Useful, e.g., for ICON on DWD's NEC supercomputer for fine-tuning.	0
pe_end_lookupgen	I (1)	Relevant for lookup table generation: end ID of the processors which compute the lookup table.	COSMO: num_compute-1 ICON: n_workers-1
llookup_interp_mode_dualpol	L (1)	Relevant for actual table lookup. Option to choose the interpolation method with respect to hydrometeor density q or mean size x: 1 = the (historically grown and **default**) previous method: • Z_h, Z_v, Z_{vz}, A_h linear in log-log space • $\mathrm{Re}(\rho_{hv})$, $\mathrm{Im}(\rho_{hv})$, K_{dp}, A_{dp} cubic Hermite spline in lin-lin space 2 = Cubic Hermite spline in lin-lin space for all parameters. **Recommended for polarization parameters!** Interpolation with respect to T and the melting state proxy is always bilinear. Linear in log-log space means that log of the parameter is linearily interpolated with respect to $\log q$ respectively $\log x$. This is the traditional method in EMVORADO, which works well for Z_h and A_h but fails for other polarimetric parameters. The cubic interpolation in lin-lin space avoids interpolation artifacts, e.g., values of $\rho_{hv} > 1$.	1
lextdbz	L (1)	For `dbz_meta(`i`)%itype_refl=1,5,6`: 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.
ydir_mielookup_read	C (`cmaxlen`)	For `dbz_meta(`i`)%itype_refl=1,5,6` 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.	' '

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
ydir_mielookup_write	C (`cmaxlen`)	For `dbz_meta(`*i*`)%itype_refl=1,5,6` 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.
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.

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
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 12: 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 azimuthally scanning radar with axisymmetric 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 azimuthal 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.

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
ydirradarout	C (`cmaxlen`)	Basic 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`. There is the possibility to have separate output subdirectories for different volume data output streams (`voldata_ostream(`k`)%output_subdir`), radar composites (`ysubdircomp`) and feedback files (`ysubdirfof`).	' '
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` or to a subdirectory herein. There is the possibility to define up to n=5 output streams by different elements of the derived type `voldata_ostream(`n`)`. The different components of this type (file format, file name pattern, output variables etc.) are given in the following. It is also possible to limit the output to certain observation times and certain elevations only, but only globally at the moment, i.e., same for all output streams.	.TRUE.
voldata_ostream(n)%format	C (12)	Character string to indicate the desired output format for the n-th output stream, cf. Section 6.1.1. Can be either • `'ascii'`, `'ascii-gzip'`, • `'cdfin'`, `'cdfin-mulmom'`, • `'grib2'`, `'grib2-mulmom'`, or • `'f90-binary'`. `'cdfin-mulmom'` and `'grib2-mulmom'` enable to write multi-moment files, i.e., all radar moments in one file instead of one file per moment, while `'cdfin'` and `'grib2'` produce single moment files. Note that grib2 formats at the moment allow only `'zrsim'` and `'zrobs'`!. Besides having one file per output time step, the cdfin and grib2- formats also enable to write a series of consecutive output time steps ("time batch") to a single output file per radar moment / radar station (setting `voldata_ostream(`n`)%content_tref` and `voldata_ostream(`n`)%content_dt` appropriately below). **The recommended formats are `'cdfin'` or `'cdfin-mulmom'`.**	'ascii'
voldata_ostream(n)%grib2_packingtype	C (12)	Only effective if `voldata_ostream(`n`)%format = 'grib2'` or `'grib2-mulmom'`: character string to indicate the desired grib2 packing type (compression) for the n-th output stream, cf. Section 6.1.1. Can be either • `'grid_simple'` (no compression) • `'grid_ccsds'` (compression from the aec lib - `eccodes` must be compiled with aec lib), or • `'png'` (compression from the png lib - `eccodes` must be compiled with png lib).	'grid_ccsds'

Table 12: continued

Name	**Kind (Dim.)**	**Description / Remarks**	**Default**
voldata_ostream(*n*)%output_subdir	C (`cmaxlen`))	Subdirectory under `ydirradarout` for the storage of volume scan files. If it starts with a “/”, it is taken as an absolute path instead.	’ ’

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
voldata_ostream(*n*)%file_pattern	C (`cmaxlen`)	File pattern for *n*-th output stream for file formats `'cdfin'`, `'cdfin-mulmom'`, `'grib2'`, `'grib2-mulmom'`, `'ascii'`, `'ascii-gzip'`, `'f90-binary'`. This enables a flexible definition of output file names as a mix of arbitrary text parts and the following placeholders: • `<stationid>` (mandatory): will result in e.g. `'id-010908'` • `<varname>`: parameter identifier Tab. 15. If `voldata_ostream(`*n*`)%format='*-mulmom'`, this will result in `'allobs'` or `'allsim'` or `'allgeom'`. • `<tmodelini>` (optional): absolute datetime of model start in format YYYYMMDDhhmmss • `<tmodelini_mm>` (optional): same, but in format YYYYMMDDhhmm without seconds • `<tstart>` (only cdfin- and grib2-formats): absolute date and time of start of time window of file content in format YYYYMMDDhhmmss • `<tstart_mm>`: same, but in format YYYYMMDDhhmm without seconds • `<tend>` (only cdfin- and grib2-formats): absolute date and time of end of time window of file content in format YYYYMMDDhhmmss • `<tend_mm>`: same, but in format YYYYMMDDhhmm without seconds • `<tact>`: absolute date and time of actual model time in format YYYYMMDDhhmmss • `<tact_mm>`: same, but in format YYYYMMDDhhmm without seconds • `<tvvzstart>` (only cdfin- and grib2-formats): forecast lead time of start of time window of file content in format DDhhmmss • `<tvvzend>` (only cdfin- and grib2-formats): forecast lead time of end of time window of file content in format DDhhmmss • `<tvvzact>`: forecast lead time of actual model time in format DDhhmmss • `<scantype>` (mandatory): identifier for the scantype. If not specified, it will be `rs_meta(i)%scanname`, e.g. `'PPI0800'`. If the files should later be used as OSSE nature run input, the keywords `'volscan'` or `'precipscan'` are needed. For this, set `voldata_ostream(`*n*`)%pat_scantype_precipscan` and `voldata_ostream(`*n*`)%pat_scantype_volscan` to these names. • `<dom>`: identifier for the model domain, will result in e.g. `'01'`. As indicated, some of these are optional and some are mandatory depending on the format. If there is no indication above, these rules apply: - `<tmodelini>` is optional. From the other time keys at least one is mandatory, no matter which. - `<varname>` is mandatory for `'cdfin'`, `'grib2'` and `'ascii*'` and optional for `'cdfin-mulmom'`, `'grib2-mulmom'`. - `<dom>` is optional, but may become mandatory if there is more than one radar-active model domain and if the output of at least two domains/ output streams goes into the same output directory. (continued in next row)	(see next row!)

Table 12: continued

Name	**Kind (Dim.)**	**Description / Remarks**	**Default**
voldata_ostream(n)%file_pattern	C (`cmaxlen`)	EMVORADO will throw an appropriate error message about missing mandatory keys. Time strings for ascii- and f90-binary contain also seconds, i.e., YYYYMMDDhhmmss. **The default patterns for cdfin and grib2 are** `'cdfin_<varname>_<stationid>_<tstart_mm>_<tend_mm>_<scantype>'` `'grib2_<varname>_<stationid>_<tstart_mm>_<tend_mm>_<scantype>'` **The default patterns for ascii and f90-binary are** `'<varname>_<stationid>_<scantype>_<tmodelini>_<tvvzact>_<levtype>.dat` (file extension for f90-binary is `.bin`)	(see to the left)
voldata_ostream(n)%pat_scantype_volscan	C (12)	Alternative identifier for volume scans for the placeholder `<scantype>` in `voldata_ostream(`n`)%file_pattern`.	'volscan'
voldata_ostream(n)%pat_scantype_precipscan	C (12)	Alternative identifier for DWD's precipitation scans for the placeholder `<scantype>` in `voldata_ostream(`n`)%file_pattern`.	'precipscan'
voldata_ostream(n)%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.`.	' '
voldata_ostream(n)%content_tref	R (1)	For `'cdfin'`, `'cdfin-mulmom'`, `'grib2'`, `'grib2-mulmom'` output files: reference time [s] for time batches. `cdfin_tref=` 0.0 means that batches are synchronized to the model start time	0.0
voldata_ostream(n)%content_dt	R (1)	For `'cdfin'`, `'cdfin-mulmom'`, `'grib2'`, `'grib2-mulmom'` 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.**	1E6
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 13). 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

