

July
2025
DRAFT

ICON Tutorial



Working with the ICON Model

F. Prill, D. Reinert, D. Rieger, G. Zängl

Acknowledgments

Many people contributed to this manuscript.

The section on ICON physics was partly provided by S. Schäfer (radiation, Section 3.8.1), J. Helmert (land-soil model TERRA, Section 3.8.9), D. Klocke (convection parameterization, Section 3.8.6), M. Köhler (cloud-cover parameterization and turbulence, Sections 3.8.7, 3.8.8), D. Mironov (summary of sea-ice and lake model, Sections 3.8.10, 3.8.11), M. Raschendorfer (turbulence, Section 3.8.8), and A. Seifert (grid-scale microphysics parameterization, Section 3.8.5), DWD Physical Processes Division.

Section 6.2 (ICON-LAM nudging) includes contributions by S. Borchert, DWD. Section 6.5 was created mostly based on experiences by U. Blahak, DWD, for tuning the COSMO model for tropical setups.

M. Jacob has provided Section 8.5 which contains an introduction to the accelerator-specific code in ICON.

S. Rast (MPI-M) provided useful specifics on the grid construction, internal representation of fields and other details, see Rast (2017), in particular in Ch. 9. Section 9.5 on the Community Interface was contributed by N.-A. Dreier (DKRZ) and M. Haghghatnasab (DWD).

The Section 10.4.2 (Post-Processing using Fieldextra) has been provided by P. Baumann and J.-M. Bettems, MeteoSwiss. Section 10.3.3 (Visualization with R) has been contributed by J. Förstner, DWD Physical Processes Division.

Chapter 11 was in its original form provided by R. Potthast and A. Fernandez del Rio, DWD Data Assimilation Division.

Chapter 12 on the CLM Community was provided by S. Brienens and C. Steger, DWD Climate Projection Division.

DRAFT VERSION



The CC license “BY-NC-ND” allows others only to download the publication and share it with others as long as they credit the publication, but they can’t change it in any way or use it commercially.

Publisher

Deutscher Wetterdienst
Business Area “Research and Development”
Frankfurter Straße 135
63067 Offenbach
www.dwd.de

Editors

Daniel Reinert, DWD,
daniel.reinert@dwd.de
Daniel Rieger, DWD,
daniel.rieger@dwd.de
Florian Prill, DWD,
florian.prill@dwd.de

Contents

0. Preface	1
0.1. How to Obtain a Copy of the ICON Model Code	2
0.2. Further Documentation	3
0.3. How This Document Is Organized	4
1. Installation of the ICON Model Package	5
1.1. ICON for Numerical Weather Prediction (NWP)	5
1.1.1. Directory Layout	5
1.1.2. Namelist Input for the ICON Model	7
1.2. Libraries Needed for Data Input and Output	7
1.2.1. The Climate Data Interfaces (CDI) – <code>externals/cdi</code>	8
1.2.2. The NetCDF library – <code>libnetcdf.a</code>	8
1.2.3. The ECMWF ecCodes package	8
1.2.4. GRIB Definition Files	9
1.3. Configuring and Compiling the Model Code	10
1.3.1. Configuring and Compiling	10
1.3.2. Example: Configuration and build process for the NEC SX-Aurora	12
1.4. The DWD ICON Tools	13
2. Necessary Input Data	17
2.1. Horizontal Grids	17
2.1.1. ICON Grid Files	19
2.1.2. ICON “Nests”	21
2.1.3. Mapping of Geodesic Coordinates to the Sphere	22
2.1.4. Download of Predefined Grids	23
2.1.5. Grid Generator: Invocation from the Command Line	24
2.1.6. Grid Generator: Invocation via the Zonda Web Interface	26
2.1.7. Which Grid File is Related to My Simulation Data?	27
2.1.8. Planar Torus Grids	28
2.2. Initial Conditions	28
2.2.1. Obtaining DWD Initial Data	29
2.2.2. Obtaining ECMWF IFS Initial Data	35
2.2.3. Remapping Initial Data to Your Target Grid	37
2.3. Boundary Data Preparation for ICON-LAM	44
2.4. External Parameter Files	51
2.4.1. ExtPar Software	51
2.4.2. Topography Information	52
2.4.3. Additional Information for Surface Tiles	52
2.4.4. Parameter Files for Radiation	53

3. Model Description	57
3.1. Governing Equations	58
3.1.1. Computation of the Virtual Potential Temperature	62
3.2. Simplifying Assumptions in the Recent Model Version	62
3.3. The Model Reference State	64
3.4. Vertical Coordinates	65
3.4.1. Terrain-following Hybrid Gal-Chen Coordinate	67
3.4.2. SLEVE Coordinate	68
3.5. Temporal Discretization	70
3.5.1. Basic Idea	70
3.5.2. Implementation Details	71
3.6. Tracer Transport	77
3.6.1. Directional Splitting	78
3.6.2. Horizontal Transport	79
3.6.3. Vertical Transport	81
3.6.4. Reduced Calling Frequency	84
3.6.5. Some Practical Advice	86
3.7. Physics-Dynamics Coupling	88
3.7.1. ICON Time-Stepping	90
3.7.2. Fast and Slow Processes	92
3.7.3. Isobaric vs. Isochoric Coupling Strategies	94
3.8. ICON NWP-Physics in a Nutshell	96
3.8.1. Radiation	96
3.8.2. Non-orographic gravity wave drag	99
3.8.3. Sub-grid scale orographic drag	99
3.8.4. Saturation Adjustment	102
3.8.5. Cloud Microphysics	105
3.8.6. Cumulus Convection	106
3.8.7. Cloud Cover	107
3.8.8. Turbulent Diffusion	108
3.8.9. Land-Soil Model TERRA	112
3.8.10. Lake Parameterization Scheme FLake	115
3.8.11. Sea-Ice Parameterization Scheme	117
3.8.12. Reduced Model Top for Moist Physics	118
3.9. Variable Resolution Modeling	120
3.9.1. Parent-Child Coupling	122
3.9.2. Processing Sequence	130
3.9.3. Technical and Performance Aspects	132
3.10. Reduced Radiation Grid	133
4. Running Idealized Test Cases	135
4.1. Main Switches for Idealized Test Cases	135
4.1.1. Activating/De-activating Main Model Components	135
4.1.2. Specifying the Computational Domain(s)	136
4.1.3. Integration Time Step and Simulation Length	137
4.2. Jablonowski-Williamson Baroclinic Wave Test	138
4.2.1. Recommended Namelist Settings	138
4.2.2. Enabling Passive Tracers	139

4.2.3. Activation of Nested Domains	141
4.3. Straka Density Current Test	142
4.3.1. Relevant Namelist Switches in <code>nh_testcase_nml</code> :	145
5. Running Real Data Cases	147
5.1. Model Initialization	147
5.1.1. Basic Settings for Running Real Data Runs	147
5.1.2. Starting from Uninitialized DWD Analysis	151
5.1.3. Starting from Uninitialized DWD Analysis with IAU	153
5.1.4. Starting from Initialized DWD Analysis	154
5.1.5. Starting from IFS Analysis	154
5.2. Starting or Terminating Nested Domains at Runtime	155
6. Running ICON-LAM	157
6.1. Limited Area Mode vs. Nested Setups	157
6.2. Nudging in the Boundary Region	158
6.3. Model Initialization	162
6.4. Reading Lateral Boundary Data	163
6.4.1. Naming Scheme for Lateral Boundary Data	165
6.4.2. Pre-Fetching of Boundary Data (Mandatory)	166
6.5. Tropical Setup	166
7. Model Output	169
7.1. Settings for the Model Output	169
7.1.1. Output on Regular Grids and Vertical Interpolation	171
7.1.2. Remarks on the Horizontal Interpolation	172
7.1.3. Interpolation onto Rotated Lat-Lon Grids	174
7.1.4. Output Rank Assignment	175
7.2. Checkpointing and Restart	176
7.3. Meteogram Output	178
8. Parallelization and Performance Aspects	181
8.1. Modes of Parallel Execution	181
8.2. Settings for Parallel Execution	182
8.3. Best Practice for Parallel Setups	183
8.3.1. MPI Tasks and OpenMP Threads	183
8.3.2. Blocking (<code>nproma</code>)	184
8.3.3. Mixed Single/ Double Precision in ICON	184
8.3.4. Bit-Reproducibility	184
8.4. Basic Performance Measurement	185
8.5. ICON on Accelerator Devices (GPUs)	186
8.5.1. Configuring and Compiling ICON-OpenACC	187
8.5.2. Special Namelist Options for ICON-OpenACC	188
8.5.3. Implementation Details	189
9. Programming ICON	191
9.1. Representation of 2D and 3D Fields	191

9.2.	Data Structures	196
9.2.1.	Description of the Model Domain	196
9.2.2.	Date and Time Variables	198
9.2.3.	Data Structures for Physics and Dynamics Variables	199
9.2.4.	Parallel Communication	201
9.3.	NWP Call Tree	202
9.4.	Implementing Own Diagnostics	202
9.5.	The ICON Community Interface ComIn	207
9.5.1.	Introduction	207
9.5.2.	Example Plugins	208
9.5.3.	Building Blocks of a Fortran ComIn Plugin	210
9.5.4.	Step-by-step Tutorial	212
10.	Post-Processing and Visualization	215
10.1.	Retrieving Data Set Information	215
10.1.1.	The ncdump Tool	215
10.1.2.	CDO – Climate Data Operators	216
10.2.	Plotting Data Sets on Regular Grids: ncview	217
10.3.	Plotting Data Sets on the Triangular Grid	218
10.3.1.	Visualization with Python	218
10.3.2.	NCL – NCAR Command Language	219
10.3.3.	Visualization with R	222
10.4.	Post-Processing of Data Sets	226
10.4.1.	Post-Processing using the CDO	226
10.4.2.	Post-Processing using Fieldextra	228
11.	ICON's Data Assimilation System and Analysis Products	231
11.1.	Data Assimilation	231
11.1.1.	Variational Data Assimilation	232
11.1.2.	Ensemble Kalman Filter	233
11.1.3.	Hybrid Data Assimilation	233
11.1.4.	Surface Analysis	234
11.2.	Assimilation Cycle at DWD	234
11.3.	Incremental Analysis Update	236
11.3.1.	Variable Transformation	237
11.3.2.	Technical Hints	239
11.4.	Analysis Products	241
11.4.1.	Uninitialized Analysis for IAU	241
11.4.2.	Uninitialized Analysis	243
11.4.3.	Initialized Analysis	245
12.	ICON-CLM	249
12.1.	The CLM Community	249
12.2.	Activities of the CLM Community	249
12.2.1.	CORDEX	249
12.2.2.	COPAT	250
12.3.	The ICON as Regional Climate Model - ICON-CLM	250
12.3.1.	Lateral Boundary Conditions	251

12.3.2. Running with Restart Files	251
12.3.3. Calendar	251
12.3.4. Land Surface Scheme	252
12.3.5. Ocean and Sea Ice	252
12.3.6. Time-dependent External Forcings	252
12.3.7. Model Output	253
12.4. Runtime Environment SPICE	254
Appendix A. Table of NWP Output Variables	255
Bibliography	271
Index of Namelist Parameters	283

0. Preface

The ICON (ICOsahedral Nonhydrostatic) modeling framework is a unified global numerical weather prediction and climate modeling system.

Initially, ICON was developed as a joint project by the German Weather Service (DWD) and the Max-Planck-Institute for Meteorology (MPI-M). Currently, the institutions behind ICON are the DWD, the MPI-M, the German Climate Computing Center (DKRZ), the Karlsruhe Institute of Technology (KIT), and the Swiss Center for Climate Systems Modeling (Federal Office of Meteorology and Climatology MeteoSwiss and ETH Zurich as C2SM partners).

The main goals formulated in the initial phase of the collaboration were ([Zängl et al., 2015](#))

- applicability on a wide range of scales down to $\mathcal{O}(1\text{ km})$ and beyond, which requires a nonhydrostatic dynamical core,
- better conservation properties than in the existing global models, with the obligatory requirement of exact local mass conservation and mass-consistent transport,
- better scalability on future massively parallel high-performance computing architectures,
- the availability of some means of static mesh refinement. ICON is capable of mixing one-way nested and two-way nested grids within one model application, combined with an option for vertical nesting. This allows the global grid to extend into the mesosphere (which facilitates the assimilation of satellite data) whereas the nested domains extend only into the lower stratosphere in order to save computing time.

The ICON modeling framework is now used in Germany and Switzerland for operational weather forecasting. ICON became operational in DWD's forecast system in January 2015. During the first six months only global simulations were executed with a horizontal grid spacing of 13 km and 90 vertical levels. Starting from July 21st, 2015, model simulations have been complemented by a nesting region over Europe.

In January 2018, the global 40 member ICON-EPS (Ensemble Prediction System) was released for the operational service at DWD. Since February 10th, 2021 it is complemented by the convection-permitting model setup ICON-D2 (-EPS) with 20 ensemble members and one deterministic run, which replaces the COSMO-D2 (-EPS) model ([Baldauf et al., 2011](#)). It uses a limited area domain covering Germany and some neighboring states, with approximately 2 km horizontal mesh size and 60 vertical levels.

0.1. How to Obtain a Copy of the ICON Model Code

Since January 31, 2024, the source code of the model has been released to the public under the **BSD-3C Open Source License** ([ICON Press Release, 2024](#)).

Code releases can be downloaded from the following URL:

<https://gitlab.dkrz.de/icon/icon-model>.

However, it should be noted that these public downloads do not extend to the internal development branches of the program code. Access to the source code management system *git* is limited to the development partners of the ICON project.

In addition to this option for obtaining the program code, the German Weather Service also offers a **special license for national weather services**, a “Software License Contract on Delivery, Support, and Meteorological and Climatological Services for Official Duty and Scientific Purposes”. This license is necessary to run a daily operational weather forecast using ICON-LAM.

Therefore, in terms of user support and web sites, there is a distinction between general contact addresses and those for national weather services. For the general user community, information is available at <https://www.icon-model.org>. The mailing list community@lists.icon-model.org serves to inform ICON users about current developments, workshops and other interesting things in connection with the open source versions of ICON. Please visit <https://www.icon-model.org/news/community-newsletter> to subscribe to this list.

The interests of the weather services are served by the Consortium for Small Scale Modeling (COSMO). For more information, visit the COSMO website <https://www.cosmo-model.org> and you can also contact

icon-support@cosmo-model.org

Data Services

DWD has made a number of **model forecast data sets** publicly available, mostly free of charge. This service has started in July 2017 and can be reached under <https://opendata.dwd.de/weather/nwp>. See the content description under <https://www.dwd.de/EN/ourservices/opendata/opendata.html> for a list of available data sets.

For further **data requests** with respect to DWD operational data products please contact klima.vertrieb@dwd.de.

For developers, there exists a **grid generator web service**, see Section 2.1.6, and – for prefabricated grid files – the download page <http://icon-downloads.mpimet.mpg.de> might also be useful. Finally, please note the download site for DWD’s GRIB2 definitions, <https://opendata.dwd.de/weather/lib/grib/>.

0.2. Further Documentation

The ICON model is accompanied by various other manuals and documentation. A limited list of journal publications as a starting point is available on the public ICON web site <https://www.icon-model.org/publications/reference-publications>.

We restrict ourselves to a small subset in the following.

Scientific Documentation

Up to now there is no comprehensive scientific documentation available. In this respect, we refer to the publications [Zängl et al. \(2015, 2022\)](#) and the references cited therein.

Recent information on ICON's hydrostatic dynamical core¹ and the LES model can be found in [Wan et al. \(2013\)](#), [Dipankar et al. \(2015\)](#), [Heinze et al. \(2017\)](#).

Detailed information and evaluation of the atmospheric component of ICON using the climate physics package is given by [Giorgetta et al. \(2018\)](#), [Crueger et al. \(2018\)](#). The ICON-Sapphire configuration which targets a representation of the Earth system at kilometer and subkilometer scales is described in [Hohenegger et al. \(2023\)](#).

The *Reports on ICON* are a series of non-peer-reviewed articles dedicated to ICON:

https://www.dwd.de/EN/ourservices/reports_on_icon/reports_on_icon.html

These are not attributed to DWD or MPI-M alone and allow for a fast and straightforward publication of technical and scientific contributions. All ICON developers are invited to contribute.

The extended modules for Aerosols and Reactive Trace gases (ART) are described in [Rieger et al. \(2015\)](#), [Schröter et al. \(2018\)](#). Not covered by this tutorial, a description of the ocean component ICON-O within the ICON modeling system can be found in [Korn \(2017\)](#), [Korn and Danilov \(2017\)](#), [Korn et al. \(2022\)](#).

Technical Documentation

For model users who intend to process data products of DWD's operational runs, the DWD database documentation may be a valuable resource, see [Reinert et al. \(2024\)](#). It can be found (in English language) on the DWD web site

www.dwd.de/SharedDocs/downloads/DE/modelldokumentationen/nwv/icon/icon_dbbeschr_aktuell.pdf.

A complete list of namelist switches can be found in the namelist documentation

`icon/doc/Namelist_overview/Namelist_overview.pdf`

which is deployed together with the code.

The pre- and post-processing tools of the DWD ICON Tools collection are described in more detail in the DWD ICON Tools manual, see [Prill \(2020\)](#).

¹ICON's hydrostatic dynamical core has been removed with version 2.6.4.


0.3. How This Document Is Organized

Not all topics in this manuscript are covered during the DWD ICON training workshop. Therefore, the manuscript can be used as a textbook, similar to a user manual for the ICON model. Readers are assumed to have a basic knowledge of the design and usage of numerical weather prediction models.

Even though the chapters in this textbook are largely independent, they should preferably not be treated in an arbitrary order.

- For getting started with the ICON model: read Chapters [1](#) – [5](#).
- New users who are interested in the *regional* model should read Chapter [6](#) in addition.
- More advanced topics are covered by Chapters [7](#) – [11](#).

To some extent this document can also be used as a reference manual. We refer to the index on page [283](#) for a quick look-up of namelist parameters.

Paragraphs describing common pitfalls and containing details for advanced users are marked by the symbol .

1. Installation of the ICON Model Package

The purpose of this tutorial is to give you some practical experience in installing and running the ICON model package. The version information provided in this chapter for the compilers and libraries, as well as the various installation notes, refer to the HPC systems of DWD and DKRZ, but they can directly be transferred to other systems.

1.1. ICON for Numerical Weather Prediction (NWP)

Since January 31, 2024, the climate and weather model ICON has been made available to the public under an open source license, see the [ICON Press Release \(2024\)](#). This release brings together the individual development branches of the participating institutes, especially the modules for numerical weather prediction:

- **The ICOSahedral Nonhydrostatic model (ICON)**
For this tutorial the release 2024.01-1 of the ICON code is used (state *April 2024*). It is close to DWD's currently operational version.
- **ICON-ART for aerosols and reactive trace gases**
The ART module, where ART stands for Aerosols and Reactive Trace gases, is an extension of the ICON model. With ICON-ART, the Karlsruhe Institute of Technology (KIT) has developed a model component that allows the projection of aerosols and atmospheric chemistry and their interaction with the physical state of the atmosphere. Aerosols and the chemical composition determine air quality and also influence solar radiation, clouds and precipitation.

Of course, the open source version of ICON does not only contain the dynamics and physics of the atmospheric model. It also includes the ocean model developed at MPI-M, which allows to run ICON as a fully coupled climate and Earth system model. In addition to the model component for ocean circulation, there is a model component for biogeochemistry as well as for the land biosphere and hydrological hydrological processes. These model components, however, are not covered by this tutorial.

1.1.1. Directory Layout

The ICON source code can be downloaded from the web site <https://gitlab.dkrz.de/icon/icon-model> with the following command:

```
export ICONURL=https://gitlab.dkrz.de/icon/icon-model
export ICONRELEASE=icon-2024.01-1
wget -qO- $ICONURL/-/archive/$ICONRELEASE-public/icon-model-$ICONRELEASE-public.tar.gz |
tar --transform "s/icon-model-$ICONRELEASE-public/icon/" -xvz
```

This compact command downloads the source code as a tar file from the release homepage, and extracts it into a subdirectory `icon`.

Figure 1.1 shows the directory structure of the ICON model.

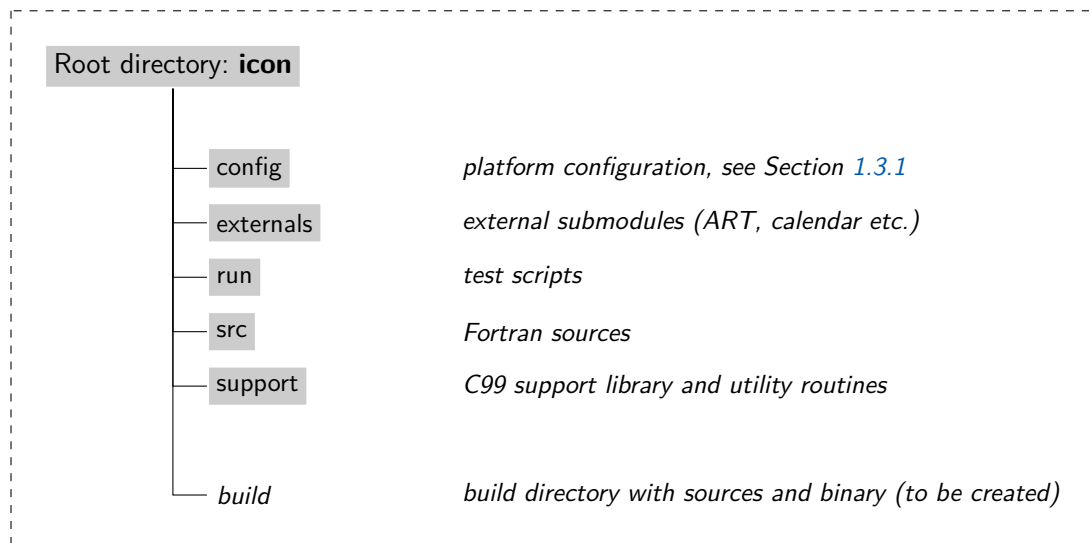


Figure 1.1.: Directory structure of the ICON model.

The most important subdirectories are described in the following:

Subdirectory `build`

We recommend to create this directory and perform an *out-of-source* build. This way, you can build ICON in several different configurations, i. e. with different compilers and features, using the same copy of the source code.

In this folder, after finishing the compilation process, you find a `bin` subdirectory containing the ICON binary `icon` and several other subdirectories containing the compiled module files.

Subdirectory `config`

Inside the `config` directory, different machine-dependent configurations are stored in configuration script files (see Section 1.3.1).

Subdirectory `src`

Within the `src` directory we have the source code of ICON including the main program and ICON modules. The modules are organized in several subdirectories:

The main program `icon.f90` can be found inside the subdirectory `src/drivers`. Additionally, this directory contains the modules for the nonhydrostatic setup.

The configuration of ICON run-time settings is implemented within the modules inside `src/configure_model` and `src/namelists`. Modules regarding the configuration of idealized test cases can be found inside `src/testcases`.

The dynamics of ICON are implemented inside `src/atm_dyn_iconam` and the physical parameterizations inside `src/atm_phy_nwp`. Surface parameterizations can be found inside `src/lnd_phy_nwp`.

1.2 Libraries Needed for Data Input and Output

Shared infrastructure modules for 3D and 4D variables are located within `src/shared`. Routines that are primarily related to horizontal grids and 2D fields (e.g. external parameters) are stored within `src/shr_horizontal`.

Modules handling the parallelization can be found in `src/parallel_infrastructure`.

Input and output modules are stored in `src/io`.

The ICON code comes with its own LAPACK and BLAS sources. For performance reasons, these libraries should be replaced by machine-dependent optimizations.

1.1.2. Namelist Input for the ICON Model

In general, the ICON model is controlled by a so-called parameter file which uses Fortran NAMELIST syntax. Default values are set for all parameters, so that you only have to specify values that differ from the default.

Assuming that ICON has been compiled successfully, the next step is to adapt these ICON namelists. First, the ICON run scripts create a file `icon_master.namelist`. This file in turn does not contain the specifications of all model parameters, but refers to a second file – usually `NAMELIST_NWP` for the ICON-NWP – that provides the settings for grids, the dynamic core and the physical parameterizations.

Discussing all available namelist switches is definitely beyond the scope of this tutorial. We will merely focus on the particular subset of namelist switches that is necessary to setup an idealized model run as well as real case runs using the NWP physics package. A complete list of namelist switches can be found in the namelist documentation

`icon/doc/Namelist_overview/Namelist_overview.pdf`

1.2. Libraries Needed for Data Input and Output

The ICON model package lets you integrate a whole variety of external libraries. See the corresponding table in the document `doc/Quick_Start.md` in the root directory of the ICON source code for a detailed list of required and optional libraries for ICON.

Three libraries play a particularly important role in the execution of I/O tasks, since they implement the reading and writing of the following data formats from or to disk:

- GRIB (*GRIdded Binary*) is a standard defined by the World Meteorological Organization (WMO) for the exchange of processed data in the form of grid point values expressed in binary form. GRIB coded data consists of a continuous bit-stream made of a sequence of octets (1 octet = 8 bits). Please note that the ICON model does support only the GRIB2 version of the standard.
- NetCDF (Network Common Data Form) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.

NetCDF files contain the complete information about the dependent variables, the history, and the fields themselves. The NetCDF file format is also used for the definition of the computational mesh (grid topology).

For more information on NetCDF see <http://www.unidata.ucar.edu>.

1.2.1. The Climate Data Interfaces (CDI) – `externals/cdi`

This library has been developed and implemented by the Max-Planck-Institute for Meteorology in Hamburg. It provides a C and Fortran interface to access climate and NWP model data. Among others, supported data formats are GRIB1/2 and NetCDF.

For more information see <https://code.mpimet.mpg.de/projects/cdi>.

A copy of the CDI is distributed together with the ICON model package. However, users can download and install this library before configuring ICON and use it instead. Note that the CDI are also used by the DWD ICON Tools, see Section 1.4 below.

1.2.2. The NetCDF library – `libnetcdf.a`

A special library, the NetCDF library, is necessary to write and read data using the NetCDF format. This library also contains tools for manipulating and visualizing the data (`ncdump` utility, see Section 10.1.1).

If the library is not yet installed on your system, you can get the source code and documentation from

<http://www.unidata.ucar.edu/software/netcdf/index.html>

This includes a description how to install the library on different platforms. Please make sure that the F90 package is also installed, since the model reads and writes grid data through the F90 NetCDF functions.

Note that there exists a restriction regarding the file size. While the classic NetCDF format could not deal with files larger than 2 GiB the new NetCDF-4/HDF5 format permits storing files as large as the underlying file system supports. However, NetCDF-4/HDF5 files are unreadable to the NetCDF library before version 4.0.

1.2.3. The ECMWF `ecCodes` package¹ – `libeccodes.a`, `libeccodes_f90.a`

The European Centre for Medium-Range Weather Forecasts (ECMWF) has developed an application programmers interface (API) to pack and unpack GRIB1 as well as GRIB2 formatted data. For reading and setting meta-data, the *ecCodes* package uses the so-called key/value approach, which means that all the information contained in the GRIB message

¹*ecCodes* is an evolution of the former GRIB-API software package. To facilitate the alternative use of both libraries, ICON implements only backward-compatible API function names. The GRIB-API libraries, however, would be `libgrib_api.a`, `libgrib_api_f90.a`.

is retrieved through alphanumeric names. Indirect use of this ecCodes library in the ICON model is implemented through the CDI.

In addition to the GRIB library, there are some command-line tools to provide an easy way to check and manipulate GRIB data from the shell. Amongst them, the most important ones are `grib_ls` and `grib_dump` for listing the contents of a GRIB file, and `grib_set` for (re)-setting specific key/value pairs.

For more information on ecCodes we refer to the ECMWF web page:

<https://confluence.ecmwf.int/display/ECC>



Installation: The source code for the ecCodes package can be downloaded from the ECMWF web page.

Please refer to the **README** for installing the ecCodes libraries, which is done with a **configure** script. Check the following settings:

- The ecCodes package can make use of optional JPEG packing of the GRIB records, but this requires the installation of additional libraries. Since the ICON model does not apply this packing algorithm, the support for JPEG can be disabled during the configure step with the option `--disable-jpeg`.
- To use statically linked libraries and binaries you should set the configure option `--enable-shared=no`.

```
./configure --prefix=/your/install/dir \
            --disable-jpeg --enable-shared=no
```

After the configuration has finished, the ecCodes library can be built with **make** and then **make install**.

1.2.4. GRIB Definition Files

The installation files of the ecCodes package always consists of two parts: First, there is the binary compiled library itself with its functions for accessing GRIB files. But, second, there is the *definitions directory* which contains plain-text descriptions of meta data.

GRIB definition files are external text files which constitute a kind of parameter database. They describe the decoding rules and the keys which are used to identify the meteorological fields. For example, these definition files contain information about the variable short name and the corresponding GRIB code triplet.

Example. In contrast to the GRIB triplet, the short name, e.g., “OMEGA” for vertical velocity (pressure), is not stored in data files. The definition file therefore constitutes an essential link: If the definition files in two institutes are different from each other it is possible that the same data file shows the record “OMEGA” on one site (our DWD system), while the same GRIB record bears the short name “w” on the other site (both have the same GRIB triplet `discipline=0,parameterCategory=2,parameterNumber=8`).

DWD-specific definition files. The ICON model accesses its input data by their name ("shortName" key). Therefore the DWD-specific definition files ("EDZW"=DWD Offenbach) are essential for the read-in process. In theory, the above situation could be solved by changing all field names in the ICON name list setup, where possible. However, it is likely that further related errors may follow in the ICON model when this searches for a specific variable name. In this case you might need to change the definition files after all.

The DWD definition files for the ecCodes package can be obtained via

<https://opendata.dwd.de/weather/lib/grib/>

The new directory needs to be communicated to the ecCodes package at run-time by setting the `ECCODES_DEFINITION_PATH` environment variable²:

```
export \
  ECCODES_DEFINITION_PATH=/yourpath/definitions.edzw:/yourpath/definitions
```

Here, the `definitions` directory provided by ECMWF is extended by DWD's own installation (`definitions.edzw`). Note that both paths have to be specified in this environment variable, and that `definitions.edzw` has to be the first! The current setting of the definition files path can be displayed with the command-line tool `codes_info`.

Note that for *writing* GRIB2 files, the ICON model does not use DWD-specific shortNames. Therefore, the model output can be written in GRIB2 format without the proper definition files at hand.



Remark on versioning: The ecCodes library does not check the version of the definition files. In case of a mismatch between these versions, ecCodes will throw an error which can not be easily attributed to a version mismatch. Thus, the user has to make sure that the ecCodes version and the definition files version matches.

1.3. Configuring and Compiling the Model Code

This section explains the configuration process of the ICON model. It is assumed that the libraries and programs discussed in Section 1.2 are present on your computer system. For convenience, the compiler version and the ecCodes version are documented in the log output of each model run.

1.3.1. Configuring and Compiling

The process of building ICON consists of two parts: *configuring* the options and compiler flags, and *building* the source code with those options and flags. Since these two processes

²Setting the `GRIB_DEFINITION_PATH` environment variable is still accepted as a fallback by the ecCodes package for backward compatibility reasons.

Fortran Compiler	Working Version(s)
GNU	gcc v11.2.0 gcc v12.1.0 gcc v12.3.0
Cray	ftn v12.0.3 ftn v16.0.1
Intel	ifort v2021.5.0 ifort v2021.6.0
NAG	nagfor v7.1.7114 nagfor v7.1.7125
NEC	nfort v5.0.1 nfort v5.1.0
NVIDIA (CPU)	nvfortran v21.3 nvfortran v23.3
NVIDIA (GPU)	nvfortran v21.3 nvfortran v23.3

Table 1.1.: Fortran compiler versions which are tested regularly and which are known to successfully build the ICON code (state *April 2024*).

are described in great detail in the `doc/Quick_Start.md` file distributed together with the ICON model, we will limit ourselves here to a brief summary and a few examples.

Due to the usage of modern Fortran 2003/2008 features, ICON places high demands on the compilers. Please make also sure that a compatible compiler for the C99 routines in the package is available. Both components, the Fortran parts and the C parts use a source pre-processor. Table 1.1 provides a list of compilers which are regularly tested and known to successfully build the recent ICON code. There is a good chance that more recent compiler versions might work as well. However, be aware that this is not necessarily the case.

The configuration step is normally done by running the `configure` script (which is part of the GNU Autotools) with command-line arguments, which, among other things, tell the script where to locate libraries and tools required for building. Again we refer to the document `README.md` in the root directory of the ICON source code which contains a very detailed description of the configuration process.

The `configure` command scans the build environment and generates an appropriate `Makefile`. The list of arguments enabling a successful configuration might be quite long and difficult to compose, therefore, instead of running the generic `configure` script directly, users are recommended to execute a corresponding platform- or machine-specific configuration wrapper that sets the required compiler and linker flags as well as the recommended set of configure options. It is also possible to add your own additional options to the call of the wrapper script. These options are then passed on to the configure command.

For the open source release, it should be noted in particular that various model components that are not publicly distributed must be deactivated³:

```
../config/wrapper/script --disable-dace --disable-emvorado
```

The wrapper scripts can be found in the respective subdirectories of the directory `icon/config`.

If your platform is not among the list of presets, or if you need to add a specific compiler or change your compiler flags, you have to set up the appropriate call of the `configure` yourself. Numerous platform-dependent options are allowed. Some more details on configure options can be found in the help of the configure command:

```
./configure --help
```

Be warned that you need some knowledge about Unix / Linux, compilers and Makefiles to make the necessary adjustments w.r.t. the computing environment. If the configuration process fails, take a look at the text file `config.log` that is created during the configuration process. This technical log file may contain hints on which particular library has been found missing.

The building stage is done with GNU `make` upon successful completion of the configuration stage. On most machines you can also compile the routines in parallel by using the GNU-`make` with the command `gmake -j np`, where `np` gives the number of processors to use (`np` typically about 8).

1.3.2. Example: Configuration and build process for the NEC SX-Aurora

The NEC platform represents a special case in the sense that two separate ICON binaries are created, which will be simultaneously executed on the x86 hosts nodes and the vector engines. For this reason, we will explain the build process here as an example.

It is advisable to create a `build` subdirectory which will contain the binaries for your computer architecture. The building system of ICON supports so-called *out-of-source builds*. This means that you can build ICON in a directory other than the source root directory. The main advantage of this is that you can easily switch between several different configurations and compilers later (each in its own build directory), while working on the same source code. For the case of the NEC SX-Aurora, we will have different builds for the x86 hosts nodes and the vector engines.

In order to start the compilation process, please log into the NEC SX-Aurora cross-compilation node `rc1` and change into the subdirectory `icon`. Please type:

```
mkdir -p build/{VE,VH}
```

```
cd build/VE
```

³Note that the options `--disable-dace` and `--disable-emvorado` are usually set by default. However, the open source release includes a wrapper script for the DWD platform, for example, that enables these packages, so this must be overridden.


```

../../config/dwd/rc1.VE.nfort
make -j6

cd ../../build/VH
../../config/dwd/rc1.VH.gcc
make -j6

```



Pre-compiled Binaries: Users of the NEC SX-Aurora system may find recent pre-compiled binaries in the following subdirectory `$NWP_BIN`. Note that the `nwp` module needs to be pre-loaded, and the directory path depends on whether the parent module `x86` (i. e. binaries for the Linux cluster) or module `sx` (i. e. binaries for the SX compute cluster) was previously loaded.

1.4. The DWD ICON Tools

The DWD ICON Tools provide a set of command line utilities for the pre- and post-processing of ICON model runs, in particular for remapping, extracting and querying ICON data files.

The DWD ICON Tools are not part of the open source release and are not generally available ICON software but were developed by the DWD as an add-on. Some chapters in this tutorial describe how to prepare input data using the ICON Tools. However, it should be emphasized that the ICON Tools are not the only method here, for example, the Climate Data Operators will be mentioned as an alternative.

All tool binaries can run in parallel on multi-core systems (OpenMP) and some offer an MPI-parallel execution mode in addition. The DWD ICON utilities use the `ecCodes` package for reading data in GRIB2 format. The `ecCodes` package is indirectly accessed by the Climate Data Interface (CDI).

Similar to the ICON model, the `dwd_icon_tools` directory contains a GNU Autotools `configure` script which, when run, scans the build environment and generates a `Makefile` appropriate for that build environment. The `configure` expects numerous platform-dependent options which can be listed by the command `configure -help`.

After the configuration has finished, the binary can be created by typing `make`.



Pre-compiled Binaries: Similar to the pre-compiled binaries of the ICON model, users of the NEC SX-Aurora system or the `rc1.dwd.de` may find recent pre-compiled binaries for the DWD ICON Tools in the following subdirectory `$NWP_BIN_UTIL`.

Note that the `nwp` module needs to be pre-loaded, and the directory path depends on whether the parent module `x86` (i. e. binaries for the Linux cluster) or module `sx` (i. e. binaries for the SX compute cluster) was previously loaded.

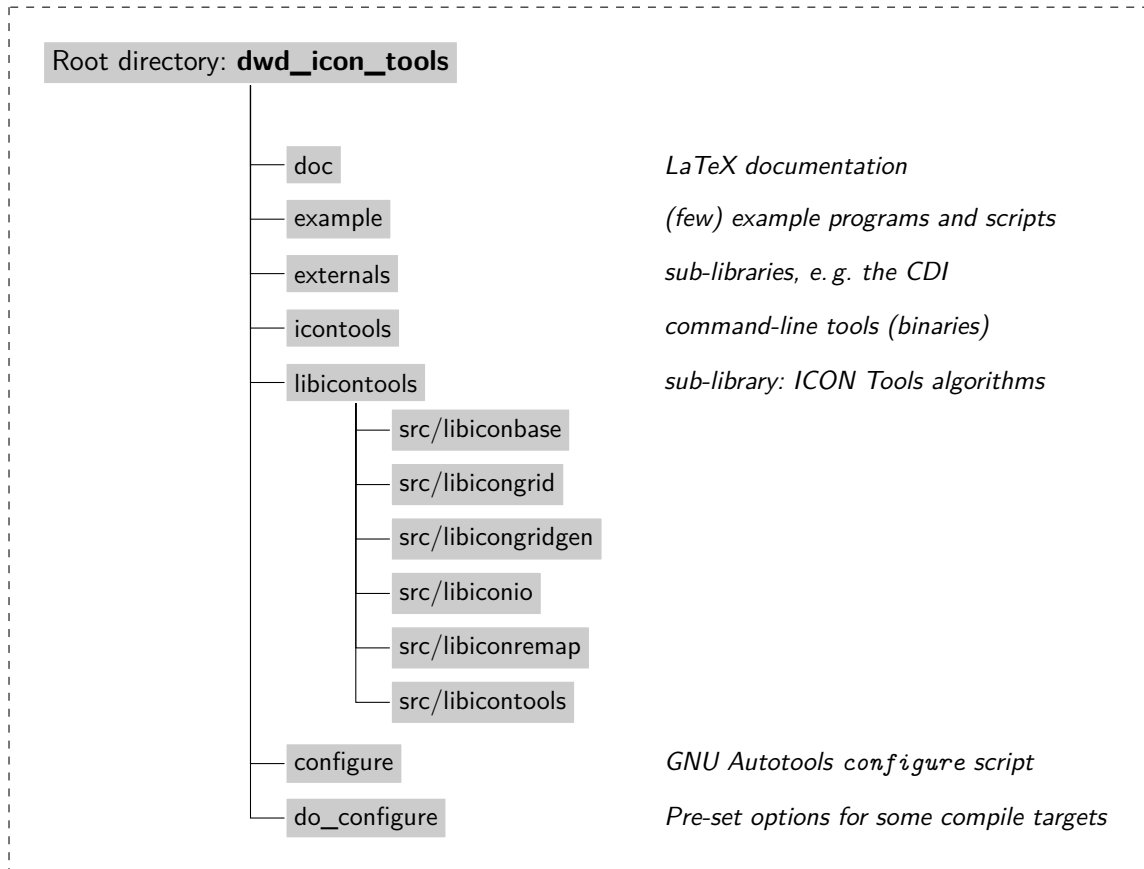


Figure 1.2.: Directory structure of the DWD ICON tools, published as a separate software package. Only the most relevant directories are shown.

The directory structure of the DWD ICON tools is shown in Fig. 1.2. We give a short overview over several tools in the following and refer to the documentation [Prill \(2020\)](#) for details.

ICONGRIDGEN

– Used in Section 2.1.5

The **icongridgen** tool is a simple grid generator. It creates icosahedral grids from scratch, which can be fed into the ICON model. Alternatively, an existing global or local grid file is taken as input and parts of this input grid (or the whole grid) are refined via bisection. No storage of global grids is necessary and the tool also provides an HTML plot of the grid. The grid generator provides the basis for the Zonda grid generator web tool (see Section 2.1.6).