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
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
dt_obs_voldata_glob	R (3)	Time triplet `from-time,to-time,increment` for desired volume data output [s since nominal model start]. Active only if `obs_times_voldata_glob(:)=-999.99`. Is used to build the list of desired output times in equal steps from `from-time` to `to-time`. If `from-time` is less than -900, the series of times starts (approximately) at model init time — which might be negative for Incremental Analysis Update (IAU) runs — and is synchronized to time 0.0 s. If `from-time` is larger than -900, it is interpreted as the exact start time of the series and the latter is not synchronized to time 0.0 s. However, for backwards compatibility, one can also give only one value in the namelist, which is interpreted as the increment and automatically shifted to the correct position in the triplet. `from-time` and `to-time` are set to -999.9 in this case. Default: missing values to indicate that all existing `rs_meta(`*i*`)%obs_times` should be output.	-999.9
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 (`cmaxlen`)	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 (`cmaxlen`)	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.	' '

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
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.
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 azimuths 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 azimuths 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 (fof) files. Only possible for `lreadmeta_from_netcdf=.TRUE.`. The files will be stored under `ydirradarout` or optionally the subdirectory `ysubdirfof` herein.	.FALSE
ysubdirfof	C (`cmaxlen`)	Subdirectory under `ydirradarout` for the storage of feedback files. If it starts with a "/", it is taken as an absolute path instead.	' '
fof_file_pattern	C (`cmaxlen`)	File pattern for feedback (fof) output files. This enables a flexible definition of fof file names as a mix of arbitrary text parts and the following placeholders: • `<stationid>`: will result in e.g. `'id-010908'` • `<scantype>`: identifier for the scantype. If not specified, it will be `rs_meta(i)%scanname`, e.g. `'PPI0800'`. • `<tmodelini>`: absolute date and time of model start in format YYYYMMDDhhmmss • `<dom>`: identifier of the model domain, will result in e.g. `'01'`. `<stationid>` and `<scantype>` are mandatory. `<dom>` is optional, but may become mandatory if there is more than one radar-active model domain and if the output of at least two domains goes into the same output directory. EMVORADO will throw an appropriate error message about missing mandatory keys. **The default pattern is** `'fof_radar_<stationid>_<scantype>_<tmodelini>.nc'`.	(see left)

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
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 13). 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 (3)	Time triplet `from-time,to-time,increment` for writing radar data to feedback files [s since nominal model start]. Active only if `obs_times_fdbk_glob(:)=-999.99`. Is used to build the list of desired output times in equal steps from `from-time` to `to-time`. If `from-time` is less than -900, the series of times starts (approximately) at model init time — which might be negative for Incremental Analysis Update (IAU) runs — and is synchronized to time 0.0 s. If `from-time` is larger than -900, it is interpreted as the exact start time of the series and the latter is not synchronized to time 0.0 s. However, for backwards compatibility, one can also give only one value in the namelist, which is interpreted as the increment and automatically shifted to the correct position in the triplet. `from-time` and `to-time` are set to -999.9 in this case. **Note that all times without the presence of actual observations are removed from the list.** Default: missing values 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 symmetric 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: azimuthal 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 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
supob_ave_width_vr	R (1)	Width of averaging area for superobing for v_r [m].	$\sqrt{2}\times$ supob_cart_resolution_d
supob_ave_width_z	R (1)	Width of averaging area for superobing for Z [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_vr_d
supob_minrange_z	R (1)	Min. range of superobing points for Z [m].	$0.75\times$ supob_ave_width_z_d
supob_azi_maxsector_vr	R (1)	Maximal azimuth sector (symmetrical 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.	$\sqrt{\mathrm{HUGE(1.0\_dp)}}$
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 12: continued

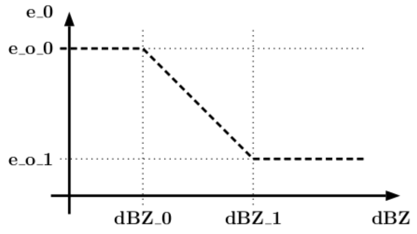
Name	Kind (Dim.)	Description / Remarks	Default
itype_obserr_vr	I (1)	Type of computation of the observation error for radial wind to be stored in the NetCDF feedback files (fof) for data assimilation, for `lreadmeta_from_netcdf=.TRUE.` and `loutradwind=.TRUE.`: 0 = constant value from namelist parameter `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 12: 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 metadata 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.
labort_if_problems_gribout	L (1)	If .TRUE., abort the model run if problems occur when writing grib output files (composites, volume scans) to disk by eccodes methods. In very few cases eccodes has issued an “Input/Output error” at runtime and the grib file could not be written correctly. The default is to abort in such a case, 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.

Table 12: 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 for each time for which at least one of the radars has an observation timestep in the list `rs_meta(`*i*`)%obs_times`. 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 output any arbitrary lat-lon grids. The following namelist parameters in the derived type `comp_meta` define the composite grid independently from the model grid. Note that the default in case of COSMO is the COSMO-model grid itself.	.FALSE.
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` in its subdirectory `ysubdircomp`. See also Sections 2.7 and 5.1.7 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.
ysubdircomp	C (`cmaxlen`)	Subdirectory under `ydirradarout` for the storage of radar composites. If it starts with a "/", it is taken as an absolute path instead.	' '

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
composite_file_pattern	C (`cmaxlen`)	File pattern for composite output files. This enables a flexible definition of output file names as a mix of arbitrary text parts and the following placeholders: • `<varname>` (optional): parameter identifier, will result in `'dbzcmp'` • `<simobs>` (optional): `'sim'` or `'obs'`, to be able to discriminate files with simulated and observed composites • `<tmodelini>` (optional): absolute date and time of model start in format YYYYMMDDhhmmss • `<tact>`: absolute model forecast date and time in format YYYYMMDDhhmmss • `<tvvzact>`: relative forecast date and time in format DDhhmmss • `<dom>`: identifier of the model domain, will result in e.g. `'01'`. One of `<tact>` and `<tvvzact>` is mandatory, the respective other is then optional. `<dom>` is optional, but may become mandatory if there is more than one radar-active model domain and if the output of at least two domains goes into the same output directory. EMVORADO will throw an appropriate error message about missing mandatory keys. **The default pattern is** `'<varname>_<simobs>_<tmodelini>_<tact>.grb2'`.	(see left)

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 = 651

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 = 716

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 = -7.5

comp_meta%startlat	R (1)	Rotated lat/lon grid for the reflectivity composites: lower left corner latitude in rotaded coordinates [°].	COSMO = startlat_tot
			ICON = -6.3

comp_meta%dlon	R (1)	Rotated lat/lon grid for the reflectivity composites: angular resolution [°] in rotated meridional direction.	COSMO = dlon
			ICON = 0.02

comp_meta%dlat	R (1)	Rotated lat/lon grid for the reflectivity composites: angular resolution [°] in rotated zonal direction.	COSMO = dlat
			ICON = 0.02

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 12: 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 14). 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

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
lcomposite_output_bub	L (1)	Only effective if `ldo_bubbles=.TRUE.`: if `lcomposite_output_bub=.TRUE.` EMVORADO uses it's own grib2-output facilities to write radar reflectivity composites to the output directory `ydirradarout` in its subdirectory `ysubdircomp`. See also Sections 2.7 and 5.1.7 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_bub=.TRUE.` is the only possibility to output the composites.**	.FALSE.
composite_file_pattern_bub	C (`cmaxlen`)	File pattern for bubble generator composite output files. This enables a flexible definition of output file names as a mix of arbitrary text parts and the following placeholders: • `<varname>` (optional): parameter identifier, will result in `'dbzcmpbub'` • `<simobs>` (optional): `'sim'` or `'obs'`, to be able to discriminate files with simulated and observed composites • `<tmodelini>` (optional): absolute date and time of model start in format YYYYMMDDhhmmss • `<tact>`: absolute model forecast date and time in format YYYYMMDDhhmmss • `<tvvzact>`: relative forecast date and time in format DDhhmmss • `<dom>`: identifier of the model domain, will result in e.g. `'01'`. One of `<tact>` and `<tvvzact>` is mandatory, the respective other is then optional. `<dom>` is optional, but may become mandatory if there is more than one radar-active model domain and if the output of at least two domains goes into the same output directory. **The default pattern is `'<varname>_<simobs>_<tmodelini>_<tact>.grb2'`.**	(see left)

comp_meta_bub%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 = 651

comp_meta_bub%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 = 716

comp_meta_bub%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_bub%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_bub%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

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default

comp_meta_bub%startlon	R (1)	Rotated lat/lon grid for the reflectivity composites: lower left corner longitude in rotaded coordinates [°].	COSMO = startlon_tot
			ICON = -7.5

comp_meta_bub%startlat	R (1)	Rotated lat/lon grid for the reflectivity composites: lower left corner latitude in rotaded coordinates [°].	COSMO = startlat_tot
			ICON = -6.3

comp_meta_bub%dlon	R (1)	Rotated lat/lon grid for the reflectivity composites: angular resolution [°] in rotated meridional direction.	COSMO = dlon
			ICON = 0.02

comp_meta_bub%dlat	R (1)	Rotated lat/lon grid for the reflectivity composites: angular resolution [°] in rotated zonal direction.	COSMO = dlat
			ICON = 0.02

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 symmetric 2D binomial smoother in grid points. Not yet in the namelist, up to now hardcoded in `radar_namelist_read.f90`	9
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
tstart_bubble_search	R (1)	For `ldo_bubbles=.TRUE.`: start time for the bubble generator [s since model start].	0.0
dt_bubble_search	R (1)	For `ldo_bubbles=.TRUE.`: time interval from one automatic bubble search to the next [s].	900.0
tend_bubble_search	R (1)	For `ldo_bubbles=.TRUE.`: end time for the bubble generator [s since model start].	HUGE(1.0_dp)
t_offset_bubble_trigger_async	R (1)	For `ldo_bubbles=.TRUE.` and in case of asynchronous 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/)

Table 12: continued

Name	Kind (Dim.)	Description / Remarks	Default
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
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 [$^\circ$]	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

Table 12: continued

Name	**Kind (Dim.)**	**Description / Remarks**	**Default**
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` per default are written into the output directory for READY-files 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. The name of the ready file can be configured by `ready_file_pattern`, see below.	.FALSE.
ydir_ready_write	C (`cmaxlen`)	For `lwrite_ready=.TRUE.`: absolute path for writing READY-files to disk. If empty, READY-files are written to `ydirradarout`.	' '
ready_file_pattern	C (`cmaxlen`)	For `lwrite_ready=.TRUE.`: File pattern for READY-files. This enables a flexible definition of READY filenames as a mix of arbitrary text parts and the following placeholders: • `<tmodelini>` (optional): absolute date and time of model start in format YYYYMMDDhhmmss • `<tact>`: absolute model forecast date and time in format YYYYMMDDhhmmss • `<tvvzact>`: relative forecast date and time in format DDhhmmss • `<dom>`: identifier of the model domain, will result in e.g. `'01'`. One of `<tact>` and `<tvvzact>` is mandatory, the respective other is then optional. `<dom>` is optional, but may become mandatory if there is more than one radar-active model domain and if the ready files of at least two domains go into the same output directory. **The default pattern is** **`'READY_EMVORADO_<tact>'`.**	(see left)
rain2mom_mu_incloud	R (1)	Experimental **(do not use unless you know what you are doing!)**. Value to overwrite with the shape parameter μ of rain drop size distribution (ie exponent of drop diameter D), which otherwise is provided by the host model. For `rain2mom_mu_incloud` <=-1.0, host model value is kept. Only applied with 2-moment microphysics scheme.	-999.9

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

Table 13: 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 13: 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

Table 13: continued

Name rs_meta(i)%	**Kind (Dim.)**	**Description / Remarks**	**Default**
%dt_obs	R (3)	Time triplet `from-time,to-time,increment` for building the list of regularily spaced observation times [s since nominal model start time]. Only relevant if no `rs_meta(`i`)%obs_times` ≥ 0.0 are given. The list of `rs_meta(`i`)%obs_times` is computed in intervals of dt_obs starting from `from-time` until `rs_meta(`i`)%nobs_times` steps, or, if `nobs_times` is not set, until `to-time`. If `from-time` is given as a missing value, the series of obs times starts approximately at the nominal model start time — which might be negative for IAU runs — and is synchonized to time 0.0 s. For backwards compatibility, one can also give only one value in the namelist, which is interpreted as the increment and automatically shifted to the correct position in the triplet. `from-time` and `to-time` are set to -999.9 in this case. Note: the default is given in this old notation, only one value!	300.0
%nobs_times	I (1)	Number of observation times. Only relevant if no `rs_meta(`i`)%obs_times` ≥ 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`.	-999
%lobstimes_ovwrt_recalc	I (1)	Only relevant if `lreadmeta_from_netcdf=.TRUE.` and station metadata from obs files are overwritten by namelist (cf. Section 5.1.6). In this case, enable re-calculation of `rs_meta(`i`)%obs_times` from `rs_meta(`i`)%dt_obs` and `rs_meta(`i`)%nobs_times` as described above. Only effective if no `rs_meta(`i`)%obs_times` ≥ 0.0 are given.	`.FALSE.`
%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

Table 13: continued

Name rs_meta(i)%	Kind (Dim.)	Description / Remarks	Default
%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
%obsfile	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'
%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

Table 13: continued

Name rs_meta(i)%	Kind (Dim.)	Description / Remarks	Default
%smth_interv_fact	R (1)	Factor to determine the azimuthal 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
%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_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 into the 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_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 any other 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

Table 13: continued

Name rs_meta(i)%	Kind (Dim.)	Description / Remarks	Default
%dt_obs_voldata	R (3)	Time triplet `from-time,to-time,increment` for desired volume data output of this station [s since nominal model start]. Active only if `rs_meta(`i`)%obs_times_voldata(:)=-999.99`. Is used to build the list of desired output times in equal steps from `from-time` to `to-time`. If `from-time` is less than -900, the series of times starts (approximately) at model init time — which might be negative for IAU runs — and is synchronized to time 0.0 s. If `from-time` is larger than -900, it is interpreted as the exact start time of the series and the latter is not synchronized to time 0.0 s. For backwards compatibility, one can also give only one value in the namelist, which is interpreted as the increment and automatically shifted to the correct position in the triplet. `from-time` and `to-time` are set to -999.9 in this case. Takes precedence over the global parameter `dt_obs_voldata_glob`. If empty or -999.99,-999.99,-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 or the range in the `rs_meta(`i`)%dt_obs_voldata` triplet.	-999
%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 into the 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`
%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 13: continued