ICONREMAP

– Used in Sections 2.2.3, 2.3

The **iconremap** utility is useful for pre-processing the initial data for the basic test setups in this manuscript. **iconremap** (*ICO*sahedral *N*onhydrostatic model *REMAP*ping) is a utility program for horizontally interpolating ICON data onto regular grids and vice versa.

Besides, it offers the possibility to interpolate between triangular grids of different grid spacing.

The `iconremap` tool reads and writes data files in GRIB2 or NetCDF file format. For triangular grids an additional grid file in NetCDF format must be provided.

Several interpolation algorithms are available: Nearest-neighbor remapping, radial basis function (RBF) approximation of scalar fields, area-weighted formula for scalar fields, RBF interpolation for wind fields from cell-centered zonal, meridional wind components u , v to normal and tangential wind components at edge midpoints of ICON triangular grids (and reverse), and barycentric interpolation.



Note that `iconremap` only performs a *horizontal* remapping, while the vertical interpolation onto the model levels of ICON is handled by the model itself.

ICONSUB

– Used in Section 2.3

The `iconsub` tool (*ICO*sahedral *Non*hydrostatic model *SUB*grid extraction) allows “cutting” sub-areas out of ICON data sets.

After reading a data set on an unstructured ICON grid in GRIB2 or NetCDF file format, the tool comprises the following functionality: It may “cut out” a subset, specified by two corners and a rotation pole (similar to the COSMO model). Alternatively, a boundary region of a local ICON grid, specified by parent-child relations, may be extracted. This execution mode is especially important for the setup of the limited area model ICON-LAM.

Multiple sub-areas can be extracted in a single run of `iconsub`. Finally, the extracted data is stored in GRIB2- or NetCDF file format.

ICONDELAUNAY

– see Section 7.1.2

The `icondelaunay` tool processes existing ICON grid files. It appends a Delaunay triangulation of the cell circumcenters to the grid file. This auxiliary triangulation can then be used for interpolation purposes.

1. Installation

2. Necessary Input Data

Before anything else, preparation is the key to success.

Alexander Graham Bell

Besides the source code of the ICON package and the technical libraries, several data files are needed to perform runs of the ICON model. There are four categories of necessary data: Horizontal grid files, external parameters, and data describing the initial state (DWD analysis or ECMWF IFS data). Finally, running ICON in limited area mode in addition requires accurate boundary conditions sampled at regular time intervals.

2.1. Horizontal Grids

In order to run ICON, it is necessary to load the horizontal grid information as an input parameter. This information is stored within so-called grid files. For an ICON run, at least one global grid is required. For model runs with nested domains, additional grids are necessary. Optionally, a reduced radiation grid for the global domain may be used (see Section 3.10).

The following nomenclature has been established: In general, by $RnBk$ we denote a grid that originates from an icosahedron whose edges have been initially divided into n parts, followed by k subsequent edge bisections. See Figure 2.1 for an illustration of the grid creation process. The total number of cells in a global ICON grid $RnBk$ is given by $n_{\text{cells}} := 20 n^2 4^k$. The cell circumcenters serve as data sites for most ICON variables. As an exception, the orthogonal normal wind is given at the midpoints of the triangle edges and is measured orthogonal to the edges.

The effective mesh size can be estimated as

$$\overline{\Delta x} \approx 5050 / (n 2^k) \text{ [km]}. \quad (2.1)$$

This formula is motivated as follows:

The average triangle area is calculated from the surface of the Earth divided by the number of triangles of the grid. Then, we imagine a square with the same area and define its edge length as the effective mesh size of the triangular grid. This results in

$$\overline{\Delta x} = \sqrt{S_{\text{earth}} / n_{\text{cells}}} = \sqrt{\frac{4 R_{\text{earth}}^2 \pi}{n_{\text{cells}}}} = \frac{R_{\text{earth}}}{n 2^k} \sqrt{\frac{\pi}{5}} \approx 5050 / (n 2^k) \text{ [km]},$$

where S_{earth} and R_{earth} define the Earth's surface and radius, respectively.

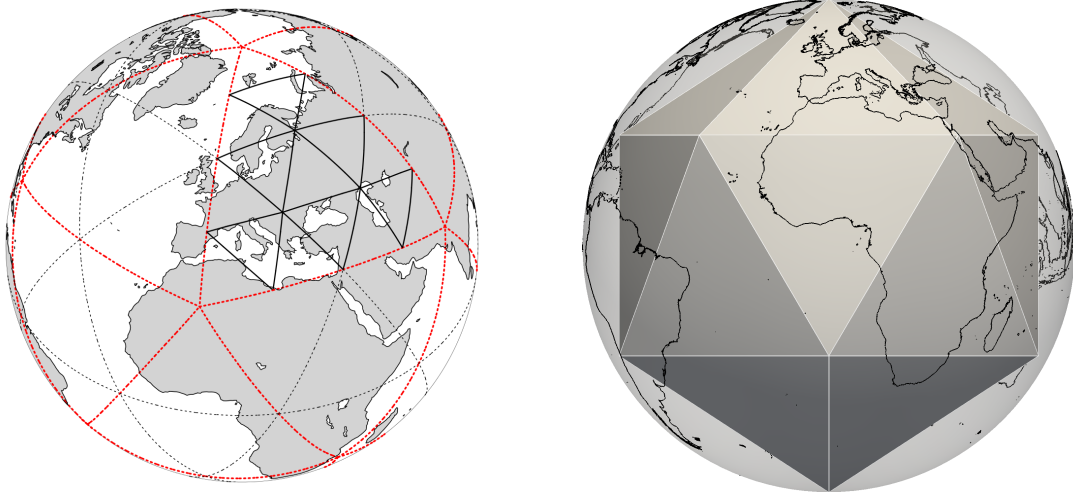


Figure 2.1.: *Left:* Illustration of the grid construction procedure. The original spherical icosahedron is shown in **red**, denoted as R1B00 following the nomenclature described in the text. In this example, the initial division ($n=2$; black dotted), followed by one edge bisection ($k=1$) yields an R2B01 grid (solid lines). *Right:* Graphical representation of the 12 pentagon points. Here, the base icosahedron is rotated exactly like the DWD operational grids.

Pentagon Points

Note that by construction, each vertex of a global grid is adjacent to exactly 6 triangular cells, with the exception of the original vertices of the icosahedron, the *pentagon points*, which are adjacent to only 5 cells.

The grids used in production at DWD have their pentagon points located at the following longitude/latitude positions (in degrees):

$$(-180, \pm 90), (0, -m), (-180, m), (\pm 36, m), (\pm 72, -m), (\pm 108, m), (\pm 144, -m)$$

where the constant m is derived from the golden ratio $\phi := \frac{1+\sqrt{5}}{2}$ as

$$m := 90 - \frac{180}{\pi} \operatorname{atan} \left(\frac{\sqrt{1 + \phi^2 + \phi^4}}{\phi} \right) \approx 26.565 \text{ [degrees]}.$$

See Fig. 2.1 for a graphical representation of the pentagon points.

Cell Neighborhoods

Apart from areas with pentagon cells, the number of cells N_c in an area covered by a central cell and N_r rings of cells around the central cell is

$$N_c(N_r) = 1 + 12 \sum_{r=1}^{N_r} r = 6 N_r (N_r + 1) + 1.$$

Then $N_c(1) = 13$, $N_c(2) = 37$, $N_c(3) = 73$, ...

Proof by mathematical induction. For the inductive step it is convenient to visualize all cells organized in horizontal rows. If the statement holds for N_r then we need to count the number of additional steps for $N_r + 1$: We have $2N_r + 1$ horizontal cell rows in the area covered by N_r . Therefore we get $4(2N_r + 1)$ new cells in these horizontal cell rows. Furthermore, we have two additional horizontal cell rows, containing $2(N_r + 1) + 1$ cells and $2(N_r + 2) + 1$ cells. In total we get the following number of new cells:

$$4(2N_r + 1) + 2(N_r + 1) + 1 + 2(N_r + 2) + 1 = 12(N_r + 1).$$

This is $N_c(N_r + 1) - N_c(N_r)$ from the formula above. \square

Dual Hexagonal Grid

The centers of the equilateral triangles contained in each triangle of the original icosahedron after triangulation are defined by the intersection of the angle bisectors (which are at the same time also altitudes) of the triangle. The centers of the triangles form a hexagonal grid that is called to be dual to the grid of triangle vertices. On the ICON grid, the centers of the slightly distorted triangles form a dual grid of slightly distorted hexagons.

Spring Dynamics Optimization

This grid on the sphere is optimized in a next step by so-called *spring dynamics*. We give the idea of the optimization only and refer to [Tomita et al. \(2002\)](#) for an in-depth description of the algorithm.

Imagine that we have a collection of springs all of them of the same strength and length. First, we attach a mass to each triangle vertex and fix it with glue on the circumscribed sphere. We replace each edge by one of the springs. Depending on the actual length of the edge, we have to tension some springs a bit more for the larger triangles, less for smaller ones. Now, the glue is melted away and the vertices move until an equilibrium is reached provided that there is some friction of the mass points on the sphere.

By this procedure, we will obtain a slightly different grid of triangles which are still slightly distorted and of unequal size, however, the vertices reached positions that reflect some “energy minimum”. These triangles are the basis of the ICON horizontal grid. Such a grid has particularly advantageous numeric properties. The North and South Pole of the Earth are chosen to be located at two vertices of the icosahedron that are opposite to each other.

2.1.1. ICON Grid Files

The unstructured triangular ICON grid resulting from the grid generation process is represented in NetCDF format. This file stores coordinates and topological index relations between cells, edges and vertices.

The most important data entries of the main grid file are

- **cell** (INTEGER dimension)
number of (triangular) cells

- **vertex** (INTEGER dimension)
number of triangle vertices
- **edge** (INTEGER dimension)
number of triangle edges
- **clon, clat** (double array, dimension: #triangles, given in radians)
longitude/latitude of the midpoints of triangle circumcircles
- **vlon, vlat** (double array, dimension: #triangle vertices, given in radians)
longitude/latitude of the triangle vertices
- **elon, elat** (double array, dimension: #triangle edges, given in radians)
longitude/latitude of the edge midpoints
- **cell_area** (double array, dimension: #triangles)
triangle areas
- **vertex_of_cell** (INTEGER array, dimensions: [3, #triangles])
The indices `vertex_of_cell(:,i)` denote the vertices that belong to the triangle *i*.
The `vertex_of_cell` index array is ordered counter-clockwise for each cell.
- **edge_of_cell** (INTEGER array, dimensions: [3, #triangles])
The indices `edge_of_cell(:,i)` denote the edges that belong to the triangle *i*.
- **clon/clat_vertices** (double array, dimensions: [#triangles, 3], given in radians)
`clon/clat_vertices(i,:)` contains the longitudes/latitudes of the vertices that belong to the triangle *i*.
- **neighbor_cell_index** (INTEGER array, dimensions: [3, #triangles])
The indices `neighbor_cell_index(:,i)` denote the cells that are adjacent to the triangle *i*.
- **zonal/meridional_normal_primal_edge** (INTEGER array, #triangle edges)
components of the normal vector at the triangle edge midpoints. Note that the edge's primal normal must not be mixed up with a primal cell's outer normal.
- **adjacent_cell_of_edge** (INTEGER array, [2, #triangle edges])
For cells i_1, i_2 adjacent to a given edge *i*, moving from i_1 to i_2 follows the direction of the edge's primal normal.
- **zonal/meridional_normal_dual_edge** (INTEGER array, #triangle edges)
These arrays contain the components of the normal vector at the facets of the dual control volume.
Note that each facet corresponds to a triangle edge and that the dual normal matches the direction of the primal tangent vector but signs can be different.
- **uuidOfHGrid** (global attribute)
Grid fingerprint (see Section 2.1.7).

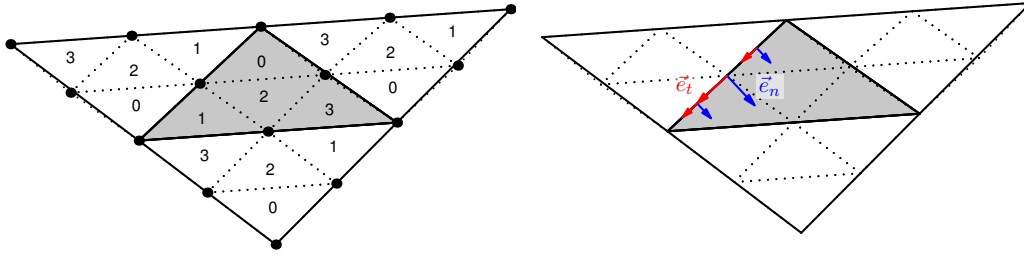


Figure 2.2.: Illustration of the parent-child relationship in refined grids. *Left:* Triangle subdivision and local cell indices. *Right:* The grids fulfill the ICON requirement of a right-handed coordinate system $[\vec{e}_t, \vec{e}_n, \vec{e}_w]$. Note: the primal tangent and the dual cell normal are aligned but do not necessarily coincide!

2.1.2. ICON “Nests”

ICON has the capability for running

- global simulations on a single grid,
- limited area simulations (see Chapter 6), and
- global or limited area simulations with refined nests (so called patches or domains).

For the subtle difference between nested and limited area setups the reader is referred to Section 6.1. Section 3.9.1 explains the exchange of information between the domains.

Additional topological information is required for ICON’s refined nests: Each “parent” triangle is split into four “child” cells, and each parent edge is split into two child edges. In the grid file only child-to-parent relations are stored while the parent-to-child relations are computed in the model setup. The local numbering of the four child cells (see Fig. 2.2) is also computed in the model setup.

The refinement information is stored in the following data entries of the grid file:

- `uuidOfParHGrid` (global attribute)
Fingerprint of the parent grid (see Section 2.1.7). If your grid does not contain the `uuidOfParHGrid` global attribute, then you’ll need the namelist parameter `dynamics_parent_grid_id`, see below.
- `parent_cell_index` (INTEGER array, dimension: `#triangles`)
Global index of coarser parent triangle in the parent grid.
- `parent_edge_index` (INTEGER array, dimension: `#triangle edges`)
Global index of parent edge (with double length) in the parent grid.
- `parent_vertex_index` (INTEGER array, dimension: `#triangle vertices`)
Global index of parent vertex in the parent grid.

Since the grids of the different refinement levels are stored in separate files, the usual way to establish a parent-child relationship between these grids is to read the header information

from the list of provided grid files, see Section 4.1.2. Then, the parent-child relationships can be inferred from the NetCDF attributes `uuidOfHGrid` and `uuidOfParHGrid`.

However, there are still older grid files in circulation which do not contain these descriptive attributes. In this case there is no other possibility than defining the parent-child relationship by a namelist parameter.

dynamics_parent_grid_id (namelist `grid_nml`, list of `INTEGER` values)

This parameter array is closely related to the namelist parameter `dynamics_grid_filename`. The i^{th} entry of `dynamics_parent_grid_id` contains the index of the parent grid of domain i . Indices start at 1, an index of 0 indicates *no parent*.

For older grid files that are still in circulation, the refinement information may be provided in a separate file (suffix `-grfinfo.nc`), which happens to be the case especially for legacy data sets. This optional *grid connectivity* file acts as a fallback at model startup if the expected information is not found in the main grid file.

Finally, note that the data points on the triangular grid are the cell circumcenters. Therefore the global grid data points are located closely to nest data points, but they *do not coincide* exactly.

2.1.3. Mapping of Geodesic Coordinates to the Sphere

Usually, geographic coordinate data is given with respect to an ellipsoidal reference system. The WGS84 ellipsoid, for example, is given by the following semi-major and minor axes:

$$a := 6.378137 \cdot 10^6 \text{ m} , \quad b := 6.356752314245 \cdot 10^6 \text{ m}$$

Input data usually refers to the *geographic (geodetic) latitude* of this ellipsoid, which is the angle that a line perpendicular to the surface of the ellipsoid at the given point makes with the plane of the Equator, see the blue line in Fig. 2.3. It must not be confused with the geocentric latitude (red in Fig. 2.3), which is the angle made by a line to the center of the ellipsoid with the equatorial plane, see Snyder (1987). A post-processing tool that misinterprets geocentric and geographic latitudes will notice a shift of up to 22 km.

The ICON model, however, uses a *spherical* approximation with an earth radius of

$$r_e = 6.371229 \cdot 10^6 \text{ m}.$$

Generally speaking, a mapping rule needs to be defined between the ellipsoid and the sphere. For the ICON model this mapping is prescribed by the ExtPar tool which pre-processes numerous invariant parameter fields, e.g. the topography, the land-sea mask, the soil type and atmospheric aerosols. The ExtPar data set will be in detail described in Section 2.4.

Here, the important fact is that the ExtPar tool re-interprets coordinates with a trivial mapping rule, i.e. WGS84 latitudes are directly applied without transformation to the sphere.

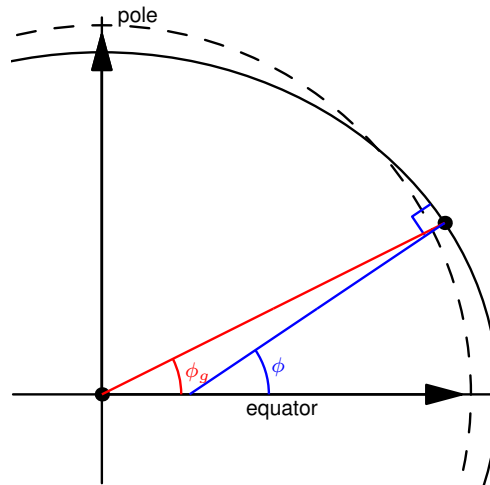


Figure 2.3.: Illustration of an ellipsoid with geodetic (geographic) latitude ϕ and geocentric latitude ϕ_g . The dashed line shows a spherical surface for comparison.

This definition must be kept consistent for all pre-processing parts of the model (grids, namelists, etc.): For example, when the user specifies meteogram locations, he usually applies a mapping from ellipsoidal to spherical coordinates, often without realizing this transformation. For ICON, however, the user must provide WGS84 coordinates to comply with the calculation rule of the ExtPar tool. The same argument holds for point source locations, geometric tracks, the center and corner locations of a grid, etc.

2.1.4. Download of Predefined Grids

For fixed domain sizes and resolutions a list of grid files has been pre-built for the ICON model together with the corresponding reduced radiation grids and the external parameters.

The contents of the primary storage directory are regularly mirrored to a public web site for download, see Figure 2.4 for a screenshot of the ICON grid file server. The download server can be accessed via

<http://icon-downloads.mpimet.mpg.de>

The pre-defined grids are identified by a *centre number*, a *subcentre number* and a *numberOfGridUsed*, the latter being simply an integer number, increased by one with every new grid that is registered in the download list. Also contained in the download list is a tree-like illustration which provides information on parent-child relationships between global and local grids, and global and radiation grids, respectively.

Also available on the grid file server are ICON grids provided by the Max Planck Institute for Meteorology (MPI-M). Note, however, that MPI grids generally have no matching reduced radiation grid, and are rotated 37° around the z-axis, while DWD grids are rotated

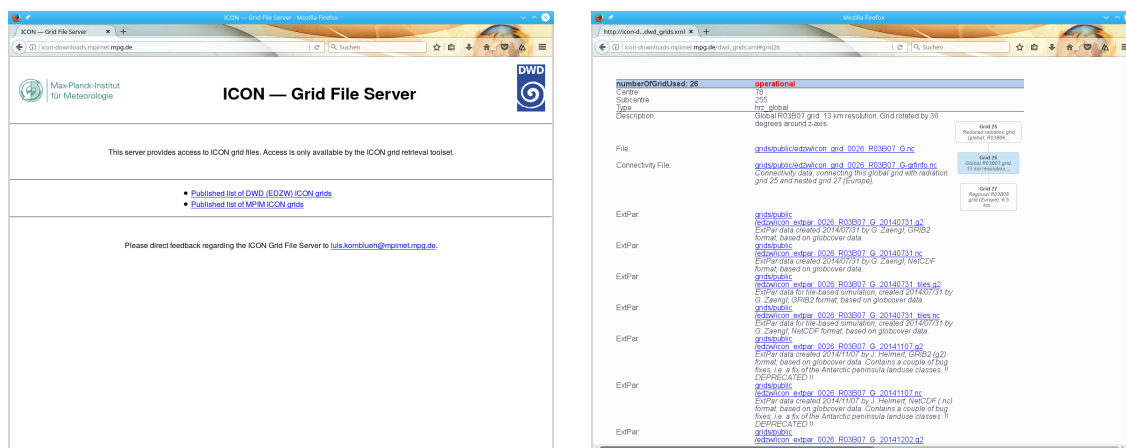


Figure 2.4.: Screenshots of the ICON download server hosted by the Max Planck Institute for Meteorology in Hamburg.