Name rs_meta(i)%	**Kind (Dim.)**	**Description / Remarks**	**Default**
%dt_obs_fdbk	R (3)	Time triplet for `from-time,to-time,increment` for writing radar data to feedback files for this station [s since nominal model start]. Active only if `rs_meta(i)%obs_times_fdbk(:)=-999.99`. Is used to build the list of desired output times in equal steps from `from-time` to `to-time`, or, if given, for `rs_meta(i)%nobs_times_fdbk` steps. If `from-time` is less than -900, the series of times starts approximately at model init time — which might be negative for IAU runs — and is synchronized to time 0.0 s. If `from-time` is larger than -900, it is interpreted as the exact start time of the series and the latter is not synchronized to time 0.0 s. For backwards compatibility, one can also give only one value in the namelist, which is interpreted as the increment and automatically shifted to the correct position in the triplet. `from-time` and `to-time` are set to -999.9 in this case. **Note that all times without the presence of actual observations are removed from the list.** Takes precedence over the global parameter `dt_obs_fdbk_glob`. If empty or -999.99,-999.99,-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 or the range in the `rs_meta(i)%dt_obs_fdbk` triplet.	-999
%lvrad_to_fdbk	L (1)	If `loutradwind=.TRUE.`: Switch to decide whether radial winds should be written to the feedback files for this station i. Depending on the countrie's typical Nyquist-velocity, radial winds might not be usable because of severe aliasing.	-999
%vnyq_min_for_vr_active_fdbk	R (1)	If `loutradwind=.TRUE.` and `rs_meta(i)%lvrad_to_fdbk=.TRUE.`: Threshold of (elevation- or time-dependend) Nyquist velocity for accepting radial winds in feedback files as `ACTIVE`. This gives the possibility, for a station i with alternating Nyquist-velocities from elevation to elevation or from time to time, to filter out the "bad" radial winds with Nyquist velocity below the threshold by setting them to `PASSIVE`.	25.0
%ldbzh_to_fdbk	L (1)	If `loutdbz=.TRUE.`: Switch to decide whether reflectivities should be written to the feedback files for this station i. Depending on the data quality, this helps to eliminate stations with generally "bad" data quality.	-999
%obs_hdf5_varname_vrad	C (`cvarlen`)	Name of radial wind dataset in hdf5 input files. Might depend on country.	'VRAD' or 'VRADH'
%obs_hdf5_varname_dbhz	C (`cvarlen`)	Name of reflectivity dataset in hdf5 input files. Might depend on country.	'DBZH'
%obs_hdf5_varname_zdr	C (`cvarlen`)	Name of differential reflectivity dataset in hdf5 input files. Might depend on country.	'ZDR'
%obs_hdf5_varname_rhohv	C (`cvarlen`)	Name of Cross-correllation coefficient dataset in hdf5 input files. Might depend on country.	'RHOHV' or 'URHOHV'
%obs_hdf5_varname_kdp	C (`cvarlen`)	Name of differential phase shift dataset in hdf5 input files. Might depend on country.	'KDP'

Table 13: continued

Name rs_meta(i)%	**Kind (Dim.)**	**Description / Remarks**	**Default**
%obs_hdf5_varname_phidp	C (`cvarlen`)	Name of total differential phase dataset in hdf5 input files. Might depend on country.	'PHIDP' or 'UPHIDP'
%obs_hdf5_varname_ldr	C (`cvarlen`)	Name of linear depolarisation ratio dataset in hdf5 input files. Might depend on country.	'LDR'
%obs_hdf5_varname_cflags	C (`cvarlen`)	Name of quality flag dataset in hdf5 input files. Might depend on country. Is not present for most of the countries at the moment.	'CFLAGS'

5.1.5 Reflectivity config in namelist parameter `dbz_meta(`i`)` in `/RADARSIM_PARAMS/`

This namelist parameter is of derived type `t_dbzcalc_params` (`radar_data.f90`). It serves to hold the parameter definitions of step 1 of the reflectivity operator and is an extended version of the older version of that type described in Sections 6.3 to 6.5 of Blahak (2016). Updates are:

- The parameters for the deprecated option `dbz_meta(`i`)%itype_refl=2` are inactive and should not be used any more. They have been left out in the below table.
- The new components `dbz_meta(`i`)%llookup_mie` and `dbz_meta(`i`)%lhydrom_choice_testing`
- A number of parameters have been added to describe the settings for the EMVORADO melting scheme (Section 6.1 of Blahak, 2016).
- A slightly simplified version of the melting scheme has been introduced for efficiency reasons as an option by the new parameter `dbz_meta(`i`)%itype_Dref_fmelt`:

Option 1 is the current scheme, where the reference particle diameter D_{ref} for the size scaling of the spectral degree of melting in Eq. (29) of Blahak (2016) depends on the mean size of the PSD (“dynamic”).

Option 2 sets constant values for D_{ref} depending on the hydrometeor type. At the same time the exponent a has been adjusted in order to approximately reproduce the results from option 1 for “typical” mean particle sizes. Of course this cannot be perfect, but in light of the physical uncertainties associated with melting, this simplification seems to be justified. Its big technical advantage is that it enables to greatly reduce the number of necessary T-matrix calls during lookup table generation for dual polarization (`dbz_meta(`i`)%itype_refl=5,6`) and speeds up this process from several hours to a few minutes.

- A number of parameters have been added to control the assumptions on shape and orientation of non-spherical hydrometeors in dual polarization simuations (specifically for `dbz_meta(`i`)%itype_refl=5`).
- The new option `ldynamic_wetgrowth_gh` to dynamically determine the T-limits T_{mb} in Eq. (29) of Blahak (2016) for melt start of graupel and hail in each grid colunm (i, j) instead of using the fixed namelist parameters `dbz_meta(`i`)%Tmeltbegin_g/h` and `dbz_meta(`i`)%meltdegTmin_g/h`. For both graupel and hail, the uppermost height is searched for which the mean mass diameter D_m of the particle size distribution is larger than the wet growth diameter $D_{wg} = fct(T, p, qc+qr, qi)$. If $D_m > D_{wg}$, the majority of the particles are in wet growth mode and particles are assumed wet at and below that height.

  In case of using lookup tables for Mie- or T-matrix scattering, we do not add a further dimension to the tables, but approximate by an efficient interpolation between two versions L and L_{T0C} of the existing lookup tables,

$$\begin{aligned} L:&\ \text{LUT for } T_{mb} = \texttt{dbz\_meta(}i\texttt{)\%Tmeltbegin\_x} \text{ and } f_{tmin} = \texttt{dbz\_meta(}i\texttt{)\%meltdegTmin\_x} \\ L_{T0C}:&\ \text{LUT for } T_{mb} = T_{0C} \text{ and } f_{tmin} = 0 \quad . \end{aligned}$$

  where x denotes grapuel (g) or hail (h) and $T_{0C} = 273.15\,\text{K}$. L represents the most extreme allowed wet growth settings, and the default values for `Tmeltbegin_g/h` and `meltdegTmin_g/h` are appropriate.

  Let Z be any linearly additive radar moment (e.g. linear horizontal or vertical reflectivity, $RHOHV_r$, $RHOHV_i$) and LZ its value from the lookup table L as well as

$$\begin{aligned} a &= \frac{\max(T_{mb}(i,j) - \texttt{dbz\_meta(}i\texttt{)\%Tmeltbegin\_x},\ 0)}{\max(T_{0C} - \texttt{dbz\_meta(}i\texttt{)\%Tmeltbegin\_x},\ 1e-6)} \\ b &= \frac{\max(T_{0C} - T,\ 0)}{\max(T_{0C} - T_{mb}(i,j),\ 1e-6)} \\ T_{shift} &= T \ - \ T_{mb}(i,j) \ + \ \texttt{dbz\_meta(}i\texttt{)\%Tmeltbegin\_x} \end{aligned}$$

  then

$$Z = \begin{cases} b\,LZ(T_{shift}) \ + \ a(1-b)\,LZ_{T0C}(T_{0C}) \ + \ (1-a)(1-b)\,LZ(T) & \text{for } T_{mb}(i,j) < T < T_{0C} \\ a\,LZ_{T0C}(T) \ + \ (1-a)\,LZ(T) & \text{for } T \geq T_{0C} \end{cases}$$

This exploits the linear behaviour of f_{melt} with T for $T < T_{0C}$ if $T_{mb} < T_{0C}$ and behaves asymptotically correct for $T_{mb} \rightarrow$ `dbz_meta(i)%Tmeltbegin_x` and $T_{mb} \rightarrow T_{0C}$. The error compared to a full Mie calculation was found to be smaller than a few %.

Table 14: 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 `t_dbzcalc_params` (module `radar_dbzcalc_params_type.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` 5 = T-matrix computations assuming oblate spheroids (Ryzhkov et al., 2011) 6 = T-matrix computations assuming spheres (used for cross checking with Mie)	`3`
`%llookup_mie`	`L (1)`	Switch to enable the use of efficient lookup tables for Mie or T-matrix scattering. Only effective if `itype_refl=1,5,6`.	`.TRUE.`
`%igraupel_type`	`I (1)`	Type of melting graupel particle model for Mie or T-matrix Scattering `dbz_meta(i)%itype_refl=1,5,6`: 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`
`%itype_Dref_fmelt`	`I (1)`	Type of melting scheme (see above) for Mie or T-matrix Scattering `dbz_meta(i)%itype_refl=1,5,6`: 1 = default "dynamic" D_{ref} scheme 2 = simplified scheme with fixed D_{ref}	`1`
`%ext_tune_fac_pure`	`R (1)`	Tuning factor for attenuation coefficients of pure water drops and dry ice particles. Only effective if `dbz_meta(i)%itype_refl=5,6` and `lextdbz=.TRUE.`	`1.0`
`%ext_tune_fac_melt`	`R (1)`	Tuning factor for attenuation coefficients of melting hydrometeors. Only effective if `dbz_meta(i)%itype_refl=5,6` and `lextdbz=.TRUE.`	`1.0`
`%ctype_dryice_mie`	`C (3)`	String for defining the EMA of the dry cloud ice category. Particles are assumed to be one-layered spheres of ice-air mixtures. The 3 characters are accoding to Table 38 in Section 6.4 of Blahak (2016). Only effective if `dbz_meta(i)%itype_refl=1,5,6`.	`'mis'`
`%ctype_wetice_mie`	`C (6)`	String for defining the EMA of the melting cloud ice category. 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). Only effective if `dbz_meta(i)%itype_refl=1,5,6`.	`'mawsms'`

Table 14: continued

Name dbz_meta(i)%	Kind (Dim.)	Description / Remarks	Default
`%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=1,5,6`.	`'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=1,5,6`.	`'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=1,5,6`.	`'mis'`
`%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=1,5,6`.	`'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=1,5,6`.	`'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=1,5,6`.	`'mws'`

Table 14: continued

Name dbz_meta(i)%	Kind (Dim.)	Description / Remarks	Default
`%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_i`	`R (1)`	Temperature [K], above which cloud ice is assumed wet. Cf. T_{mb} in Eq. (29) of Blahak (2016)	`273.15`
`%meltdegTmin_i`	`R (1)`	Degree of cloud ice melting at T=273.15 K. Cf. f_{tmin} in Eq. (29) of Blahak (2016)	`0.0`
`%Tmax_min_i`	`R (1)`	For cloud ice the lower bound for T_{max} [K] in Eq. (29) of Blahak (2016)	`275.15`
`%Tmax_max_i`	`R (1)`	For cloud ice the upper bound for T_{max} [K] in Eq. (29) of Blahak (2016)	`278.15`
`%Tmeltbegin_s`	`R (1)`	Temperature [K], above which snow is assumed wet	`273.15`
`%meltdegTmin_s`	`R (1)`	Degree of snow melting at T=273.15 K	`0.0`
`%Tmax_min_s`	`R (1)`	For snow the lower bound for T_{max} [K]	`276.15`
`%Tmax_max_s`	`R (1)`	For snow the upper bound for T_{max} [K]	`283.15`
`%Tmeltbegin_g`	`R (1)`	Temperature [K], above which graupel is assumed wet	`263.15`
`%meltdegTmin_g`	`R (1)`	Degree of graupel melting at T=273.15 K	`0.2`
`%Tmax_min_g`	`R (1)`	For graupel the lower bound for T_{max} [K]	`276.15`
`%Tmax_max_g`	`R (1)`	For graupel the upper bound for T_{max} [K]	`278.15`
`%Tmeltbegin_h`	`R (1)`	Temperature [K], above which hail is assumed wet	`263.15`
`%meltdegTmin_h`	`R (1)`	Degree of hail melting at T=273.15 K	`0.2`
`%Tmax_min_h`	`R (1)`	For graupel the lower bound for T_{max} [K]	`278.15`
`%Tmax_max_h`	`R (1)`	For graupel the upper bound for T_{max} [K]	`303.15`
`%ldynamic_wetgrowth_gh`	`L (1)`	If .TRUE., option for dynamic determination of T_{mb} in each grid column instead of using the fixed namelist parameters `%Tmeltbegin_g/h` and `%meltdegTmin_g/h` . See Section 5.1.5 for more details.	.FALSE.
`%polMP_x`	`T`	Structure to hold hydrometeor shape and orientation settings for individual hydrometeor types `x` with `x = r,i,s,g,h`. Hydrometeors are generally modelled as homogeneous oblate spheroids. Their shape is described by the aspect ratio (AR), defined here as the ratio of the semi-minor and semi-major axes of the spheroids. The spheroids are assumed to have no preferential orientation, with their maximum cross section parallel to the horizon on average (i.e. $\mu_\alpha = 0°$) and with canting angles α out of the horizontal following a Gaussian distribution with a specified width σ_α. Only effective if `dbz_meta(i)%itype_refl=5`.	

Table 14: continued

Name dbz_meta(i)%	Kind (Dim.)	Description / Remarks	Default
`%polMP_x%ARmodel`	`C (8)`	String identifier of the shape model. Available are user-defined hydrometeor size dependent polynomials (`ARmodel='Poly'`) as well as "full model" (including specification of orientation) and shape model parametrizations from literature, typically identified by first letter of author name, two digits for the publication's year and possible further characters identifying variants. The current options are available at the time of writing (see `aspectratio_singlephase` in `radar_mie_-specint.f90` for the complete implementation): `'Poly'` = First order polynomial in particle size (in terms of maximum diameter D [m]) with a lower bound, i.e. aspect ration AR = $max(\texttt{c0} + \texttt{c1} \cdot D, \texttt{ARmin})$. `'R11'` = Hydrometeor-type specific "full model" parametrization according to Ryzhkov et al. (2011). Internal `sig` can be overruled by user-specified one. `'BPRO'` = Hydrometeor-type specific "full model" parametrization as implemented as default in the Bonn Polarimetric Radar forward Operator (B-PRO; Xie et al., 2021, Trömel et al., 2021). Internal `sig` can be overruled by user-specified one. `'B02r'` = Rain intended aspect ratio according to Brandes et al. (2002). `'A13iP'` = Cloud ice intended density-size-relation based aspect ratio according to Andrić et al. (2013), plates variant. `'A13iD'` = Cloud ice intended density-size-relation based aspect ratio according to Andrić et al. (2013), dendrites variant. `'M96iTP'` = Cloud ice intended aspect ratio according to Matrosov et al. (1996), solid thick plates variant. The identifier is case insensitive. There is currently no internal check that parametrizations intended for specific hydrometeor types are exclusively applied for these (i.e., the user has the freedom to apply `A13i*` also for other frozen hydrometeor types or even for rain). However, from the "full model" parametrizations always the type specific parametrization is internally applied (i.e., there is currently no mechanism that allows the user to apply e.g. graupel-specific `R11` to snow).	`'R11'`
`%polMP_x%c0`	`R (1)`	Only applied if `ARmodel='Poly'`. The offset of the first order polynomial.	`0.0`
`%polMP_x%c1`	`R (1)`	Only applied if `ARmodel='Poly'`. The slope of the first order polynomial.	`0.0`
`%polMP_x%ARmin`	`R (1)`	Only applied if `ARmodel='Poly'`. The lower bound of the polynomial, lower values are reset to `ARmin`.	`0.0`
`%polMP_x%sig`	`R (1)`	Needs to be explicitly specified unless a "full model" parametrization is applied. Width σ_α of the assumed Gaussian distribution of canting angles α.	`-99.0`