36°. Finally, note that the grid information of some of the older DWD grids (no. 23 – 40) is split over two files: The user needs to download the main grid file itself *and* a *grid connectivity* file (suffix `-grfinfo.nc`).

2.1.5. Grid Generator: Invocation from the Command Line

There are (at least) three grid generation tools available for the ICON model: One grid generation tool has been developed at the Max Planck Institute for Meteorology by L. Linardakis¹. Second, in former releases, the ICON model itself was shipped together with a standalone tool `grid_command`. This program has finally been replaced by another grid generator which is contained in the DWD ICON Tools.

In this section we will discuss the grid generator `icongridgen` that is contained in the DWD ICON Tools, because this utility also acts as the backend for the publicly available web tool. The latter is shortly described in Section 2.1.6. It is important to note, however, that this grid generator is not capable of generating non-spherical geometries like torus grids, see Section 2.1.8.



Minimum version required: Grid files that have been generated by the `icongridgen` tool contain only child-to-parent relations while the parent-to-child relations are computed in the model setup. Therefore these grids only work with ICON versions newer than ~ September 2016.

Grid Generator Namelist Settings

The DWD ICON Tools utility `icongridgen` is mainly controlled using a Fortran namelist.

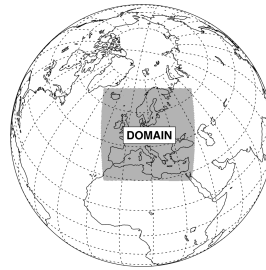
¹see the repository <https://gitlab.dkrz.de/mpim-sw/grid-generator>.

The command-line option that is used to provide the name of this file and other available settings are summarized via typing

```
icongridgen --help
```

The Fortran namelist `gridgen_nml` contains the filename of the parent grid which is to be refined and the grid specification is set for each child domain independently. For example (COSMO-EU nest) the settings are

```
dom(1)%region_type = 3
dom(1)%lrotate     = .true.
dom(1)%hwidth_lon  = 20.75
dom(1)%hwidth_lat  = 20.50
dom(1)%center_lon  = 2.75
dom(1)%center_lat  = 0.50
dom(1)%pole_lon    = -170.00
dom(1)%pole_lat    = 40.00
```



For a complete list of available namelist parameters we refer to the documentation ([Prill, 2020](#)).

The `icongridgen` grid generator checks for overlap with concurrent refinement regions, i.e. no cells are refined which are neighbors or neighbors-of-neighbors (more precisely: vertex-neighbor cells) of parent cells of another grid nest on the same refinement level. Grid cells which violate this distance rule are “cut out” from the refinement region. Thus, there is at least one triangle between concurrent regions.



Minimum distance between child nest boundary and parent boundary: A second, less well-known constraint sometimes leads to unexpected (or even empty) result grids: In the case that the parent grid itself is a bounded regional grid, no cells can be refined that are part of the indexing region (of width `bdy_indexing_depth`) in the vicinity of the parent grid’s boundary.

Settings for ICON-LAM

When the grid generator `icongridgen` is targeted at a limited area setup (for ICON-LAM), two important namelist settings must be considered:

- *Identifying the grid boundary zone.* In Section 2.3 we will describe how to drive the ICON limited area model. Creating the appropriate boundary data makes the identification of a sufficiently large boundary zone necessary.

This indexing is enabled through the following namelist setting in `gridgen_nml`:
`bdy_indexing_depth = 14.`

This means that 14 cell rows starting from the nest boundary are indexed and can be identified in the ICON-LAM setup, which is described in Section 2.3. See Fig. 2.12 for an illustration of such a boundary zone.

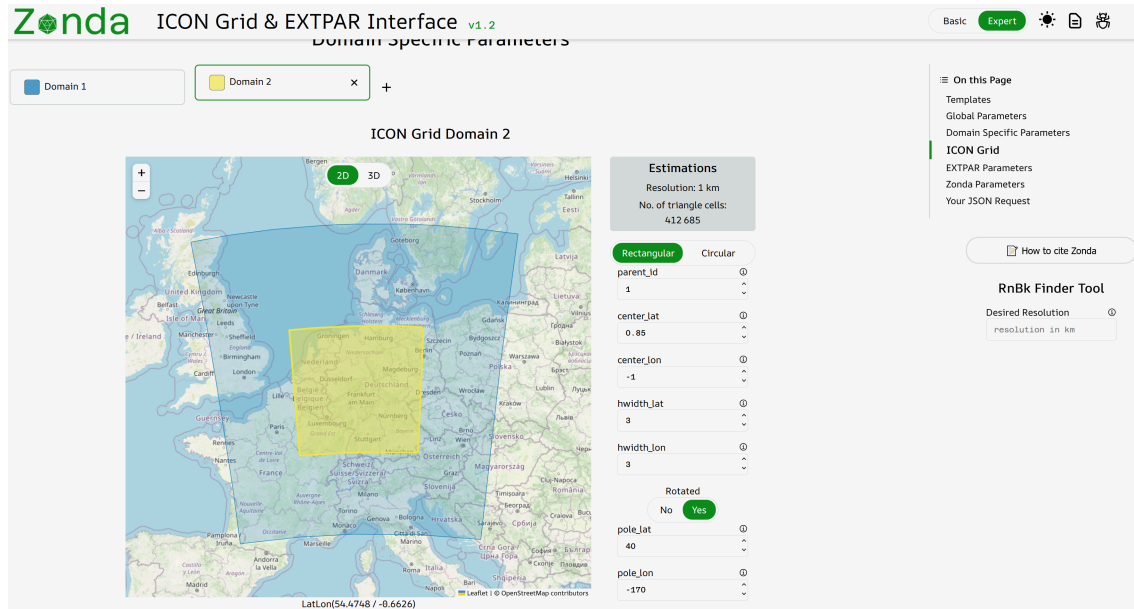


Figure 2.5.: Web browser screenshot of the Zonda tool.

- *Generation of a coarse-resolution radiation grid* (see Section 3.10 for details).

The creation of a separate (local) parent grid with suffix `*.parent.nc` is enabled through the following namelist setting in `gridgen_nml`:

```
dom(:)%lwrite_parent = .TRUE.
```

Note that a grid whose child-to-parent indices are occupied by such a coarse grid can no longer be used in a nested configuration together with a global grid.

2.1.6. Grid Generator: Invocation via the Zonda Web Interface

Zonda is a web interface designed to facilitate the generation of ICON grid files and External Parameter data (ExtPar) on ICON triangular grids for research and on-demand simulations. Zonda makes use of containerized versions of the ICON grid generator (see Section 2.1.5) and ExtPar (see Section 2.4). It constructs the Fortran and Python namelist setups and runs the containers on a public server.

The invocation of Zonda is realized as a two-step process:

- In the frontend (<https://zonda.ethz.ch/>), the user specifies the domain(s) including the appropriate settings for the external parameter generation. Example configurations and an expert mode with additional choices are available to assist the user. The user gets a JSON code snippet containing the chosen configuration which is required for the second step.
- In the backend (<https://github.com/C2SM/zonda-request>), the JSON code has to be pasted into a Github issue. Then, the Github CI triggers the generation of the

ICON Grid and ExtPar data. These files in NetCDF 4 format are then provided as .zip file.

The Zonda web interface comes with documentation <https://zonda.ethz.ch/docs>. Furthermore, the available options in the frontend contain tooltips with short descriptions and links to the documentation of the respective option.



Access to Zonda: You need only a Github account to use Zonda.

2.1.7. Which Grid File is Related to My Simulation Data?

ICON data files do not (completely) contain the description of the underlying grid. This is an important consequence of the fact that ICON uses unstructured, pre-generated computational meshes which are the result of a relatively complex grid generation process. Therefore, given a particular data file, one question naturally arises: *Which grid file is related to my simulation data?*

The answer to this question can be obtained with the help of two meta-data items which are part of every ICON data and grid file (either a NetCDF global file attribute or a GRIB2 meta-data key):

- **numberOfGridUsed**
This is simply an integer number, as explained in the previous sections. The **numberOfGridUsed** helps to identify the grid file in the public download list. If the **numberOfGridUsed** differs between two given data files, then these are not based on the same grid file.
- **uuidOfHGrid**
This acronym stands for *universally unique identifier* and corresponds to a binary data tag with a length of 128 bits. The UUID can be viewed as a fingerprint of the underlying grid. Even though this is usually displayed as a hexadecimal number string, the UUID identifier is not human-readable. Nevertheless, two different UUIDs can be tested for equality or inequality.

The meta-data values for **numberOfGridUsed** and **uuidOfHGrid** offer a way to track the underlying grid file through all transformations in the scientific workflow, for example in

- external parameter files
- analysis data for forecast input
- data files containing the diagnostic output
- checkpointing files (restarting).

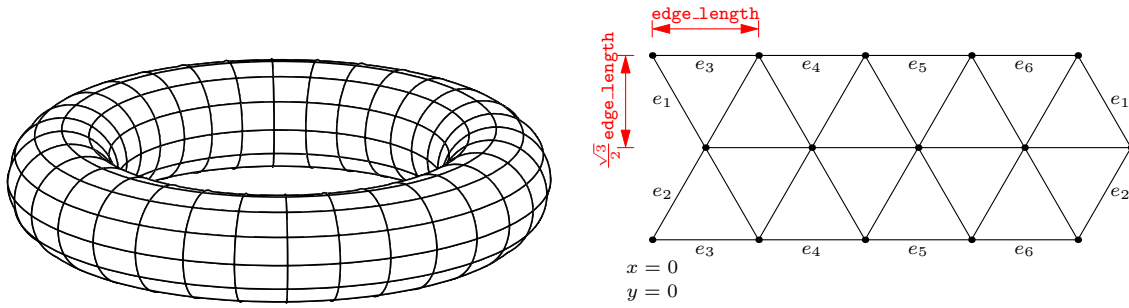


Figure 2.6.: Topological representation of the torus geometry and its triangulation.

2.1.8. Planar Torus Grids

As a special mode for numerical experiments, ICON allows for planar torus grids. Note that the torus geometry and the corresponding global meta-data (NetCDF attributes) are *not* generated by the "standard grid generator" in Section 2.1.5, but require L. Linardakis' grid generator tool².

The torus grid has double periodic boundaries. It consists of equal-sided triangles, with edge length `edge_length`, which is a namelist parameter of the grid generator, and height $\frac{\sqrt{3}}{2} \text{edge_length}$, see Fig. 2.6.

The lon-lat parameterization of the torus is

$$(\text{lon}, \text{lat}) = [0, 2\pi] \times [-\text{max_lat}, \text{max_lat}]$$

where $\text{max_lat} := \frac{\pi}{18} \equiv 10$ degrees (hard-coded in the torus grid generator). Variables related to the lon-lat parameterization are stored as the data type `t_geographical_coordinates` in the ICON code.

The Cartesian coordinates of the torus grid are: $v = (x, y, 0)$ where

$$(x, y) \in [0, \text{domain_length}] \times [0, \text{domain_height}]$$

The lengths `domain_length`, `domain_height` are stored as global attributes in the grid file. Variables related to the Cartesian mesh are stored as the data type `t_cartesian_coordinates` in the ICON code.

2.2. Initial Conditions

Global numerical weather prediction (NWP) is an initial value problem. The ability to make a skillful forecast heavily depends on the accuracy with which the present atmospheric (and surface/soil) state is known. In addition to that, running forecasts with a limited area model requires accurate boundary conditions sampled at regular time intervals.

²see the repository <https://gitlab.dkrz.de/mpim-sw/grid-generator>.

Initial conditions are usually generated by a process called *data assimilation*. Data assimilation combines irregularly distributed (in space and time) observations with a short term forecast of a general circulation model (e.g. ICON) to provide a "best estimate" of the current atmospheric state. Such *analysis products* are provided by several global NWP centers.

In the following we will present various data sets that can be used to drive the ICON model and explain how these data can be retrieved. In addition we will explain how these data can be remapped to the targeted ICON grid, if necessary. Remapping is one of the basic pre-processing steps which are visualized in Figure 2.7.

In general, each computational domain, i.e. also a nested domain, requires a separate initial data file. A "workaround" to start a nested simulation without the need to provide initial data for the nest is discussed in Section 5.2.

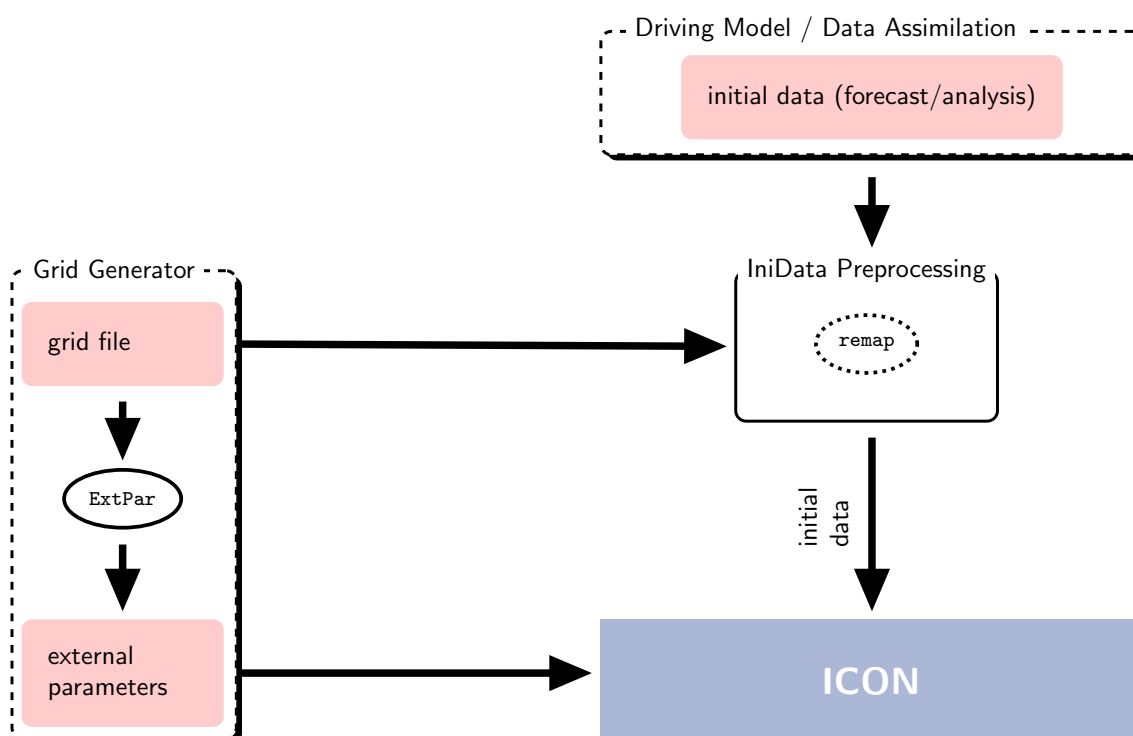


Figure 2.7.: Basic pre-processing steps for ICON (without limited area mode) which include the generation of grids and external parameters as well as the remapping of initial conditions. The grid generation process and the external parameters are described in the Sections 2.1 and 2.4. The initial data processing is covered by Section 2.2.3.

2.2.1. Obtaining DWD Initial Data

The most straightforward way to initialize ICON is to make use of DWD's analysis products, which are generated operationally every 3 hours and stored in GRIB2 format on

the native ICON grid. Deterministic as well as ensemble analysis products are available. Deterministic products are generated by a hybrid Ensemble Variational Data assimilation (En-Var), which combines variational and ensemble methods. Ensemble products are based on a Localized Ensemble Transform Kalman Filter (LETKF) approach. See Chapter 11 for more information on DWD's data assimilation system.

Choosing the Right Product

DWD provides a set of different analysis products. They all constitute a "best estimate" of the atmospheric state, but differ in some physical and technical aspects. Choosing the right product is crucial and depends on the targeted application.

Some of the analysis products consist of two files: a first guess file and an analysis file. The term *first guess* denotes a short-range forecast of the NWP model at hand, whereas the term *analysis* denotes all fields which have been updated by the assimilation system.

Several combinations of these files exist, with specific pros and cons:

Uninitialized analysis for IAU

This product is meant for starting the model in *Incremental Analysis Update* (IAU) mode. IAU is a model internal filtering technique for reducing spurious noise introduced by the analysis (see Section 11.3). This product consists of a first guess file and an analysis file. The latter contains analysis *increments*, which is the difference between the analysis and the first guess. The validity dates of both files differ. The validity date of the first guess is shifted ahead of the analysis date by 90 min for the global analysis and by 5 min for the regional ICON-D2 analysis.

While this initialization method performs best in terms of noise reduction, it bears the disadvantage that the corresponding analysis product cannot be interpolated horizontally in a straightforward manner. This prevents its use on custom target grids. The underlying reason is that the analysis product contains tiled surface data. Remapping of tiled data sets makes no sense, since the tile-characteristics can differ significantly between individual source and target grid cells. Only *aggregated* surface fields can safely be remapped (see Section 3.8.9 for more details on the surface tile approach).

A field inventory list is provided in Section 11.4.1.

Plain uninitialized analysis

This product consists of a first guess file and an analysis file, with the latter containing *full* analysis fields instead of increments. The validity date of both files matches the analysis date.

When using this product, the model state is abruptly pulled towards the analyzed state right before the first time integration step. Thus, no noise filtering procedure is included. This conceptually easy approach comes at the price of a massively increased noise level at model start. Due to the lack of tiled surface data, this product can be interpolated horizontally to arbitrary custom target grids without any hassle.

A field inventory list is provided in Section 11.4.2.

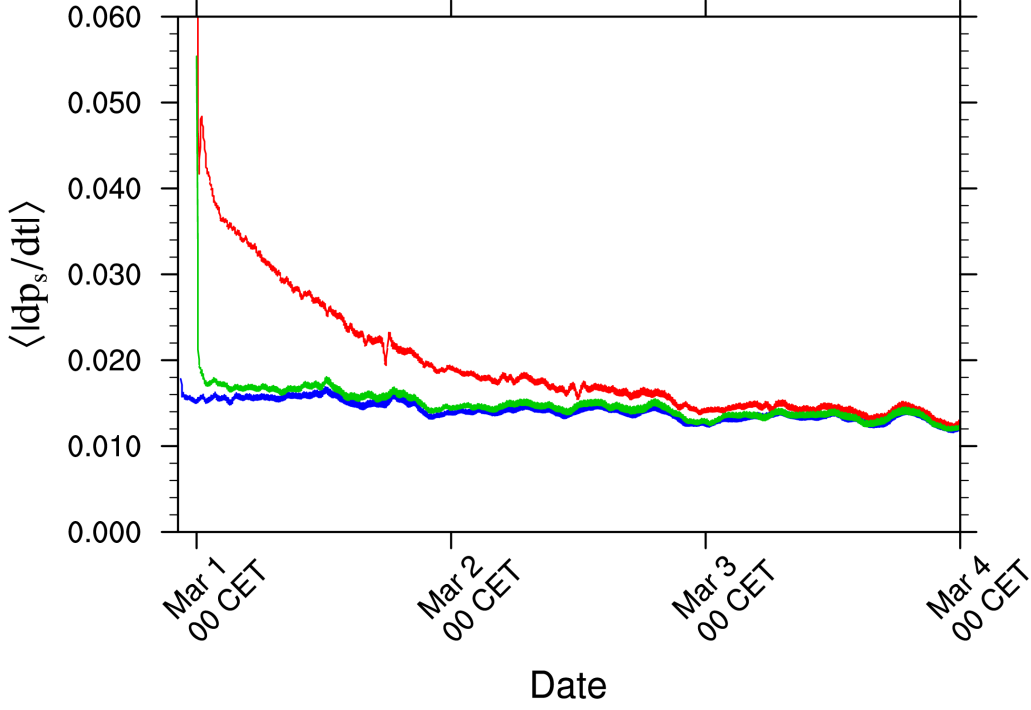


Figure 2.8.: Area averaged absolute surface pressure tendency in hPa as a function of simulation time for a deterministic global model run. Curves differ in terms of the way the model is initialized, with the *uninitialized analysis for IAU* in blue, the *uninitialized analysis* in red and the *initialized analysis* in green.

Initialized analysis

This product consists of a single file only, containing the analyzed state. First guess and analysis fields have already been merged and filtered by means of an asymmetric IAU (see Section 11.3). The noise level induced by this product is very moderate. In addition, this product can safely be interpolated to arbitrary custom target grids.

A field inventory list is provided in Section 11.4.3.