5.1.6 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 two parameters `rs_meta(`i`)%station_id` and `rs_meta(`i`)%scanname`, because internally in EMVORADO different scan strategies of the same station (e.g., having DWD volume scans and precipitation scans in one run) are treated as two different stations. For this matching, 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 13. 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 = 265.15,

rs_meta(2)%station_id = 10950,
rs_meta(2)%scanname   = 'PPI0080',
dbz_meta(2)%Tmeltbegin_g = 265.15,
```

`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.

If there is a block with a `dbz_meta(`*i*`)%station_id` and `rs_meta(`*i*`)%scanname` that is not contained in any of the observation files, this station is added as an additional simulated station but without contributing to any observation composite or to the bubble generator.

Changes/extentions of the observation times can also be achieved by re-defining the `rs_meta(`*i*`)%obs_times` directly, or by setting `rs_meta(`*i*`)%dt_obs` and `rs_meta(`*i*`)%nobs_times` together with `rs_meta(`*i*`)%lobstimes_ovwrt_recalc=.TRUE.` The latter means that `rs_meta(`*i*`)%obs_times` are re-calculated from `rs_meta(`*i*`)%dt_obs` and `rs_meta(`*i*`)%nobs_times`.

In this way, e.g., it is possible to have realtime forecasts where the most recent observations until the start of the model run can be used for the warm bubble generator, and at the same time synthetic observations can be generated for all stations until the end of the forecast time range, which might be in the future and have no observations yet.

5.1.7 A remark about output of radar composites

As mentioned in Section 2.7, observed and simulated Z_h composites (no polarimetry or radial wind!) 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_glob` defined appropriately in the namelist `/RADARSIM_PARAMS/`. `eleindlist_for_composites_glob` is a global setting for all radar stations, but it can be adjusted for station i by the derived type parameter `rs_meta(`*i*`)%eleindlist_for_composite`. The composites are computed for each time for which at least one of the radars has an observation timestep in the list `rs_meta(`*i*`)%obs_times`.

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 is output in a separate grib file. To distinguish it from the other composites, the given “level” is 0. The underlying composite grid may be differently chosen, but has the same default, and the user might choose a different elevation for compositing.

The grid for both 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/`. For the “normal” composites this is the the derived type `comp_meta`, see Table 12:

- `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`

and for the warm bubble generator, this is `comp_meta_bub`:

- `comp_meta_bub%ni`
- `comp_meta_bub%nj`

- comp_meta_bub%pollon
- comp_meta_bub%pollat
- comp_meta_bub%polgam
- comp_meta_bub%startlon
- comp_meta_bub%startlat
- comp_meta_bub%dlon
- comp_meta_bub%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. / lcomposite_output_bub = .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. / lcomposite_output_bub = .TRUE.`) and the default is the COSMO-D2 grid.

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

To control the reflectivity computation on the model grid for the standard COSMO 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 instance `dbz` of the derived type `t_dbzcalc_params` in each output namelist (COSMO: `/GRIBOUT/`). This derived type is exactly the same as `dbz_meta(`*i*`)` in `/RADARSIM_PARAMS/` (step 1 of the reflectivity operator), whose components have already been described in Table 14. Just replace the prefix `dbz_meta(`*i*`)` by `dbz` when you put it into the COSMO namelist `/GRIBOUT/`.

Each element of this derived type can be specified in the namelist, e.g., `dbz%itype_refl=1`. All available type components are listed in Table 14. 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 from `/RADARSIM_PARAMS/` at another place. 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 5.1.6).

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`.

6 Output of EMVORADO

The output of EMVORADO is written into the directory `ydirradarout` given in namelist `/RADARSIM_PARAMS/`. Before repeating a run for the same date using the same output directory (a situation which is common in development work) **it is advisable to delete the output from the previous run**, because EMVORADO does not automatically overwrite "old" files. This is because some of the outputs described in the next sections append to pre-existing files, for example, when multiple timesteps are written into the same file. In this situation, EMVORADO cannot distinguish between existing files from the actual run and old files from a previous run, so it does not overwrite existing output files.

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 indicate which parameter is in the file. Tab. 15 lists the possible output parameters, whose availability depends on EMVORADOs configuration.

To organize the output of volume scan data to the user' needs, EMVORADO offers the possibility for having differernt output streams. The derived type `t_voldata_ostream` (`radar_data_namelist.f90`) defines the properties of an output stream. The namelist parameter `voldata_ostream(`n`)` in the `/RADARSIM_PARAMS/` is a vector of n instances of this type. Each element triggers one separate output stream. At the moment the maximum allowed number of n is 5.

The different components of `voldata_ostream(`n`)` are described in detail in Tab. 12. Just put as much instances into the namelist as much output streams are required. Here an example for 3 streams:

```
&RADARSIM_PARAMS

  lvoldata_output = .TRUE.

    ind_ele_voldata_glob = 1,2,4,6,8  ! only these elevations (indices)
    dt_obs_voldata_glob = -999.9, -999.9, 300.0  ! output every 300 seconds, synchronized with time

    ! .. in these 3 output streams:
    voldata_ostream(1)%format = 'cdfin-mulmom'
    voldata_ostream(1)%file_pattern = 'cdfin_<stationid>_<varname>_<tstart>-<tend>_<scantype>',
    voldata_ostream(1)%output_subdir = 'multi-moment-cdfins/'
    voldata_ostream(1)%content_dt = 3600.0,
    voldata_ostream(1)%content_tref = 0.0,
    voldata_ostream(1)%output_list = 'all'

    voldata_ostream(2)%format = 'cdfin'
    voldata_ostream(2)%file_pattern = 'cdfin_<stationid>_<varname>_<tstart>_<scantype>',
    voldata_ostream(2)%output_subdir = 'geolocation-cdfins/'
    voldata_ostream(2)%content_dt = 3600.0,
    voldata_ostream(2)%content_tref = 0.0,
    voldata_ostream(2)%output_list = 'losim', 'lasim', 'hrsim'

    voldata_ostream(3)%format = 'grib2'
    voldata_ostream(3)%grib2_packingtype = 'grid_ccsds'
    voldata_ostream(3)%file_pattern = 'grib2_<stationid>_<varname>_<tstart>-<tend>_<scantype>',
    voldata_ostream(3)%output_subdir = 'reflectivity-gribs/'
    voldata_ostream(3)%content_dt = 3600.0,
    voldata_ostream(3)%content_tref = 0.0,
    voldata_ostream(3)%output_list = 'zrsim','zrobs',
/
```

It is further possible by namelist paramters to thin out elevations as well as observation times from volume scan data files, but not as part of the output streams. This can be done on the radar station level with the relevant namelist parameters `rs_meta(i)%ind_ele_voldata`, `rs_meta(i)%obs_times_voldata` and `rs_meta(i)%dt_obs_voldata`, or with their global values (equal for all radar stations) `ind_ele_voldata_glob`, `obs_times_voldata_glob` and `dt_obs_voldata_glob`.

Table 15: List of the available datasets for output of volume scan data.

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] H-pol; -999.99=missing value, -99.99=correct 0	`loutdbz=.TRUE.`
`'zdrsim'`	simulated differential reflectivity [dB]; -999.99=missing value	`loutdbz=.TRUE.` and `loutpolstd=.true.` or `loutpolall=.true.` and `rs_meta(i)%itype_refl=1,5,6`
`'rhvsim'`	simulated cross-correllation coefficient [-]; -999.99=missing value	`loutdbz=.TRUE.` and `loutpolstd=.true.` or `loutpolall=.true.` and `rs_meta(i)%itype_refl=1,5,6`
`'kdpsim'`	simulated specific differential phase shift [°/km]; -999.99=missing value	`loutdbz=.TRUE.` and `loutpolstd=.true.` or `loutpolall=.true.` and `rs_meta(i)%itype_refl=1,5,6`
`'phidpsim'`	simulated total differential phase shift [°]; -999.99=missing value	`loutdbz=.TRUE.` and `loutpolstd=.true.` or `loutpolall=.true.` and `rs_meta(i)%itype_refl=1,5,6`
`'ldrsim'`	simulated linear depolarization ratio [-]; -999.99=missing value	`loutdbz=.TRUE.` and `loutpolall=.true.` and `rs_meta(i)%itype_refl=1,5,6`

`'ahsim'`	simulated twoway attenuation coefficient [db/km] H-pol;
	-999.99=missing value
		`loutdbz=.TRUE.` and
		`lextdbz=.TRUE.`
		and
		`rs_meta(i)%itype_refl=1,5,6`

`'ahpisim'`	simulated path integrated attenuation [dB] H-pol;
	-999.99=missing value
		`loutdbz=.TRUE.` and
		`lextdbz=.TRUE.`
		and
		`rs_meta(i)%itype_refl=1,5,6`

`'adpsim'`	simulated twoway differential attenuation [dB/km]; -999.99=missing value	`loutdbz=.TRUE.` and `loutpolall=.true.` and `rs_meta(i)%itype_refl=1,5,6`
`'adppisim'`	simulated path integrated differential attenuation; -999.99=missing value	`loutdbz=.TRUE.` and `loutpolall=.true.` and `rs_meta(i)%itype_refl=1,5,6`
`'vrsim'`	simulated radial wind [m/s]; -999.99=missing value	`loutradwind=.TRUE.`

Table 15: continued

Keyword	Parameter	Dependencies ("if ...")
`'zrobs'`	observed radar reflectivity in dBZ H-pol; -999.99=missing value	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.`
`'zdrobs'`	observed differential reflectivity [dB]; -999.99=missing value	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.` and `loutpolstd=.true.` or `loutpolall=.true.`
`'rhvobs'`	observed cross-correlation coefficient [-]; -999.99=missing value	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.` and `loutpolstd=.true.` or `loutpolall=.true.`
`'kdpobs'`	observed specific differential phase shift [°/km]; -999.99=missing value	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.` and `loutpolstd=.true.` or `loutpolall=.true.`
`'phidpobs'`	observed total differential phase shift [°]; -999.99=missing value	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.` and `loutpolstd=.true.` or `loutpolall=.true.`
`'ldrobs'`	observed linear depolarization ratio [-]; -999.99=missing value	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.` and `loutpolall=.true.`
`'vrobs'`	observed radial wind. Dealiasing depends on namelist switch `ldealiase_vr_obs`; -999.99=missing value	`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.`
`'zrsupsim'`	super-observations of simulated reflectivity [dBZ] H-pol for development purposes	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.`
`'zrsupobs'`	super-observations of observed reflectivity [dBZ] H-pol for development purposes	`lreadmeta_from_netcdf=.TRUE.` and `loutdbz=.TRUE.`
`'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`
`'losupsim'`	geographic longitude of super-observation reference points [°]	`lreadmeta_from_netcdf=.TRUE.`

Table 15: continued

Keyword	**Parameter**	**Dependencies (“if . . . ”)**
`'lasupsim'`	geographic latitude of super-observation reference points [°]	`lreadmeta_from_netcdf=.TRUE.`
`'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, azimuth 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 is fixed and cannot be modified by `vodata_ostream(`n`)%file_pattern`:

`<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, azimuths 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 default file name convention is very similar to the CDFIN input files. Note that it can be modified by `vodata_ostream(`n`)%file_pattern` (Tab. 12):

`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'`) as "observations" (simulated truth) in an OSSE run (no polarization parameters yet possible). 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'`.
- There are numerous global attributes to describe the model run of the hosting COSMO- or ICON-model: which model suite is it, is it an ensemble, if yes, which member and so on. Their names are inspired by the grib2 keys used in the model's output.
- 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!

'cdfin-mulmom': Similar to `cdfin`, but all radar parameters are in one file.

The default naming convention is similar to `cdfin`, but the variable name (e.g., `zrsim`) is replaced by `allsim` or `allobs` or `allgeom`, respectively.

'grib2': Produces grib2-files where one elevation is one grib2 record. Only for simulated and observed reflectivity "zrsim" or "zrobs" at the moment.

Otherwise very similar to "cdfin" above: 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 "grib2" grib2-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 default file name convention is as follows, but can be modified by `vodata_ostream(`n`)%file_pattern` (Tab. 12):

`grib2_<datasetname>_id-XXXXXX_<starttime>_<endtime>_<scantype>`

- `<datasetname>`: can be only "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

- `grib2_zrsim_id-010132_201707101500_201707101500_volscan`
- `grib2_vrsim_id-010132_201707101500_201707101500_volscan`
- `grib2_zrobs_id-010132_201707101500_201707101500_precip`
- `grib2_lasim_id-010132_201707101200_201707101200_volscan`
- `grib2_losim_id-010132_201707101200_201707101200_volscan`
- `grib2_hrsim_id-010132_201707101200_201707101200_volscan`

This needs compilation with the additional pre-processor flag `-DGRIB_API` and the DWD's `grib_api` or `eccodes` distribution (center=EDZW), with support for ccscs-compression (`aec` library).

The file content is dictated by WMO standards. WMO defines two different templates, one for simulated volume scans and a different one for observed volume scans. Both are similar in that one record is one PPI-Elevation and a volume scan consists of several such records, but they differ in the grib keys which describe the content. For example, observed volume scans do not have any keys that describe the model run which writes the files to disk, whereas this information is given for simulated scans. Many other keys are different as well. The keys can be inspected from an existing grib2-file by using tools like `grib_ls` and `grib_dump`.

To reduce file size, we reduce internal precision to 16 bit, apply an internal bitmap for "0-values" and activate the internal `ccsds`-compression offered by eccodes.

This output should only be chosen if DWD's eccodes distribution with a version $\geq$ 2.20.0 is available! Its main purpose is the possibility to efficiently store the volume scan data in DWD's SKY model output database.

'grib2-mulmom': Similar to `grib2`, but all radar parameters are in one file.

The default naming convention is similar to `grib2`, but the variable name (e.g., `zrsim`) is replaced by `allsim` or `allobs`, respectively.

'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`

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 azimuth-range-boxes around the points of a quasi-kartesian horizontal grid on the PPI-planes.

The files contain among other metadata 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 Z_h 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/`.

For COSMO and if the composite grid is chosen equal to the COSMO model grid (default for `comp_meta`), 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 and for ICON, and it allows the composite grid to be different from the model grid, so it is the recommended option. The default filename convention for EMVORADO composite files is

`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). The filename pattern may be modified by `composite_file_pattern` (Tab. 12).

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”).

The special composite for the warm bubble generator is computed if `ldo_bubbles=.TRUE.` and is output in a separate file if `lcomposite_output_bub = .TRUE.` The default filename convention is

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

and may be modified by `composite_file_pattern_bub` (Tab. 12).

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” azimuth 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

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

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 by default 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