The level of spurious noise that emerges from each of these analysis products is compared in Figure 2.8. It shows the area averaged absolute surface pressure tendency as a function of simulation time, which is a measure of spurious gravity-noise induced by spurious imbalances in the initial conditions. It is defined as

$$\begin{aligned} \left\langle \left| \frac{dp_s}{dt} \right| \right\rangle &= \frac{1}{A} \sum_i \left| \frac{dp_s}{dt} \right| \Delta a_i \\ &= \frac{1}{A} \sum_i \left(\sum_k |-g \nabla_h \cdot (\bar{\rho} \hat{\mathbf{v}}_h) \Delta z_k| \right) \Delta a_i, \end{aligned}$$

with A denoting the earth's surface and Δa_i denoting the area of the i th cell. The mathematical steps to obtain the pressure tendency equation are discussed at the end of Section 3.2. It can be seen that for the uninitialized analysis (red line) the noise level at simulation start is significantly increased when compared to the other two products. It takes about two days of model forecast for the noise levels to align. The uninitialized analysis for IAU (blue) performs best in terms of noise-level, but keep in mind that some data

Table 2.1.: Characteristics of DWD’s analysis products. The recommended product for standalone model runs (without data assimilation) is highlighted in blue .

	Uninitialized analysis for IAU	Uninitialized analysis	Initialized analysis
# of files	2	2	1
noise level	low	high	moderate
analysis increments	yes	no	no
surface tile information	yes	no	no
interpolation possible	no	yes	yes
available for det/ens	yes/yes	yes/yes	yes/yes

fields cannot easily be interpolated in the horizontal, such that the application of this mode is typically restricted to the horizontal grids used operationally at DWD (Det/Ens/D2: 13 km/26 km/2 km grid spacing)

The specific pros and cons of the different analysis products is summarized in Table 2.1.



Important note: For external users we strongly recommend to use the *initialized analysis* for model initialization, since it constitutes a good compromise between accuracy and practicability.

Downloading Initial Conditions

ICON initial conditions are stored in DWD’s meteorological data management system SKY. Here, for better performance, meta and binary data are stored separately: The meta data are stored in a relational database, sorted by data category and time. The binary data are stored temporarily on a hard drive and subsequently moved into the DWD’s tape archive using the archiving components in SKY.

A prerequisite for data retrieval is a valid account for the database “roma”. The database can be accessed in the following ways:

- If you have access to DWD’s Linux cluster rcn1*, please contact klima.vertrieb@dwd.de, in order to gain additional access to the database. Data retrieval will then be possible by using either SKY’s query language directly, or by using the PAMORE command-line tool (the latter will be explained below).
- An alternative way to access the database is to use the web-based PAMORE service, see the web page

<https://www.dwd.de/EN/ourservices/pamore/pamore.html>

This website is meant for external users who do not have direct access to DWD's computer systems and requires a user account. To this end please fill out the registration form

Registration form for PAMORE web service



DWD's operational analysis and forecast products for the ICON model are being stored in the SKY database since 2015-01-20. However, the set of data fields stored is subject to continuous changes and improvements. I.e. the inclusion of additional fields has become necessary with the activation of more advanced physical parameterizations. Similarly, horizontal and vertical resolution have been increased, as more powerful HPC systems became available. In November 2022 the number of vertical levels of the global deterministic and ensemble system has changed from 90 to 120 levels. At the same time the horizontal resolution of the ensemble system has been enhanced from 40 km to 26 km.

Further note that the set of data fields of the early months is likely to be incomplete with regard to the initialization procedure that is explained in this tutorial. For example, the surface tile approach (see Section 3.8.9) has been activated no earlier than December 2015.

Data retrieval with PAMORE via command-line. PAMORE (*PARallel MOdel data RETrieve* from Oracle databases) is a high-level tool for the retrieval of (model) data from DWD's meteorological data management system SKY.

A full set of command-line options can be obtained via `pamore -h`. Alternatively, they are accessible on the web via

<https://webservice.dwd.de/pamore.html>

In order to retrieve, for example, initial data on the native ICON grid from February 1, 2019 00 UTC, the following command lines can be used for the different analysis products:

Uninitialized analysis for IAU (deterministic)

Global domain with 13 km grid spacing

```
pamore -d 2019020100 -lt m -iglo_startdata -iau
```

year
month
day
hour

Global domain (13 km) and nested EU domain (6.5 km)

```
pamore -d 2019020100 -lt m -iglo_eu_startdata -iau
```

Note that the EU domain lacks a separate analysis for the atmosphere. If required, it must be interpolated (horizontally) from the global domain.

Uninitialized analysis (deterministic)

Global domain with 13 km grid spacing

```
pamore -d date -lt m -iglo_startdata
```

Global domain (13 km) and nested EU domain (6.5 km)

```
pamore -d date -lt m -iglo_eu_startdata
```

Here, *date* must be replaced by the desired date in the format YYYYMMddhh (see above).

Initialized analysis (deterministic)

Global domain with 13 km grid spacing

```
pamore -d date -hstart 0 -hstop 0 -lt a \
      -model iglo -iglo_startdata_0
```

Nested EU domain (6.5 km):

```
pamore -d date -hstart 0 -hstop 0 -lt a \
      -model ieu -iconlam_startdata_0
```

ICON-D2 limited area domain (2.1 km):

```
pamore -d date -model ilam -iles_startdata_0
```

Please note that for the nested domain the optional input field **TKE** is not available and that **Z0** and **H_SNOW** are only available since 2018-03-14.



Important note for ensemble products: With the additional options

- **-ires** r3b06 and
- **-enum** *num*, where *num* specifies an ensemble member (e.g. 3) or a range of ensemble members (e.g. 3 – 8),

analysis products can also be picked from the global LETKF analysis ensemble consisting of 40 members with 26/13 km horizontal grid spacing. This does, however, not hold for initialized analysis products.

Initialized analysis products for ICON ensemble members are not archived and, hence, are not available from DWD's database.

Data retrieval with PAMORE via the web form. The PAMORE web service allows the user-defined selection of model fields by navigating through a sequence of HTML forms. Alternatively, the web site offers a plain command-line interface.

For the specific task of retrieving ICON initial data, we strongly suggest to take the latter path. By making direct use of the above PAMORE command-lines and pasting them into the HTML form (see Figure 2.9), you can minimize the risk of missing some fields.

After submitting your database request, the data will be extracted from the database and stored on an FTP server for download. Once your request has been processed, you will

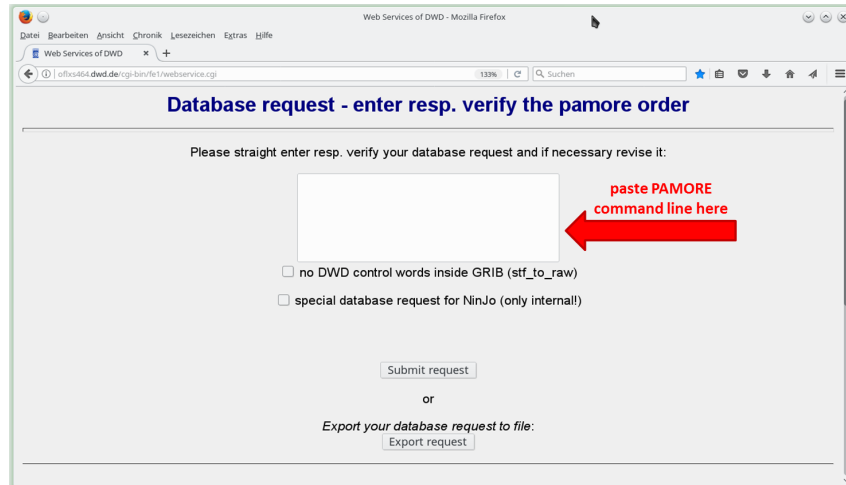


Figure 2.9.: Screenshot of the PAMORE web service. It shows the HTML form where you can place your PAMORE command line request directly.

receive an e-mail with information about the FTP server address and the path to your data.

Important note: This web-based service is currently out of maintenance after several years of operation. It should be used without warranty.

2.2.2. Obtaining ECMWF IFS Initial Data

Model runs may also be initialized by “external” analysis files produced by the Integrated Forecasting System (IFS) that has been developed and is maintained by the European Centre for Medium-Range Weather Forecasts (ECMWF).

The ICON code contains a script for the automatic request for IFS data from the MARS data base on non-rotated regular lat-lon grids. The Meteorological Archival and Retrieval System (MARS)

<https://confluence.ecmwf.int/display/UDOC/MARS+user+documentation>

is the main repository of meteorological data at ECMWF. A full list of recommended IFS analysis fields is provided in Table 2.2.

The script for importing from MARS must be executed on the ECMWF computer system. It is located in the subdirectory

`icon/scripts/preprocessing/mars4icon_smi`

In order to retrieve, for example, T1279 grid data with 137 levels for the July 1, 2013, the following command line is used:

```
./mars4icon_smi -r 1279 -l 1/to/137 -d 2013070100 -O -L 1 -o 20130701.grb -p 5
```

Table 2.2.: Recommended **IFS analysis fields** on a regular lat-lon grid, as retrieved by the script `mars4icon_smi`. Optional fields are marked in [blue](#). The second column indicates ICON's query name during read in. This is the name which the field must be given to when it is remapped onto the native ICON grid (see also Section [2.2.3](#)).

shortName ECMWF	shortName ICON	Unit	Description
Atmosphere			
U, V	U, V	m s^{-1}	horizontal velocity components
OMEGA	W	Pa s^{-1}	vertical velocity
T	T	K	temperature
FI	GEOP_ML	$\text{m}^2 \text{s}^{-2}$	model level geopotential (only the surface level is required)
QV	QV	kg kg^{-1}	specific humidity
CLWC	QC	kg kg^{-1}	cloud liquid water content
CIWC	QI	kg kg^{-1}	cloud ice content
CRWC	QR	kg kg^{-1}	rain water content
CSWC	QS	kg kg^{-1}	snow water content
LNPS	LNPS	-	logarithm of surface pressure
Soil/Surface			
SST	SST	K	sea surface temperature
CI	CI	[0,1]	sea-ice cover
Z	GEOP_SFC	$\text{m}^2 \text{s}^{-2}$	surface geopotential
TSN	T_SNOW	K	snow temperature
SD	W_SNOW	m of water eqv.	water content of snow
RSN	RHO_SNOW	kg m^{-3}	density of snow
ASN	ALB_SNOW	[0,1]	snow albedo
SKT	SKT	K	skin temperature
STL[1/2/3/4]	STL[1/2/3/4]	K	soil temperature level 1/2/3/4
SWVL[1/2/3/4]	SMIL[1/2/3/4]	$\text{m}^3 \text{m}^{-3}$	soil moisture index (SMI) layer 1/2/3/4
SRC	W_I	m of water eqv.	water content of interception storage
LSM	LSM	[0,1]	land/sea mask

Further options are shown by typing `./mars4icon_smi -h`

Note that prior to 2013-06-25 12 UTC, only 91 instead of 137 vertical levels were used by the operational system at ECMWF. For more information, regarding changes in the ECMWF model, see

<https://www.ecmwf.int/en/forecasts/documentation-and-support/changes-ecmwf-model>

The `iconremap` tool contains an example script `xce_ifs2icon.run` for the interpolation of IFS data onto the ICON grid.

When initializing from “external” analysis files, ICON requires the soil moisture index `SMI` and not the volumetric soil moisture content `SWV` as input. The conversion from `SWV` to `SMI` is performed as part of the MARS request (`mars4icon_smi`). However, this conversion is not reflected in the variable short names. The fields containing `SMI` are still named `SWVL x` , with x denoting the soil layer index. The ICON model, however, expects them to be named `SMI x` . Therefore, the proper output name `SMI x` must be specified explicitly in the namelist `input_field_nml` of `iconremap` (see the example namelist on p. 43).

Other field names must be adjusted as well during the remapping process described in Section 2.2.3. In order to read the data with ICON, it is necessary to rename the fields to the ICON-internal short names, according to the second column of Table 2.2.

If IFS data are retrieved on rotated lat-lon grids, care must be taken regarding the definition of the basis vectors of vector quantities (i.e. horizontal wind components). Usually, basis vectors of vector quantities are rotated in accordance with the rotated coordinate axes. This, however, would be inconsistent with the implicit assumption made by the `iconremap` tool, where the local basis vectors always point into zonal and meridional directions. Therefore, before the remapping tool can be applied, it is necessary to rotate any vector quantities into the zonal/meridional geographical reference system. Please see also the discussion on ICON’s native interpolation onto lat-lon grids in Section 7.1.3.



Note on IFS vertical coordinate: During the initialization phase of ICON, the 3D height coordinate field for IFS input data is derived from three fields: the surface geopotential, the surface pressure and the virtual temperature on model levels. As some IFS fields use orography in transformed space and some do not, it is important to choose consistent fields. The MARS script provided with the ICON code usually retrieves the lowermost level of the 3D geopotential `GEOP_ML` (FI in IFS naming), and the logarithm of the surface pressure `LNPS` (LNPS in IFS naming). Alternatively, `GEOP_SFC` (Z in IFS naming) might also be used to create the 3D vertical height field, but this requires the usage of `PS` instead of `LNPS`, in order to be consistent.

When using IFS data for surface initialization, `GEOP_SFC` must be available for topography-dependent corrections of the soil temperatures. As a fallback, `GEOP_ML` can be used instead of `GEOP_SFC`.

ICON does not perform any cross checks for this. The best way is to provide `GEOP_ML` and `GEOP_SFC` together with `LNPS` to ICON.

2.2.3. Remapping Initial Data to Your Target Grid

Often it is desirable to run ICON at horizontal resolutions which differ from those of the available initial data. Common examples are high-resolution limited area runs, which start from operational ICON analysis, or the initialization from IFS analysis, which is provided on a lat-lon grid (see Section 2.2.2). In these cases horizontal remapping of the initial data is necessary. Note that there is no need for vertical interpolation as a separate pre-processing step. The ICON model itself will take care of the interpolation onto the

model levels, assumed that the user has provided the height level field `HHL` and set the appropriate namelist options (see the namelist parameter `init_mode`).

After the successful download, the analysis data must be interpolated from a regular or triangular grid onto the ICON target grid. We will describe two options to perform this task, the Climate Data Operators (CDO) and the `iconremap` utility from the DWD ICON Tools. The following sections on remapping with CDO and `iconremap` are concluded by some remarks on particular data fields.

Remapping Initial Data using CDO

The Climate Data Operators (CDO, Schulzweida, 2024) are a collection of command-line operators to manipulate and analyze NetCDF and GRIB data. A short overview including references is provided in Section 10.1.2.

Table 2.3 provides a non-exhaustive list of horizontal remapping methods offered by CDO. The following examples make use of first order conservative remapping. The usage of other methods follows the same syntax, allowing for a simple adaption of the commands to the remapping task at hand.

Table 2.3.: Overview of CDO remapping methods (non-exhaustive).

Command	Description
<code>remapnn</code>	Nearest neighbor remapping.
<code>gennn</code>	Generate nearest neighbor interpolation weights.
<code>remapcon</code>	First order conservative remapping.
<code>gencon</code>	Generate first order conservative interpolation weights.
<code>remapcon2</code>	Second order conservative remapping.
<code>gencon2</code>	Generate second order conservative interpolation weights.
<code>remap</code>	Remapping using precalculated weights.

Remapping data fields from a source grid to a target grid can be done with one single command:

```
cdo remapcon,localgrid.nc:N1 -selname,FIELDS \
-setgrid,ingrid.nc:N2 data-file.nc out-file.nc
```

Where the following input has to be specified by the user:

<code>localgrid.nc</code>	Target grid to which the data is remapped
<code>N1</code>	Position of the target sub-grid in <code>localgrid.nc</code>
<code>ingrid.nc</code>	Source grid from which the data is remapped
<code>N2</code>	Position of the source sub-grid in <code>ingrid.nc</code>
<code>data-file.nc</code>	Source data file with data corresponding to <code>ingrid.nc</code>

out-file.nc Output of remapped data corresponding to *localgrid.nc*
FIELDS Short names of fields to be remapped.

Note that there are several ways to define a grid in CDO, e.g. by using predefined shortcuts, text representation, or a data file. For details see [Chapter 1.5](#) of the CDO documentation ([Schulzweida, 2024](#)).

For many applications, the explicit specification of *FIELDS* is not necessary. By default, CDO interpolates all fields in the data set to the target grid. ICON will simply ignore surplus fields that are not needed to initialize the model. If, however, a user wants to choose different remapping methods for different fields, this can, for example, be realized by consecutive CDO commands for each remapping method with distinct comma-separated lists of *FIELDS*. Afterwards, the resulting individual NetCDF files can be merged, for example by using *cdo merge*.

Choosing the correct sub-grid ICON grid files contain sub-grids for cells, edges and vertices. The correct sub-grid is not detected automatically by CDO. Thus the sub-grid needs to be specified explicitly. The output of *cdo sinfov localgrid.nc* contains a paragraph similar to the following:

```
...
Grid coordinates :
  1 : unstructured      : points=4827  nvertex=6
                        grid : number=101  position=0
                        vlon : -0.07922569 to 0.3825339 radian
                        vlat : 0.7070242 to 1.037639 radian
                        available : cellbounds
                        uuid : 7f1e5eb3-fc61-a692-b99f-830dab7e2440
  2 : unstructured      : points=9376  nvertex=3
                        grid : number=101  position=0
                        clon : -0.07495915 to 0.3791283 radian
                        clat : 0.7087018 to 1.035884 radian
                        available : cellbounds
                        uuid : 7f1e5eb3-fc61-a692-b99f-830dab7e2440
  3 : unstructured      : points=4827
...

```

This allows to identify the sub-grid for cell-based variables as number 2 since it uses *clon* and *clat* (see [Section 2.1.1](#)). CDO by default makes use of grid 1, so the sub-grid needs to be specified explicitly.

With *cdo gencon* (see [Table 2.3](#)), CDO can precalculate the weightings for source to target grid interpolation. This can speed up the interpolation process in case of multiple files. Since this is especially interesting for the remapping of several dates of lateral boundary data, this will be described further in [Section 2.3](#).



Using CDO for GRIB2 data: In general, CDO is capable of handling the GRIB2 data format similar to the NetCDF data format. For decoding of GRIB2 data, CDO makes use of the *ecCodes* library. Thus, for a consistent interpretation of field short names, the `ECCODES_DEFINITION_PATH` needs to be set correctly (see Section 1.2.4). Adding `-f nc` to the remap command, converts the GRIB2 input to NetCDF output. Caution is advised regarding the correctness and completeness of metadata when converting from GRIB2 to NetCDF.

Remapping Initial Data using DWD ICON tools

This section describes remapping initial data to your target grid using the `iconremap` utility from the DWD ICON Tools in batch mode.

A typical namelist for processing initial data has the following structure:

```
&remap_nml
  in_grid_filename = INPUT_GRID_FILENAME
  in_filename      = INPUT_FILENAME_GRB
  in_type          = 1                      ! 1: regular grid, 2: ICON grid
  out_grid_filename = ICON_GRIDFILE
  out_filename     = OUTPUT_FILENAME_NC
  out_type         = 2                      ! ICON grid
  out_filetype     = 4                      ! NetCDF format
/

! DEFINITIONS FOR INPUT DATA
!
&input_field_nml ! temperature
  inputname      = "T"
  outputname     = "T"
/
&input_field_nml ! horiz. wind comp. U
  inputname      = "U"
  outputname     = "U"
/
&input_field_nml ! horiz. wind comp. V
  inputname      = "V"
  outputname     = "V"
/
...
```

For each of the variables to be remapped, the script must contain a namelist `input_field_nml` which specifies details of the interpolation methods and the output name. In this example, the 3D temperature field `T` and the horizontal wind components `U`, `V` are remapped from a regular grid onto a triangular ICON grid³. Variables are usually

³Note that in this example, the GRIB2 input *data* file also contains the specification of the input *grid*. Therefore the two namelist parameters `INPUT_GRID_FILENAME` and `INPUT_FILENAME_GRB` are identical!

accessed through their name (character string), but note that for GRIB1 input data the correct field parameter must be provided with the namelist parameter `code`.



Multiple time steps in data file: The remap tool can process files only if they contain a single time step. Furthermore, the tool requires that GRIB records corresponding to a particular variable are stored in contiguous sections. Third, the remapping process fails, if GRIB records are not ordered with respect to levels.



Submitting jobs to the DWD system: ICONREMAP binaries that are pre-compiled for use on the `rc1.dwd.de` should be submitted without MPI. If the subdomain is large, the request may have to be sent to `rc_big` with increased `memsz` setting.

An alternative pre-compiled binary for remapping (but not subsetting) with MPI is available here:

```
/hpc/rhome/for0adm/nwp/x86_vh/util/bin/iconremap
```

This binary should be submitted with `mpiexec -vh -np 4`, to the `sx_norm` queue.

A detailed documentation of the ICON remap command-line options and namelist parameters can be found under `dwd_icon_tools/doc/icontools_doc.pdf`, i.e. [Prill \(2020\)](#). If the DWD ICON tools fail and if the cause of the error does not become clear from the error message, you may increase the output verbosity by setting the command-line options `-v`, `-vv`, `-vvv` etc.

For both, DWD analysis data and ECMWF IFS data, the DWD ICON Tools contain example scripts which generate the required namelists (i.e. `remap_nml` and `input_field_nml`). These scripts are

DWD initial data:

```
dwd_icon_tools/example/runscripts/create_ic_dwd2icon
```

IFS initial data:

```
dwd_icon_tools/example/runscripts/create_ic_ifs2icon
```

The scripts contain a machine dependent batch system header and job launch command which must be adapted to the respective target platform.

Masking of surface fields When remapping, it is possible to make use of the land sea mask information of the source grid (`var_in_mask="FR_LAND"` in `input_field_nml`) in order to mask out specific points. This can be particularly useful when remapping surface fields. In the example below we mask out water points so that only land points contribute to the interpolation stencil for soil moisture.

```

&input_field_nml                ! soil moisture index layer 1
  inputname      = "SWVL1"
  outputname     = "SMIL1"
  var_in_mask    = "FR_LAND" ! field to be used for masking
  in_mask_threshold = 0.5    ! threshold for masking values of input grid
  in_mask_below  = .TRUE.   ! values <= mask_theshold are masked on input grid
/

```

This feature, however, should be used with caution, as it can lead to uninitialized points (missing values) on the target grid. It might happen that isolated land points on the target grid, e.g. small islands, do not have a counterpart on the source grid, which then leads to a zero-sized interpolation stencil. In general, masking and the occurrence of missing values in `iconremap` is based on four mechanisms:

1. First, input data may contain missing values already from the beginning. These are removed for (cell-based) data fields from the interpolation stencils, e. g., of the RBF interpolation. Of course, an interpolation stencil can not only shrink by this process, but even disappear, especially the trivial 1-point stencil of the nearest-neighbor interpolation. As mentioned above, a missing value is also created at the target point in this case.
2. Second, the user can specify masking of the input data, based on a separate data field. There are several namelist parameters for this purpose:
The option `var_in_mask` (`input_field_nml`) specifies the name of a 2D variable in the input file. This field is then evaluated cell by cell according to a threshold criterion specified by the threshold `in_mask_threshold` (`input_field_nml`) and the LOGICAL flag `in_mask_below` (`input_field_nml`).
3. Third, masking can also be specified for the output grid, for example to set missing values on water points that did not exist in the source grid. For this purpose there are the following namelist parameters:
 - The `out_mask_filename` (`remap_nml`) namelist parameter allows to read a separate file containing the land-sea mask.
 - Similar to `var_in_mask`, a 2D field `var_out_mask` (`input_field_nml`) is evaluated cell-by-cell according to a threshold criterion specified by the threshold `out_mask_threshold` (`input_field_nml`) and the LOGICAL flag `out_mask_below` (`input_field_nml`).
4. A final, special type of masking occurs when the `iconremap` target area is not contained in the (regionally bounded) source grid, but extends beyond its boundaries. Here there is a switch `linside_domain_test` (`remap_nml`) test, which performs an appropriate test.

Remarks on particular data fields

Wind fields. For the wind fields, the standard remapping would interpolate the samples from each component U, V separately. This approach has been chosen in the examples

above. However, the standard method completely decouples the components of the vector fields. It does not take into account the fact that it is a vector-field tangent to the sphere.

Therefore the ICON Tools are also capable of interpolating the edge-normal wind components v_n . See the ICON Tools namelist documentation regarding the interpolation $U, V \leftrightarrow v_n$. A special namelist parameter (RBF shape parameter, see Section 7.1.2) must be set for this vector field interpolation with radial basis functions.

Soil Moisture Fields SMI1, SMI2, SMI3, SMI4. In accordance with the remark in Section 2.2.2, care must be taken to properly rename the fields $SWVLx$ to $SMIx$ in the case that "external" IFS analysis files are remapped.

With CDO, a variable can be renamed using the command

```
cdo chname,in-field,out-field in-file out-file
```

where *in-field* and *out-field* are the input and output shortnames of the field to be renamed.

For iconremap, an example namelist `input_field_nml` is given below:

```
&input_field_nml          ! soil moisture index layer 1
  inputname               = "SWVL1"
  outputname              = "SMIL1"
/
```

Soil water content W_SO. The soil water content W_SO is the prognostic soil moisture variable of ICON. ICON is able to read in either W_SO or SMI, with the latter being converted automatically to W_SO during the initialization phase.

If soil moisture fields need to be remapped, it is strongly recommended to remap SMI instead of W_SO . Remapping of W_SO might lead to nonphysical soil water contents, which is related to the fact that the soil types of the source and target grid points might well differ.

If SMI is unavailable¹ in the initial data, it can be diagnosed from W_SO prior to the remapping step with the help of a small Fortran program named `smi_w_so.f90`. It ships with the ICON code and can be found in the subdirectory `icon/scripts/postprocessing/tools`. Note that `smi_w_so.f90` requires the soil type field `SOILTYP` as additional input. It can be extracted from the external parameter file matching the source grid (see Section 2.4) and must then be concatenated with the file containing W_SO .

¹Starting from 2018-03-14 (2018-07-11), DWD's operational forecast products contain the soil moisture index SMI on the EU domain (global domain) (`vv=0` output, initialized analysis).

2.3. Boundary Data Preparation for ICON-LAM

When running ICON in limited area mode (LAM), lateral boundary conditions must be provided. In real case applications these are time dependent and must be updated periodically by reading input files. To this end, forecast or analysis data sets from a *driving model* may be used which, however, need to be interpolated horizontally to the ICON grid first.

In this section we briefly describe the process of generating these lateral boundary conditions (LBCs). Again, the basic pre-processing steps for ICON are visualized in Figure 2.10, where the additional pre-processing of boundary data constitutes the main difference to Figure 2.7.

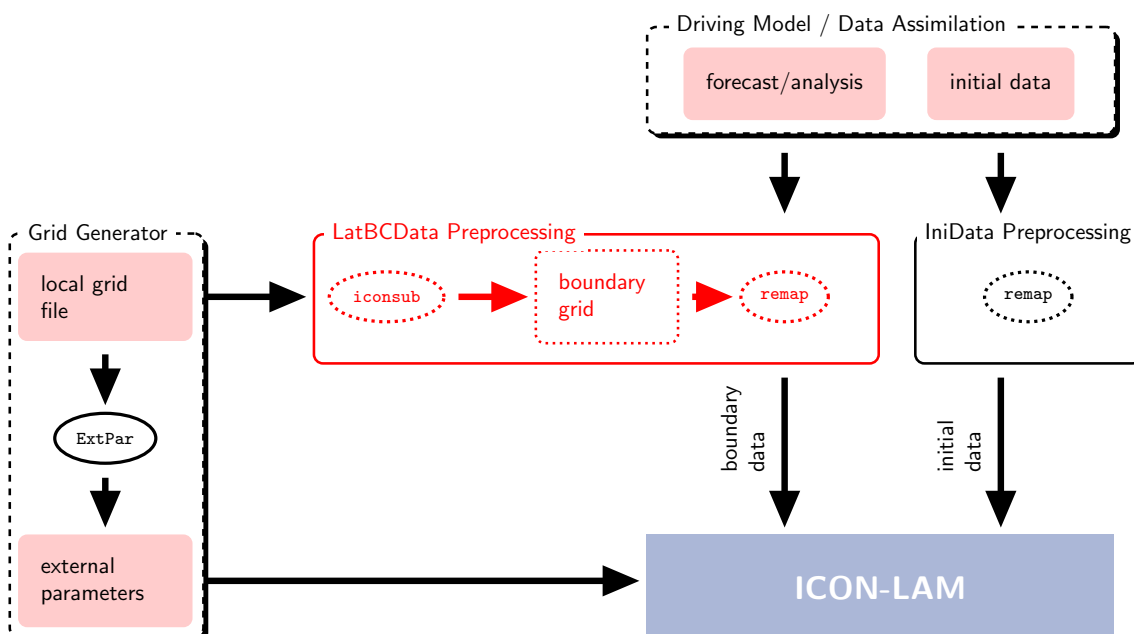


Figure 2.10.: Basic pre-processing steps for ICON-LAM (compare to Fig. 2.7). The grid generation process and the external parameters are described in the Sections 2.1 and 2.4. The initial data processing is covered by Section 2.2.3. Finally, the lateral boundary data (LatBCData) pre-processing, which extracts the boundary data, is described in Section 2.3. This pre-processing step is necessary for the limited area mode ICON-LAM.

Boundary Data Retrieval

The raw data files which are intended to be used as LBCs must contain one of the sets of variables depicted in Figure 2.11, on either a triangular ICON grid, or a regular latitude-longitude grid.

The fact that different sets of variables can be used provides some flexibility in terms of the driving model. As indicated in Figure 2.11, sets I to III are typical for ICON, COSMO and IFS, respectively.

Set I (e.g. ICON)										
$\left\{ \begin{array}{c} U, V \\ \text{or} \\ VN \end{array} \right\}$	W,	THETA_V,	DEN,	QV,	QC,	QI,	QR,	QS,	HHL	
Set II (e.g. COSMO)										
U,	V,	W,	T,	P,	QV,	QC,	QI,	QR,	QS,	HHL
Set III (e.g. IFS)										
U,	V,	OMEGA,	T,	LNSP,	QV,	QC,	QI,	QR,	QS,	FI

Figure 2.11.: Sets of variables that may serve as lateral boundary conditions for ICON-LAM, with examples of driving models given in brackets. Optional fields are marked in gray. Blending of the sets is not allowed. In case of IFS please see Table 2.2 for the requested internal ICON names to which these GRIB2 short names must be mapped. LNSP denotes the logarithm of the surface pressure, and FI is the surface geopotential.



Important note: The 3D field HHL (geometric height of model half levels above mean sea level) is constant data. It needs only be contained in the raw data file whose validity date matches the model start date. In case of set III (IFS) the field HHL is computed by ICON during read-in.

In situations where the user forgets to provide the HHL field, the ICON model aborts with a non-intuitive error. ICON complains that geopotential is missing, when in fact the HHL variable is missing. The details for this behavior are explained in Section 6.4: ICON follows a “decision tree” to deduce the contents of the input data. When HHL is missing in the data set, the model initialization assumes a pressure based coordinate system.

Similar to the retrieval of initial conditions in Section 2.2.1, it is also possible to download lateral boundary data from the DWD database.

Lateral boundary conditions from deterministic ICON forecasts

The following PAMORE command retrieves lateral boundary conditions from DWD’s *forecast* database category:

```
pamore -d date -hstart hh -hstop hh -hinc hh -model xx -ilam_boundary
```

Use `-model iglo`, for retrieving global 13 km forecast data, `-model ieu` for 6.5 km forecast data on the ICON-EU domain, and `-model ilam` for 2.1 km forecast data on the ICON-D2 domain. The meaning of the remaining command line arguments is as follows:

- `-d date`
specifies the start date in the format *YYYYMMddhh*

- `-hstart hh -hstop hh`
specifies the requested time range in hours
- `-hinc hh`
specifies the temporal resolution (increments) in hours
- `-ilam_boundary`
enables the variable set II (see Fig. 2.11) for lateral boundary conditions.

Example: The following example retrieves lateral boundary conditions from the ICON-EU domain for a time range of 36 hours and a temporal resolution of 2 hours, starting at 2024032212:

```
pamore -d 2024032212 -hstart 0 -hstop 36 -hinc 2 -model ieu -ilam_boundary
```



Important note: Please note that ICON forecast data have an expiration date. They are deleted from DWD's data base after 18 months. If you are interested in lateral boundary data which date back longer than 18 months, please use the following PAMORE command:

Lateral boundary conditions from ICON's assimilation cycle

The following PAMORE command retrieves lateral boundary conditions from DWD's *assimilation* database category:

```
pamore -d date -hstart hh -hstop hh -model xx -hindcast_ilam
```

Use `-model iglo`, for retrieving global 13 km forecast data, `-model ieu` for 6.5 km forecast data on the ICON-EU domain, and `-model ilam` for 2.1 km forecast data on the ICON-D2 domain. Similar to the previous PAMORE command, the parameter `-hindcast_ilam` enables set II (see Fig. 2.11) for lateral boundary conditions. It extracts the data from the assimilation database category, for which there exists no expiration date. The command line argument `-hinc` is not applicable in this case. The temporal resolution is pre-set to 1 hour.

Example: The following example retrieves lateral boundary conditions from the global domain for a time range of 48 hours and a temporal resolution of 1 hour, starting at 2017092200:

```
pamore -d 2017092200 -hstart 0 -hstop 48 -model iglo -hindcast_ilam
```

In particular the above example extracts the 1 h, 2 h, 3 h forecast data from consecutive first guess runs (which are launched every 3 hours in the global assimilation cycle), starting at 2017092200. The final product consists of hourly lateral boundary data spanning the time range [2017092200, 2017092400].

This PAMORE command is particularly useful, if the user wants to perform so called hindcast experiments.

Remapping Lateral Boundary Data Using CDO

The remapping of lateral boundary data to a target grid using CDO is basically identical to the remapping of initial data which is described in Section 2.2.3. This task can be done by the following command:

```
cdo remapcon,localgrid.nc:N1 \
  -setgrid,ingrid.nc:N2 data-file.nc out-file.nc
```

Please note the remarks regarding GRIB2 data format, grid description and remapping methods provided in Section 2.2.3.

The computationally expensive part of the remapping process is the calculation of the weighting factors. For multiple files to be remapped, these factors can be precalculated, for example by using `cdo gencon`. This can be realized as follows:

```
# Generate weightings
cdo -P 4 gencon,localgrid.nc:N1 -selgrid,N2
  ingrid.nc weightings.nc

for data-file in data-files
do
  cdo remap,localgrid.nc:N1,weightings.nc
    data-file out-file.nc
done
```

In the previous command, the list of fields to be remapped *FIELDS* was not specified explicitly for brevity. Furthermore, please note that using precalculated weightings does not work with missing values (see Section 10.4.1).



Using an auxiliary boundary grid: As described in the following section, lateral boundary data can be provided on an auxiliary grid containing boundary cells only, which can avoid I/O bottlenecks. Although it is currently not possible to create such an auxiliary grid with CDO, it is possible to use CDO to remap data to an existing auxiliary boundary grid. Practically, there is no difference in the above commands except for the choice of the file `localgrid.nc`.

Remapping Lateral Boundary Data Using DWD ICON tools

As an alternative to CDO, the remapping of lateral boundary data can also be performed using the DWD ICON tools (Section 1.4). Here we describe the individual steps of the interpolation onto the boundary zone of a limited area grid. However, it can also be found summarized in an example script of the DWD ICON tools, namely the run script `create_lbc_dwd2icon` in the directory `dwd_icon_tools/icontools`, which processes a whole directory of raw data files (hereafter referred to as the script variable “DATADIR”). The `create_lbc_dwd2icon` script can be submitted to the PBS batch system of DWD’s

NEC SX-Aurora. In order to run it on other machines, the batch system header and mpirun command must be adapted accordingly.

The output files are written to the directory specified in the variable `OUTDIR`. The input files are read from `DATADIR`, therefore this directory should not contain other files and should not be identical to the output folder.

The second necessary input besides the directory names `DATADIR` and `OUTDIR` are the grid file names. For the mapping procedure that is described in the following two sections, these are specified by

```
INGRID="input_grid_file" # file name of input grid
LOCALGRID="grid_file.nc" # file name of limited area (output) grid
```

Step 1: Extract Boundary Region from the Local Grid File

In the first pre-processing step for ICON-LAM we create an auxiliary grid file which contains only the cells of the boundary zone. This step needs to be performed only once before generating the boundary data.



Important note: Note that this step is not allowed if vertical boundary nudging is used in addition to lateral boundary nudging. This corresponds to the namelist parameter setting `nudge_type=1` (namelist `nudging_nml`). In this case, boundary data must be provided for the entire local (limited area) grid, rather than for a boundary strip only, see Section 6.2.

We use the `iconsub` program from the collection of ICON Tools, see Section 1.4, with the following namelist:

```
&iconsub_nml
  grid_filename    = "${LOCALGRID}",
  output_type      = 4,
  lwrite_grid      = .TRUE.,
/
&subarea_nml
  ORDER            = "${OUTDIR}/grid_file_lbc.nc",
  grf_info_file    = "${LOCALGRID}",
  min_refin_c_ctrl = 1
  max_refin_c_ctrl = 14
/
```

Running the `iconsub` tool creates a grid file `grid_file_lbc.nc` for the boundary strip. The cells in this boundary zone are identified by their index in a special meta-data field, the `refin_c_ctrl` index, e.g. `refin_c_ctrl = 1, ..., 14`, see Figure 2.12.

The width of the extracted boundary strip in terms of cell rows is specified by the namelist parameters `min_refin_c_ctrl` and `max_refin_c_ctrl`. The maximum allowed value for `max_refin_c_ctrl` is given by `max(refin_c_ctrl) == bdy_indexing_depth`, i.e. the

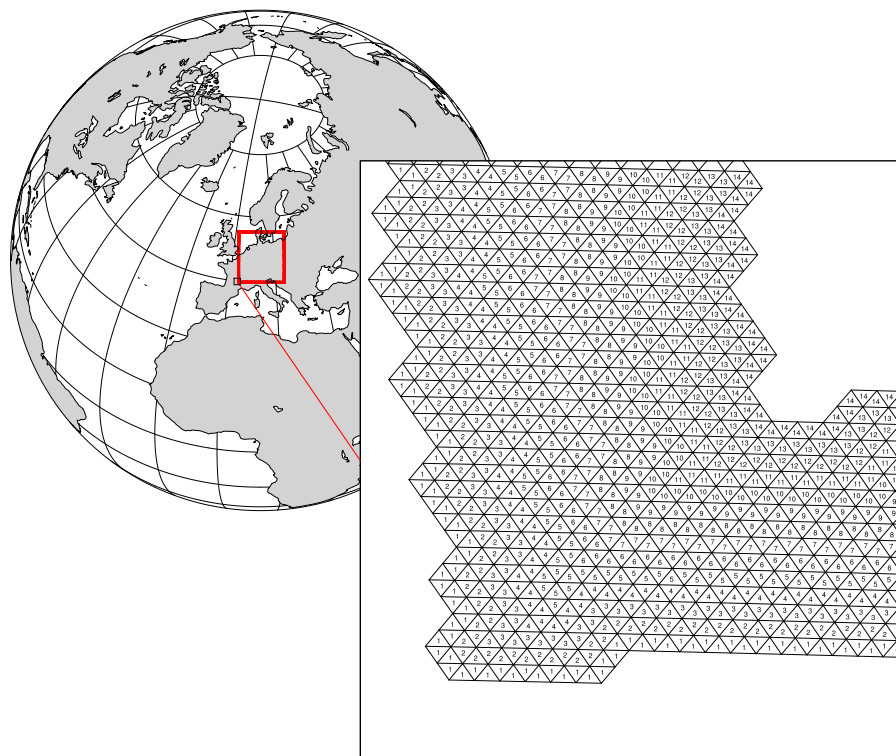


Figure 2.12.: Illustration of the ICON-LAM boundary zone. The cells are identified by their `refin_c_ctrl` index, e.g. `refin_c_ctrl = 1, ..., 14`.

boundary indexing depth that has been chosen when generating the limited area grid (see Section 2.1.5). The width of the extracted boundary strip must be equal or larger than the width of the lateral nudging zone in the limited area run for which it is foreseen. A safe setting would be `max_refin_c_ctrl = max(refin_c_ctrl)`, as used in this example.



Reference to the original grid: For a given boundary region file `grid_file_lbc.nc` the question may arise, which original grid was used for its creation. This is analogous to the explanation in Section 2.1.7: The boundary region files contain a link to the original grid file in the form of a unique fingerprint `uuidOfOriginalHGrid`.

Step 2: Creating Boundary Data

Any of the data sources explained in the Sections 2.2.1 and 2.2.2 can be chosen for the extraction of boundary data. To be more precise, boundary data originating from ICON, IFS, and COSMO have successfully been used. Data sets from other global or regional models may work as well, but have not been tested yet.