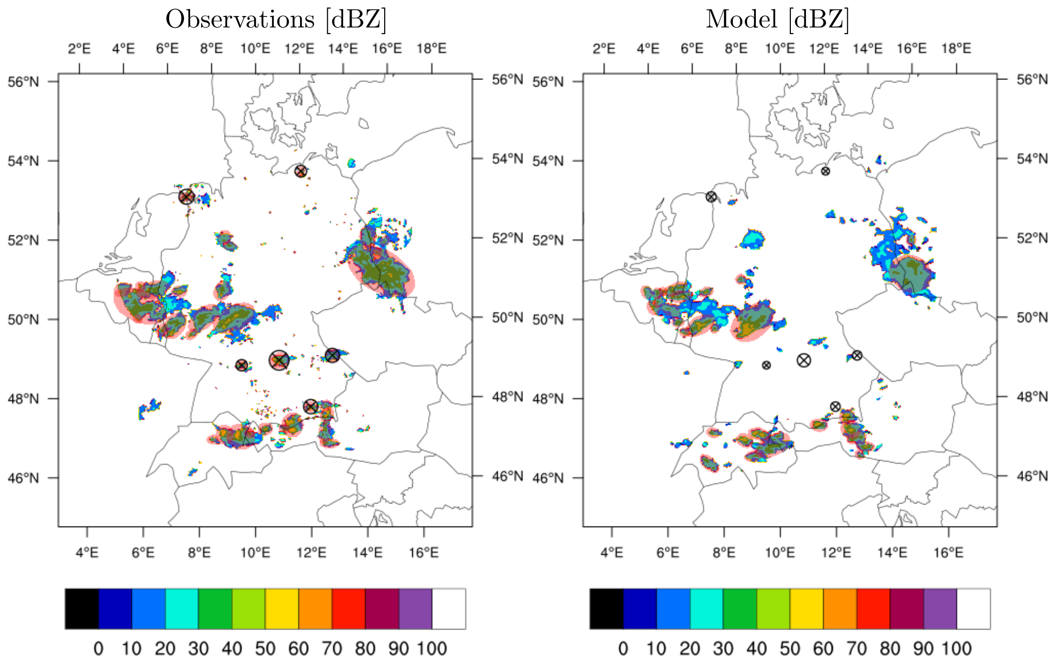

Observations [dBZ]
Model [dBZ]
2°E 4°E 6°E 8°E 10°E 12°E 14°E 16°E 18°E
56°N 54°N 52°N 50°N 48°N 46°N
4°E 6°E 8°E 10°E 12°E 14°E 16°E
0 10 20 30 40 50 60 70 80 90 100

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.

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 12. The bubble parameters have similar meaning as corresponding parameters in the COSMO idealized framework (Blahak, 2015). Table 16 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 16 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., $H_{bub} = 2$ 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 distance that corresponds to 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}. The bubble triggering time itself is not delayed.

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.

In case of asynchonous radar IO, there is a possibility to delay the transfer of the information about detected missing cells from the output PEs to the compute PEs by a certain amount of time (`t_offset_bubble_trigger_async`). This allows the compute PEs to continue model integration while missing cells are detected. Otherwise, compute PEs would have to wait until completion of the bubble search on the IO PEs, which would destroy the runtime advantage of asynchronous IO. However, this comes at the expense of a delayed bubble triggering. The bubble position will be advected downstream similar as and on top of Δt_{advect}.

Table 16: Typical parameters to configure the warm bubble generator in a 2-km-scale model. Cf. Table 12.

	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	0.04	-	bubble_heatingrate (’cos-hrd’)
Δt_{heat}	s	200.0	-	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

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 `/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 (namelist parameter `bubble_type`).

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_organize.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_driver.f90`, `radar_mie_iface_cosmo_1mom.f90`, `radar_mie_iface_cosmo_2mom.f9` 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_output_driver.f90` and `radar_model2rays.f90` contain the generic interface routines for step 2 of the operator, which are directly called from `organize_radar()` in `radar_organize.f90`.
  Down the call tree, these routines call further routines for computing `superobservations` and for output of `volume data`, `feedback files` and `composites` (grib2), which are contained in `radar_output_station.f90`, `radar_output_station_smth.f90`, `radar_output_country_obs.f90`, `radar_output3d_ascii.f90`, `radar_output_composites.f90`, `radar_output_fdbk.f90`, `radar_output_readyfile.f90` and `radar_output_volscan.f90`.
- `radar_obs_meta_read.f90` contains the code for reading the radar station metadata from observation files, if any are used.
- `radar_obs_meta_list.f90` contains the background metadata 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_model2rays.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_organize.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_driver.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_com-
      pute + 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 Z_h, polarization parameters and hydrometeor terminal fallspeed

The corresponding subroutines `calc_dbz_vec_modelgrid()` (Z_h, polarization parameters), `calc_fallspeed_vec_modelgrid()` (terminal fallspeed) and `init_lookup_mie()` (initialization of lookup tables) from `radar_mie_iface_cosmo_driver.f90` are called from step 2-routines from within EMVORADO, when Z_h, polarization parameters 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 Z_h, polarization parameters and v_r

All corresponding subroutines regarding interpolation from model grid to radar bins are contained in `radar_model2rays.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.

If observation files are used, the code for reading the station metadata 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 metadata type `rs_meta(`i`)` depending on `icountry`, as well as background metadata lists for each known radar station for cross-checking.

8.2.4 Implementation of "traditional" grid point reflectivity (Z_h) 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- or T-matrix lookup tables if required by the choice `dbz%itype_refl=1,5,6` 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 subroutines at these points, the grid point reflectivity output has been extended by options for the Mie-, T-matrix and Rayleight-Oguchi-methods `dbz%itype_refl=1,3,5,6` 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=1,5,6`) 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 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 `t_dbzcalc_params`, which is declared in module `radar_data.f90`.

- Add new component to declaration of derived type `t_dbzcalc_params` in `radar_data.f90`.
- Add corresponding background default value to the declaration block of
  `TYPE(t_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 metadata defaults for a group of similar radars. This set serves as a background default for the station metadata `rs_meta` and, as mentioned earlier in Table 12, 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 12.

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>)`

9 Acknowledgements

It all started by two postdocs sitting in the same room at the Institute for Meteorology and Climate Research (IMK), Karlsruhe Institute of Technology (KIT). Axel Seifert (IMK, nowadays DWD) worked on cloud microphysical parameterization schemes and needed a radar forward operator to evaluate his developments by observations of a C-Band research radar. After his initial Mie-coding, Ulrich Blahak (IMK, nowadays DWD) took over and developed a code for grid point reflectivities. The further development into the full EMVORADO with all its physical capabilities (and its tricks to be efficient at the same time) by Yuefei Zeng, Dorit Jerger and Ulrich Blahak at IMK profited from a grant within the programme "Extramural Research" of the German Weather Service (DWD). During this whole time, we are indepted to Klaus D. Beheng (IMK) for his constant support, encouragement and many discussions on cloud microphysics and radar meteorology.

During this process Ulrich Blahak moved from KIT to DWD, and EMVORADO profited from many valuable contributions and discussions with numerous colleages from DWD (Christoph Schraff, Andreas Rhodin, Klaus Stephan, just to name a few). We thank also Daniel Leuenberger from MeteoSwiss for many very valuable discussions and some initial code contributions to solve tricky MPI-problems. EMVORADO was brought into operational service at DWD for assimilation of Z_h and v_r in the regional NWP-models COSMO-D2 and ICON-D2 by contributions of many DWD colleagues from project SINFONY and the data assimilation section. This profited also from research on radar data assimilation within the Hans-Ertel-Zentrum (HErZ) at the Meteorological Institute of Ludwig Maximilians University Munich (LMU).

Jeffrey Snyder and Jacob Carlin (NSSL, Oklahoma, USA) laid the foundations of the upgrade to dual polarimetric capabilities, which was taken over and continued by Jana Mendrok (DWD) with support from grant BL 945/2-1 by Deutsche Forschungsgemeinschaft (DFG) within the Priority Programme "PROM".

References

Andrić, J., M. R. Kumjian, D. S. Zrnić, J. M. Straka and V. M. Melnikov, 2013: Polarimetric signatures above the melting layer in winter storms: An observational and modeling study, *J. Appl. Meteor. Clim.*, **52**(3), 682–700.

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/cosmo/docs/techReport28.pdf`.

Brandes, E. A., G. Zhang and J. Vivekanandan, 2002: Experiments in Rainfall Estimation with a Polarimetric Radar in a Subtropical Environment, *J. Appl. Meteor.*, **41**(6), 674–685.

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`.

Matrosov, S. Y., R. F. Reinking, R. A. Kropfli and B. W. Bartram, 1996: Estimation of ice hydrometeor types and shapes from radar polarization measurements, *J. Atmos. Ocean. Tech.*, **13**(1), 85–96.

Prill, F., D. Reintert, D. Rieger and G. Zängl, 2020: *Working with the ICON model*, Deutscher Wetterdienst, Offenbach, Germany, URL `https://www.dwd.de/EN/ourservices/nwv_icon_tutorial/nwv_icon_tutorial_en.html`.

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

Ryzhkov, A., 2001: Interpretation of polarimetric radar covariance matrix for meteorological scatterers: Theoretical analysis, *J. Atmos. Ocean. Tech.*, **18**, 315–328.

Ryzhkov, A., M. Pinsky, A. Pokrovsky and A. Khain, 2011: Polarimetric Radar Observation Operator for a Cloud Model with Spectral Microphysics, *J. Appl. Meteor. Clim.*, **50**(4), 873–894.

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`.

Trömel, S., C. Simmer, U. Blahak, A. Blanke, S. Doktorowski, F. Ewald, M. Frech, M. Gergely, M. Hagen, T. Janjic, H. Kalesse-Los, S. Kneifel, C. Knote, J. Mendrok, M. Moser, G. Köcher, K. Mühlbauer, A. Myagkov, V. Pejcic, P. Seifert, P. Shrestha, A. Teisseire, L. von Terzi, E. Tetoni, T. Vogl, C. Voigt, Y. Zeng, T. Zinner and J. Quaas, 2021: Overview: Fusion of radar polarimetry and numerical atmospheric modelling towards an improved understanding of cloud and precipitation processes, *Atmos. Chem. Phys.*, **21**(23), 17291–17314.

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.

Xie, X., P. Shrestha, J. Mendrok, J. Carlin, S. Trömel and U. Blahak, 2021: Bonn Polarimetric Radar forward Operator (B-PRO), URL `https://git2.meteo.uni-bonn.de/projects/pfo/`.

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.