We define the following namelist for the `iconremap` program from the collection of ICON Tools. This happens automatically in our ICON-LAM pre-processing script:

```
&remap_nml
  in_grid_filename = "${INGRID}"
  in_filename      = "input_data_file"
  in_type          = 2
  out_grid_filename = "${OUTDIR}/grid_file_lbc.nc"
  out_filename     = "${OUTDIR}/data_file_lbc.nc"
  out_type         = 2
  out_filetype     = 4
/
```

For the usual case that the directory `${DATADIR}` contains several input files at once, they must be processed one after the other. So for the parameter `in_filename` the name of each raw data file is used successively. With respect to the output filename `data_file_lbc.nc` it is a good idea to follow a consistent naming convention. See Section 6.4.1 on the corresponding namelist setup of the ICON model.

The parameters `in_type=2` and `out_type=2` specify that both grids correspond to triangular ICON meshes (`in_grid_filename` and `out_grid_filename`). Additionally, a namelist `input_field_nml` is appended for each of the pre-processed variables.

Note that the `input_data_file` must contain only a single time step when running the `iconremap` tool. The `iconremap` tool therefore must be executed repeatedly in order to process the whole list of boundary data samples.

After the specification of the filenames, the remapping parameters for all variables of the Set I in Fig. 2.11 are defined. Regarding the HHL field an additional remark is in order: Since this variable needs only be contained in the raw data file whose validity date matches the envisaged model start date (see the remark in Section 2.3), we set this field as optional:

```
&input_field_nml
  inputname      = "HHL"
  outputname     = "z_ifc"
  intp_method    = 3
  loptional      = .TRUE.
/
```

The fully assembled namelist can then be used to call the `iconremap` interpolation tool.

In this context the following technical detail may considerably speed up the pre-processing: The `iconremap` tool allows to store and load interpolation weights to and from a NetCDF file. When setting the namelist parameter `ncstorage_file` (character string) in the `iconremap` namelist `remap_nml`, the remapping weights are loaded from a file with this name. If this file does not exist, the weights are created from scratch and then stored for later use. Note that for MPI-parallel runs of the `iconremap` tool multiple files are created. Re-runs require exactly the same number of processes.

2.4. External Parameter Files

External parameter fields describe properties of the Earth’s surface and atmosphere, which can be assumed to be invariant during the course of a typical NWP forecast (i. e. a couple of days). Examples are the topography, the land-sea mask, the soil type and atmospheric aerosols. Most of the fields are constant in time while some are available on a monthly basis in order to represent the seasonal cycle. They are read by the model during startup.

The generation is based on a set of raw datafields which are listed in Table 2.4. The full list of external parameter fields is given in Table 2.5.

Table 2.4.: Raw datasets from which the ICON external parameter fields are derived.

Dataset	Source	Resolution
GLOBE	NGDC	30"
MERIT	IIS	3"
(limited domain: 90 N - 60 S)		
REMA	PGC	200 m
(limited domain: 62 S - 90 S)		
ASTER	METI/NASA	1"
(limited domain: 60 N - 60 S)		
GlobCover 2009	ESA	10"
DSMW	FAO	5'
SeaWIFS NDVI Climatotology	NASA/GSFC	2.5'
CRU-CL	CRU-UEA	0.5°
GACP Aerosol Optical thickness	NASA/GISS	4x5°
GLDB	DWD/RSRU/MeteoFrance	30"
MODIS albedo	NASA	3'

Abbreviations (raw datasets): ASTER Advanced Spaceborne Thermal Emission and Reflection Radiometer Global Digital Elevation Model; CRU-CL Climate Research Unit – Gridded climatology of 1961-1990 monthly means; CRU-UEA Climate Research Unit – University of East Anglia; DSMW Digital Soil Map of the World; ESA European Space Agency; FAO Food and Agricultural Organization; GACP Global Aerosol Climatology Project; GLDB Global Lake Database; GlobCover 2009 Global Land Cover Map for 2009; GLOBE Global Land One-km Base Elevation Project; GSFC Goddard Space Flight Center; IIS Institute of Industrial Sciences, The University of Tokyo; MERIT Multi-Error-Removed Improved-Terrain (MERIT) DEM; METI Ministry of Economy, Trade, and Industry; MODIS Moderate Resolution Imaging Spectroradiometer; NASA National Aeronautics and Space Administration; NGDC NOAA National Geophysical Data Center; PGC U.S. Polar Geospatial Center; REMA The Reference Elevation Model of Antarctica; SeaWIFS Sea-viewing Wide Field-of-view Sensor.

2.4.1. ExtPar Software

The ExtPar software (ExtPar – External Parameters for numerical weather prediction and climate application) is able to generate external parameters for the different models

GME, COSMO, HRM and ICON. Experienced users can run ExtPar on UNIX or Linux systems to transform raw data from various sources into domain-specific data files. For ICON, ExtPar will output the fields given in Table 2.5 in the NetCDF file format and GRIB2 on the native triangular grid. For a more detailed overview of ExtPar, the reader is referred to the *User and Implementation Guide* of ExtPar, [Asensio and Messmer \(2014\)](#), and, additionally [Smiatek et al. \(2008, 2016\)](#).

The ExtPar pre-processor is a COSMO software and not part of the ICON training course release. Still, the ExtPar tool can be accessed via the Zonda web service (see Section 2.1.6). Similar as for the grid files, for fixed domain sizes and resolutions some external parameter files for the ICON model are available for download via

<http://icon-downloads.mpimet.mpg.de>

The generation of the data represents a resource-intensive process. At the moment, it is not possible to generate the Extpar file for R2B10 grid at DWD on the rc1.dwd.de.

2.4.2. Topography Information

Among various other fields, the external parameter files provide topography information, see Table 2.5. The HSURF dataset contains the geometric height of the earth's surface above sea level (unit: m), where the raw data of the terrain model used (GLOBE, ASTER) is aggregated over the grid box/triangle and the aggregated value is assigned to the triangle center point. Therefore, HSURF is *not* identical to the specific mean sea level height of the point lying under the center of the triangle or the maximum altitude of the area lying under the triangle (as e.g. in aeronautical charts).

Besides, please note the following remark: The topography contained in the ExtPar data files is in general *not identical* to the topography data which is used by the model. This is because at start-up, after reading the ExtPar data, the topography field is optionally filtered by a smoothing operator (`n_iter_smooth_topo > 0` in `extpar_nml`). Therefore, for post-processing purposes it is necessary to use the topography height `topography_c` (GRIB2 short name HSURF) from the model output (cf. Section 7 and Appendix A). The same applies to the fields `DEPTH_LK`, `FR_LAND`, `FR_LAKE`, and `Z0`, which are unconditionally modified by ICON.

2.4.3. Additional Information for Surface Tiles

ExtPar data files are available for download with and without additional information for surface tiles. See Section 3.8.9 for details on the tile approach.

ExtPar files suitable for the tile approach are indicated by the suffix `_tiles`. They are also applicable when running the model without tiles. ExtPar files without the suffix “`_tiles`”, however, must only be used when running the model without tiles (`ntiles = 1`, namelist `1nd_nml`).

The data files do not differ in the number or type of fields, but rather in the way some fields are defined near coastal regions. Without the `_tiles` suffix, various surface parameters

(e.g. `SOILTYP`, `NDVI_MAX`) are only defined at so-called dominant land points, i.e. at grid elements where the land fraction exceeds 50%. With the `_tiles` suffix, however, these parameters are additionally defined at cells where the land fraction is below 50%. By this, we allow for mixed water-land points. The same holds for the lake depth (`DEPTH_LK`) which is required by the lake parameterization scheme `FLake`. For files without the `_tiles` suffix, `DEPTH_LK` is only defined at dominant lake points.

2.4.4. Parameter Files for Radiation

In addition to the `ExtPar` fields, input fields for radiation are loaded into the `ICON` model. These constants fields are distributed together with the model code.

ecRad

Input files for the `ecRad` radiation scheme are located in the folder `icon/externals/ecrad/data`. The correct folder path must be passed to `ICON` via the `ICON` namelist parameter `ecrad_data_path` in the namelist `radiation_nml`.

RRTM

Input files for the `RRTM` radiation scheme are located in the folder `icon/data`. The `RRTM` scheme is `ICON`'s old implementation variant for radiation and is rarely used anymore.

rrtmg_lw.nc

parameters for radiative transfer calculation used for the underlying `RRTMG` algorithm, thermal radiation.

ECHAM6_CldOptProps.nc

Cloud optical properties for liquid clouds at 30 wavelengths used for the underlying `RRTMG` algorithm.

On default, `ICON` expects the `RRTMG` parameter files to be named as above. Renaming is possible, however the modified name must then be passed into `ICON` via the namelist parameters `lrtm_filename` and `cldopt_filename` in the namelist `nwp_phy_nml`.

Table 2.5.: External parameter fields which are requested by ICON during startup (in alphabetical order). Fields marked in blue are not read by ICON in operational NWP runs. In general they are only requested, if the respective depicted namelist parameter is set.

shortName	Description	Raw dataset
AER_SS12	Sea salt aerosol climatology (monthly mean) irad_aero=6,9 (namelist radiation_nml)	GACP
AER_DUST12	Total soil dust aerosol climatology (monthly mean) irad_aero=6,9 (namelist radiation_nml)	GACP
AER_ORG12	Organic aerosol climatology (monthly mean) irad_aero=6,9 (namelist radiation_nml)	GACP
AER_SO412	Total sulfate aerosol climatology (monthly mean) irad_aero=6,9 (namelist radiation_nml)	GACP
AER_BC12	Black carbon aerosol climatology (monthly mean) irad_aero=6,9 (namelist radiation_nml)	GACP
ALB_DIF12	Shortwave (0.3 – 5.0 μm) albedo for diffuse radiation (monthly mean) albedo_type=2 (namelist radiation_nml)	MODIS
ALB_UV12	UV-visible (0.3 – 0.7 μm) albedo for diffuse radiation (monthly mean) albedo_type=2 (namelist radiation_nml)	MODIS
ALB_NI12	Near infrared (0.7 – 5.0 μm) albedo for diffuse radiation (monthly mean) albedo_type=2 (namelist radiation_nml)	MODIS
DEPTH_LK	Lake depth	GLDB
EMIS_RAD	Surface longwave (thermal) emissivity itype_lwemiss=1 (namelist extpar_nml)	GlobCover 2009
EMISS	Surface longwave (thermal) emissivity derived from satellite measurements (monthly mean) itype_lwemiss=2 (namelist extpar_nml)	CAMEL (combined ASTER and MODIS)
FOR_D	Fraction of deciduous forest ntiles=1 (namelist lnd_nml)	GlobCover 2009
FOR_E	Fraction of evergreen forest ntiles=1 (namelist lnd_nml)	GlobCover 2009
FR_ICE	Land glacier fraction Taken from FR_LUC, if missing	
FR_LAKE	Lake fraction (fresh water)	GLDB
FR_LAND	Land fraction (excluding lake fraction but including glacier fraction)	GlobCover 2009
FR_LUC	Land-use class fraction	
HSURF	Topographic height at cell centers	MERIT, REMA, ASTER, GLOBE
LAI_MX	Leaf area index in the vegetation phase ntiles=1 (namelist lnd_nml)	GlobCover 2009
NDVI_MAX	Normalized differential vegetation index	SeaWIFS

Continued on next page

Table 2.5.: *Continued from previous page*

NDVI_MRAT	Proportion of monthly mean NDVI to yearly maximum (monthly mean)	SeaWiFS
PLCOV_MX	Plant covering degree in the vegetation phase ntiles=1 (namelist lnd_nml)	GlobCover 2009
ROOTDP	Root depth ntiles=1 (namelist lnd_nml)	GlobCover 2009
RSMIN	Minimum stomatal resistance ntiles=1 (namelist lnd_nml)	GlobCover 2009
SOILTYP	Soil type	DSMW
SSO_STDH	Standard deviation of sub-grid scale orographic height	MERIT, REMA, ASTER
SSO_THETA	Principal axis-angle of sub-grid scale orography	MERIT, REMA, ASTER
SSO_GAMMA	Horizontal anisotropy of sub-grid scale orography	MERIT, REMA, ASTER
SSO_SIGMA	Average slope of sub-grid scale orography	MERIT, REMA, ASTER
T_2M_CL	Climatological 2m temperature Serves as lower boundary condition for soil model	CRU-CL
T_2M_CLIM	Climatological 2m temperature (monthly mean) itype_vegetation_cycle>1 (namelist extpar_nml)	
TOPO_CLIM	Interpolated topographic height for T_2M_CLIM data itype_vegetation_cycle>1 (namelist extpar_nml)	
T_SEA	Sea surface temperature climatology (monthly mean) sstice_mode=2 (namelist lnd_nml)	
Z0	Surface roughness length (over land), containing a contribution from subgrid-scale orography itype_z0=1 (namelist nwp_phy_nml)	GlobCover 2009

2. Input Data

3. Model Description

Before I came here I was confused about this subject. Having listened to your lecture I am still confused. But on a higher level.

Enrico Fermi

This chapter is devoted to a summary of ICON’s model structure. The principal components are illustrated in Fig. 3.1:

Dynamics	The centerpiece of the numerical weather prediction system is the <i>dynamical core</i> , which integrates the discrete equations for fluid motion forward in time. ICON’s dycore will be shortly described in Sections 3.1–3.5.
Tracer Advection	The dynamical core is followed by the numerical advection scheme, e.g. for humidity and cloud water. Section 3.6 focuses on the different methods available in ICON.
Physics	The former components are then coupled to parameterizations for processes such as convection that occur on scales too small to be resolved directly. We present a comprehensive overview of the physics parameterizations (NWP-mode) in Sections 3.7–3.8.

Finally, the chapter is concluded with the discussion of variable resolution modeling.

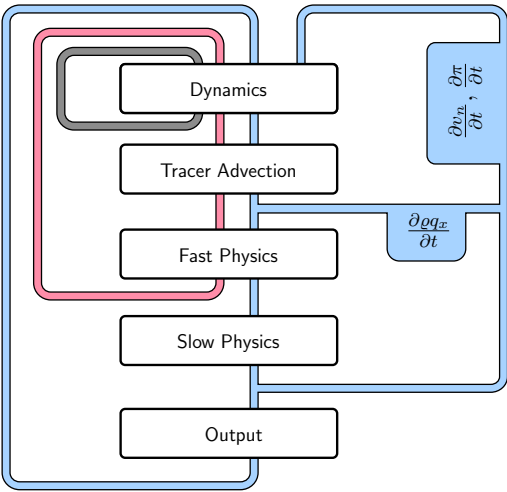


Figure 3.1.: ICON’s model structure. This flow chart will be revisited in detail in Fig. 3.9.

3.1. Governing Equations

The prognostic equation set of the ICON model ([Gassmann and Herzog, 2008](#)) describes a two-component system consisting of dry air and water, where water may occur in all three phases including precipitating drops and ice particles.

As described in [Wacker and Herbert \(2003\)](#), an equation set for the air mixture can be derived by first introducing a reference velocity into the governing equations of each air constituent. The equations for momentum, mass and energy of the mixture, as given below, are then formed as a sum of the constituent-specific equation sets. The specific form of the governing equations for the mixture depends on the choice of the velocity reference frame.

In ICON we have chosen a barycentric reference frame. The corresponding barycentric reference velocity

$$\mathbf{v}_b = \frac{\sum_k \rho_k \mathbf{v}_k}{\sum_k \rho_k}, \quad (3.1)$$

is the mass weighted average of the constituent-specific advective velocities \mathbf{v}_k , with ρ_k denoting the partial density of constituent k . For simplicity, \mathbf{v}_b will be denoted \mathbf{v} in the following.

Any velocity vector in ICON is represented as

$$\mathbf{v} = v_n \mathbf{e}_n + v_t \mathbf{e}_t + w \mathbf{e}_k,$$

where \mathbf{e}_n and \mathbf{e}_t are the unit vectors normal and tangential to the edge of a triangular cell (defined at edge midpoints), and v_n , v_t are the normal and tangential vector components. The unit vectors are chosen in such a way that $\mathbf{e}_t \times \mathbf{e}_n = \mathbf{e}_k$ holds, where \mathbf{e}_k points upwards. In other words, $(\hat{v}_t, \hat{v}_n, \hat{w})$ form a right-handed system.

In order to separate turbulent fluctuations from the mean flow, a density weighted averaging (known as Hesselberg averaging) is applied. Every field $\phi = \hat{\phi} + \phi''$ gets decomposed into a density-weighted mean $\hat{\phi}$ and a deviation ϕ'' , with $\hat{\phi} = \overline{\rho\phi}/\bar{\rho}$, and subsequent averaging ([Hesselberg, 1925](#)). $\bar{\phi}$ denotes the classic Reynolds average. More details on density-weighted average calculus can be found, e. g., in [Zdunkowski and Bott \(2003\)](#).

The Hesselberg-averaged, fully compressible equation system of ICON is presented below. It is based on the common assumption of a spherical geoid and includes the shallow atmosphere approximation. A deep-atmosphere extension without the shallow atmosphere approximation is implemented as well, but is beyond the scope of the tutorial. We refer to [Borchert et al. \(2019\)](#) for details.

The equation set is written in the prognostic variables horizontal edge-normal velocity \hat{v}_n (3.4), vertical velocity \hat{w} (3.5), Exner pressure (3.6)

$$\bar{\pi} = \left(\frac{\bar{p}}{p_{00}} \right)^\kappa, \quad (3.2)$$

where $\kappa = R_d/c_{pd}$ is the Poisson constant, $p_{00} = 1000\text{hPa}$ is a reference pressure and \bar{p} is the total air pressure, total density of the air mixture $\bar{\rho}$ (3.7) with

$$\bar{\rho} = \sum_k \bar{\rho}_k = \bar{\rho} \sum_k \hat{q}_k, \quad (3.3)$$

and partial densities $\bar{\rho}_k$ (3.8), with mass fractions $\hat{q}_k = \bar{\rho}_k/\bar{\rho}$. The index $k \in [d, v, c, r, i, f]$ refers to the constituents dry air, water vapor, cloud water, rain water, cloud ice and frozen precipitating particles (such as snow, graupel, hail, etc.). We emphasize that the density of dry air $\bar{\rho}_d$ is a diagnostic quantity in ICON, which is diagnosed from Eq. (3.3) when needed. If the 2D velocity vector at a cell edge is needed, the velocity component tangent to a cell edge \hat{v}_t is obtained from \hat{v}_n using a Radial Basis Function (RBF) reconstruction, following [Narcowich and Ward \(1994\)](#).

$$\frac{\partial \hat{v}_n}{\partial t} + \frac{\partial \hat{K}_h}{\partial n} + (\hat{\zeta} + f)\hat{v}_t + \hat{w} \frac{\partial \hat{v}_n}{\partial z} = -c_{pd}\hat{\theta}_v \frac{\partial \bar{\pi}}{\partial n} - \frac{1}{\bar{\rho}} (\nabla_h \cdot \overline{\rho \mathbf{v}'' \mathbf{v}''}) \cdot \mathbf{e}_n \quad (3.4)$$

$$\frac{\partial \hat{w}}{\partial t} + \hat{\mathbf{v}}_h \cdot \nabla \hat{w} + \hat{w} \frac{\partial \hat{w}}{\partial z} = -c_{pd}\hat{\theta}_v \frac{\partial \bar{\pi}}{\partial z} - g - \frac{1}{\bar{\rho}} \frac{\partial}{\partial z} \overline{\rho \mathbf{v}'' \mathbf{v}''} \quad (3.5)$$

$$\frac{c_{vd}c_{pd}}{R_d} \bar{\rho} \hat{\theta}_v \frac{\partial \bar{\pi}}{\partial t} + c_{pd} \bar{\pi} \nabla \cdot (\bar{\rho} \hat{\mathbf{v}} \hat{\theta}_v) = c_{pd} \bar{\pi} \bar{\rho} \hat{\theta}_v \chi \nabla \cdot \hat{\mathbf{v}} + c_{pd} \bar{\pi} \bar{\rho} \bar{Q} \quad (3.6)$$

$$\frac{\partial \bar{\rho}}{\partial t} + \nabla \cdot (\bar{\rho} \hat{\mathbf{v}}) = 0 \quad (3.7)$$

$$\frac{\partial \bar{\rho} \hat{q}_k}{\partial t} + \nabla \cdot (\bar{\rho} \hat{q}_k \hat{\mathbf{v}}) = -\nabla \cdot (\bar{J}_k^z \mathbf{k} + \overline{\rho q_k'' \mathbf{v}''}) + \bar{\sigma}_k \quad (3.8)$$

For the definition of additional symbols and variables, we refer to Table 3.1. The corresponding data structures containing the physics and dynamics variables are outlined in Section 9.2.

The equation system is closed by the equation of state and a set of (approximate) boundary conditions for vertical momentum flux $\bar{\rho} \hat{w}$ and vertical diffusion flux

$$\bar{J}_k^z = \bar{\rho}_k (\hat{w}_k - \hat{w}) . \quad (3.9)$$

The latter describes the motion of each constituent k relative to the barycentric reference velocity. It should not be confused with molecular or turbulent diffusion.

The equation of state is formulated in the prognostic variables $\bar{\pi}$ and $\bar{\rho}$

$$\bar{\pi} = \left(\frac{R_d}{p_{00}} \bar{\rho} \hat{\theta}_v \right)^{\frac{R_d}{c_{vd}}} \quad (3.10)$$

and allows for the conversion between Exner pressure $\bar{\pi}$ and virtual potential temperature

$$\hat{\theta}_v = \hat{T}_v \left(\frac{p_{00}}{\bar{p}} \right)^\kappa = \frac{\hat{T}_v}{\bar{\pi}}, \quad (3.11)$$

Table 3.1.: Explanation of symbols in the model equations

Symbol	Description
$\frac{\partial}{\partial n}$	horizontal derivative in edge-normal direction
$\widehat{K}_h = 0.5 (\widehat{v}_n^2 + \widehat{v}_t^2)$	horizontal component of the kinetic energy
$\widehat{\zeta} = \mathbf{k} \cdot (\nabla \times \widehat{\mathbf{v}})$	vertical component of relative vorticity
$f = 2\Omega \sin \phi$	Coriolis parameter
c_p, c_v	specific heat capacity of air mixture at constant pressure/volume
c_{pd}, c_{vd}	specific heat capacity for dry air at constant pressure/volume
$R_d = c_{pd} - c_{vd}$	Gas constant for dry air
R_v	Gas constant for water vapor
$\chi = 1 - (c_p/c_{pd})(c_{vd}/c_v)$	thermodynamic constant, dependent on fluid composition
g	acceleration of gravity
\overline{Q}	adiabatic heat source, including the turbulent heat flux (Gassmann and Herzog, 2008)
$\overline{\sigma}_k$	internal conversion rate for k th constituent (i.e. conversion among different phases or particle forms)
$\overline{\rho q_k'' v''}$	turbulent flux of k th partial mass fraction
$\overline{E}_v = \overline{\rho q_v'' v''} _s$	surface evaporation flux
$\overline{S}_k = \overline{\rho \widehat{q}_k \widehat{v}_k^T}$	sedimentation flux of k th constituent
\widehat{v}_k^T	terminal fall velocity of k th constituent

where \widehat{T}_v denotes the virtual temperature

$$\widehat{T}_v = \widehat{T}(1 + \alpha),$$

with

$$\alpha = \left(\frac{R_v}{R_d} - 1 \right) \widehat{q}_v - \sum_{k \neq v, d} \widehat{q}_k.$$

The actual method for calculating updated values of $\widehat{\theta}_v$ in ICON is detailed in Section 3.1.1.

The (approximate) lower boundary condition for $\overline{\rho \widehat{w}}$ and \overline{J}_k^z reads:

$$\overline{\rho \widehat{w}}|_s = 0 \tag{3.12}$$

$$\overline{J}_k^z|_s = \begin{cases} \overline{E}_v, & \text{if } k = v \\ \sum_{k=k_{prec}} \overline{S}_k - \overline{E}_v, & \text{if } k = d \\ 0, & \text{if } k \in \{c, i\} \\ -\overline{S}_k|_s, & \text{if } k \in k_{prec} = \{r, f\} \end{cases} \tag{3.13}$$

In particular, the lower boundary is impermeable w.r.t. to total mass. A short discussion on the consequences of this boundary condition will be given in Section 3.2.

From the prognostic equations (3.7), (3.8) for total density and partial density some important constraints can be derived, which must hold in the discretized analogue in order to achieve mass conservation. First of all, from the definition of total density (3.3) it follows that

$$\sum_k \hat{q}_k = 1 . \quad (3.14)$$

Furthermore, as the sum of the continuity equations for partial densities (3.8) over all k must yield the continuity equation for total density (3.7), the following constraints hold:

$$\sum_k \bar{\sigma}_k = 0 \quad (3.15)$$

$$\sum_k \bar{J}_k^z = 0 \quad (3.16)$$

$$\sum_k \overline{\rho q_k'' v''} = 0 . \quad (3.17)$$

Constraint (3.15) becomes obvious from physical intuition. The conversion of mass among different phases or particle forms must not create any net mass source or sink. Constraint (3.16) follows directly from Eq. (3.9) by inserting the definition of the barycentric velocity (3.1). Finally, for the proof of constraint (3.17) we refer to [Zdunkowski and Bott \(2003, p. 312\)](#).

In contrast to the original formulation of the prognostic equation set by [Gassmann and Herzog \(2008\)](#), we make use of the two-dimensional rather than the three-dimensional Lamb transformation. In general, the Lamb transformation uses vector calculus identities in order to convert the nonlinear momentum advection term $(\mathbf{v} \cdot \nabla) \mathbf{v}$ into a vector-invariant form. Vector-invariant means that no gradients of vectors appear in the prognostic velocity equation. This avoids derivatives of the coordinate basis (and, hence, Christoffel symbols) that would otherwise arise in an arbitrary coordinate frame from the nonlinear momentum advection term. In ICON, we apply the Lamb transformation only in the horizontal as follows:

$$\begin{aligned} \mathbf{v} \cdot \nabla \mathbf{v} &= \mathbf{v}_h \cdot \nabla \mathbf{v}_h + (\nabla \times \mathbf{v}_h) \times (w \mathbf{k}) + (\mathbf{v}_h \cdot \nabla w) \mathbf{k} \\ &= \nabla \left(\frac{1}{2} \mathbf{v}_h \cdot \mathbf{v}_h \right) + (\zeta \mathbf{k} \times \mathbf{v}_h) + (\nabla \times \mathbf{v}_h) \times (w \mathbf{k}) + (\mathbf{v}_h \cdot \nabla w) \mathbf{k} , \end{aligned}$$

with the 3D velocity vector \mathbf{v} , the 2D horizontal velocity vector \mathbf{v}_h and the vertical component of relative vorticity

$$\zeta = \mathbf{k} \cdot (\nabla \times \mathbf{v}) = \partial_x v - \partial_y u ,$$

see, e. g. the derivation in the appendix of [Ullrich et al. \(2017\)](#). Subsequent projection into the direction normal to a cell edge results in

$$\mathbf{e}_n \cdot (\mathbf{v} \cdot \nabla \mathbf{v}) = \frac{\partial}{\partial n} \left(\frac{1}{2} \mathbf{v}_h \cdot \mathbf{v}_h \right) + \zeta v_t + w \frac{\partial}{\partial z} v_n .$$

3.1.1. Computation of the Virtual Potential Temperature

The set of ICON's prognostic variables is sometimes causing confusion in the ICON community. Some people state that the Exner pressure $\bar{\pi}$ is the prognostic thermodynamic variable, while others refer to $\bar{\rho}\hat{\theta}_v$ as the prognostic variable, and even others state that both $\bar{\pi}$ and $\bar{\rho}\hat{\theta}_v$ are prognostic in ICON. At this point, we will try to shed some light on this, admittedly, confusing subject.

By taking the time derivative of the equation of state (3.10),

$$\frac{\partial \bar{\pi}}{\partial t} = \frac{R_d}{c_{vd}} \frac{\bar{\pi}}{\bar{\rho}\hat{\theta}_v} \frac{\partial \bar{\rho}\hat{\theta}_v}{\partial t} \quad (3.18)$$

the prognostic equation for Exner pressure (3.6) can be reformulated in terms of $\bar{\rho}\hat{\theta}_v$:

$$c_{pd}\bar{\pi} \frac{\partial \bar{\rho}\hat{\theta}_v}{\partial t} + c_{pd}\bar{\pi} \nabla \cdot (\bar{\rho}\hat{\mathbf{v}}\hat{\theta}_v) = c_{pd}\bar{\pi} \bar{\rho}\hat{\theta}_v \chi \nabla \cdot \hat{\mathbf{v}} + c_{pd}\bar{\pi} \bar{\rho}\bar{Q} \quad (3.19)$$

Note that equation (3.6) and (3.19) are equivalent, with the exception of the time derivative on the left hand side. In other words, the time evolution of the Exner pressure and virtual potential temperature are constrained by the time derivative of the equation of state (3.18).

In the dynamical core of ICON, which solves the adiabatic part of (3.6) ($\bar{Q} = 0$), $\hat{\theta}_v$ is computed from a linearized form of (3.18) rather than the equation of state (3.10), which reads

$$\frac{\bar{\rho}\hat{\theta}_v^{n+1} - \bar{\rho}\hat{\theta}_v^n}{\Delta t} = \frac{c_{vd}}{R_d} \left(\frac{\bar{\rho}\hat{\theta}_v^n}{\bar{\pi}^n} \right) \frac{\bar{\pi}^{n+1} - \bar{\pi}^n}{\Delta t}, \quad (3.20)$$

where n and $n+1$ denote the times t^n and t^{n+1} respectively, and Δt is the integration time step. Note that (3.20) is equivalent to the temporal discretized equation (3.19), which can be seen by applying the same temporal discretization to the Exner function in the prognostic equation (3.6), and substituting (3.6) into (3.20).

This procedure of computing $\hat{\theta}_v$ from (3.20) may be interpreted as solving two prognostic equations for the two thermodynamic variables $\bar{\pi}$ and $\bar{\rho}\hat{\theta}_v$ in a self-consistent manner, rather than following the usual way of solving one prognostic thermodynamic equation (e.g. for $\bar{\pi}$) and applying the equation of state (3.10) for diagnosing the second thermodynamic variable (e.g. for $\hat{\theta}_v$). As a consequence, the solution at a particular point in time is not constrained by the equation of state (3.10) – only its time evolution is. The benefit of this procedure is that global conservation of $\bar{\rho}\hat{\theta}_v$ is ensured to machine precision in the absence of diabatic terms.

This method of solving prognostic equations for two thermodynamic variables (3 when counting the equation for ρ (3.8)) has first been presented by Gassmann (2013).

3.2. Simplifying Assumptions in the Recent Model Version

The recent model version does not account for mass loss/gain due to precipitation/evaporation which, in essence, is a consequence of the approximated lower boundary conditions

for $\bar{\rho}\hat{w}$ (3.12) and \bar{J}_k^z (3.13). In order to shed some light on the nature of these approximations, we present the exact lower boundary conditions below (without proof), and compare them to (3.12) and (3.13).

$$\bar{\rho}\hat{w}|_s = \frac{\bar{E}_v - \sum_{k_{prec}} \bar{S}_k|_s}{1 - \sum_{k_{prec}} \hat{q}_k|_s} \quad (3.21)$$

$$\bar{J}_k^z|_s = \begin{cases} -\bar{\rho}\hat{q}_k\hat{w}|_s + \bar{E}_v, & \text{if } k \equiv v \\ -\bar{\rho}\hat{q}_k\hat{w}|_s, & \text{if } k \in \{d, c, i\} \\ -\bar{S}_k|_s, & \text{if } k \in k_{prec} = \{r, f\} \end{cases} \quad (3.22)$$

The exact lower boundary condition for $\bar{\rho}\hat{w}$ in a barycentric reference frame, Eq. (3.21), is approximated as

$$\bar{\rho}\hat{w}|_s = \sum_k \bar{\rho}_k \hat{w}_k|_s = 0, \quad (3.23)$$

which in terrain-following coordinates translates to $\bar{\rho}\hat{w}|_s = \bar{\rho}\hat{\mathbf{v}} \cdot \nabla h$. h denotes the topography height. Thus, the simulated atmospheric system is assumed to be closed w.r.t. total mass (i.e. no net mass flux across the surface), whereas the non-approximated system is open w.r.t. total mass (see (3.21)). In particular, the net surface mass flux is proportional to the difference of the evaporation flux $\bar{E}_v|_s$ and sedimentation fluxes $\bar{S}_k|_s$, which is usually nonzero. While neglected for the lower boundary condition of $\bar{\rho}\hat{w}$, the surface fluxes $\bar{E}_v|_s$ and $\bar{S}_k|_s$ are retained in the continuity equations for partial density (3.8). In order for the constraint $\sum_k \bar{J}_k^z = 0$ to hold at the surface, the (implicit) assumption is made that the corresponding mass loss/gain due to sedimentation/evaporation is compensated by a fictitious counter flux of dry air across the surface (see $\bar{J}_d^z|_s$ in (3.13)), such that the net mass flux is zero.

Away from the surface, the diffusion fluxes of all airborne constituents (except for dry air) are neglected in the approximated system. The diffusion fluxes of all precipitating constituents (the sedimentation fluxes), however, are retained (compare (3.22) with (3.13)). Since the continuity equation for the total density (3.7) only holds if the constraint $\sum_k \bar{J}_k^z = 0$ holds, again, the (implicit) assumption is made that a fictitious diffusion flux of dry air counteracts the sedimentation fluxes such that the total mass in a volume moving with the barycentric velocity is conserved.

In summary, the simplifying assumptions can be characterized by the following equivalent statements:

- The current model version globally conserves the total air mass instead of the dry air mass.
- The precipitation mass sink and the evaporation mass source are neglected in the total mass budget of the model.
- The net mass gain or loss due to precipitation/evaporation does not change the surface pressure.

The latter point becomes obvious, when writing down the hydrostatic pressure tendency equation and applying the approximate lower boundary condition (3.12) (see also Wacker and Herbert (2003)). Starting from the hydrostatic equation $\frac{\partial \bar{p}}{\partial z} = -\bar{\rho}g$ it follows:

$$\begin{aligned}\frac{\partial}{\partial t} \int_{\bar{p}_s}^0 dp &= -g \frac{\partial}{\partial t} \int_{z_s}^{\infty} \bar{\rho} dz \\ \frac{\partial \bar{p}_s}{\partial t} &= g \int_{z_s}^{\infty} \frac{\partial \bar{\rho}}{\partial t} dz \\ \frac{\partial \bar{p}_s}{\partial t} &= -g \int_{z_s}^{\infty} \nabla \cdot (\bar{\rho} \hat{\mathbf{v}}) dz \\ \frac{\partial \bar{p}_s}{\partial t} &= -g \int_{z_s}^{\infty} \nabla_h \cdot (\bar{\rho} \hat{\mathbf{v}}_h) dz + g (\bar{\rho} w)|_s \\ \frac{\partial \bar{p}_s}{\partial t} &= -g \int_{z_s}^{\infty} \nabla_h \cdot (\bar{\rho} \hat{\mathbf{v}}_h) dz + g \frac{1}{1 - \sum_{k \text{ prec}} \hat{q}_k|_s} \left(\bar{E}_v - \sum_{k \text{ prec}} \bar{S}_k|_s \right),\end{aligned}$$

Thus, dynamical effects of the evaporation/precipitation mass source/sink are neglected when using (3.12). I.e. related pressure changes are ignored.

3.3. The Model Reference State

Dynamics in the atmosphere are characterized by small variations of thermodynamic quantities with respect to some background state. Therefore, like many other modeling frameworks, ICON makes use of an atmospheric reference state, i.e. the thermodynamic variables are defined as the sum of a reference state and a deviation from that. The reference state is assumed to be at rest and horizontally homogeneous, constant in time, dry and hydrostatically balanced.

Any grid-scale thermodynamic variable $\hat{\psi}$ can then be written as

$$\hat{\psi}(\lambda, \phi, z, t) = \psi_0(z) + \psi'(\lambda, \phi, z, t).$$

The suffix 0 denotes the reference state while the prime denotes the grid-scale deviation. Thus, for the prognostic thermodynamic variables one gets

$$\begin{aligned}\bar{\rho}(\lambda, \phi, z, t) &= \rho_0(z) + \rho'(\lambda, \phi, z, t) \\ \bar{\pi}(\lambda, \phi, z, t) &= \pi_0(z) + \pi'(\lambda, \phi, z, t) \\ \hat{\theta}_v(\lambda, \phi, z, t) &= \theta_{v0}(z) + \theta'_v(\lambda, \phi, z, t).\end{aligned}$$

The background state components ρ_0 , π_0 , and θ_{v0} are related by the equation of state (3.10) and are hydrostatically balanced, i.e.

$$\begin{aligned}\pi_0 &= \left(\frac{R_d}{p_{00}} \rho_0 \theta_{v0} \right)^{\frac{R_d}{c_{vd}}} \\ \frac{d\pi_0}{dz} &= -\frac{g}{c_{pd} \theta_{v0}}\end{aligned}\tag{3.24}$$

The vertical reference profiles can be obtained by integration of Eq. (3.24), given that suitable boundary values are provided. The reference state in ICON is identical to the state used by the COSMO model.

In a global model like ICON, the local deviation from a horizontally homogeneous reference state can be quite substantial. Therefore, no attempt is made to linearize any part of the governing equations with regard to this reference state as it is done in models based on the anelastic assumption. Instead, the main effect of introducing a reference state in a global nonhydrostatic model is the removal of horizontal base-state pressure gradient terms in the equation of motion, i.e.

$$c_{pd}\hat{\theta}_v\frac{\partial\bar{\pi}}{\partial n} = c_{pd}\hat{\theta}_v\frac{\partial\pi'}{\partial n}.$$

This reduces the computational error in the calculation of the pressure gradient force in case of sloping coordinate surfaces. Having said that, this effect is of minor importance for ICON, as by default the horizontal pressure gradient is evaluated truly horizontal along surfaces of constant height, rather than in terrain-following coordinates along sloping coordinate surfaces (Zängl, 2012).

Changing the discretization of the horizontal pressure gradient is possible with the namelist switch `igradp_method (nonhydrostatic_nml)`, but not recommended. Nevertheless, the reference state is still of some use for ICON. In the standard configuration of ICON, explicit use of the reference state is made when computing the advective horizontal fluxes for ρ and θ_v .

With the base state at hand, the vertical acceleration due to the pressure gradient and gravity in Eq. (3.5) is rewritten as

$$\begin{aligned} -c_{pd}\hat{\theta}_v\frac{\partial\bar{\pi}}{\partial z} - g &= -c_{pd}(\theta_{v0} + \theta'_v)\frac{\partial(\pi_0 + \pi')}{\partial z} \\ &= -c_{pd}\left(\hat{\theta}_v\frac{\partial\pi'}{\partial z} + \theta'_v\frac{d\pi_0}{dz}\right) - c_{pd}\theta_{v0}\frac{d\pi_0}{dz} - g \\ &\stackrel{(3.24)}{=} -c_{pd}\left(\hat{\theta}_v\frac{\partial\pi'}{\partial z} + \theta'_v\frac{d\pi_0}{dz}\right) \end{aligned}$$

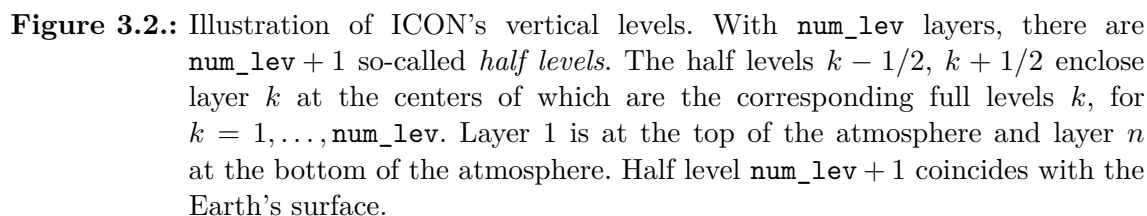
The vertical momentum equation finally reads

$$\frac{\partial\hat{w}}{\partial t} + \hat{\mathbf{v}}_h \cdot \nabla\hat{w} + \hat{w}\frac{\partial\hat{w}}{\partial z} = -c_{pd}\left(\hat{\theta}_v\frac{\partial\pi'}{\partial z} + \theta'_v\frac{d\pi_0}{dz}\right) - \frac{1}{\bar{\rho}}\frac{\partial}{\partial z}\overline{\rho\mathbf{v}''\mathbf{v}''}.$$

The perturbation fields π' and θ'_v are obtained from the predicted full fields and reference fields via $\psi' = \hat{\psi} - \psi_0$.

3.4. Vertical Coordinates

In a nonhydrostatic model, it cannot be taken for granted that the pressure is monotonously decreasing with increasing altitude. Moreover, the pressure at a certain point does not necessarily represent the mass of the air column above, as it is the case for



In ICON the choice is a height based coordinate system that follows the terrain and consequently, the top and bottom triangle faces are inclined with respect to the tangent plane on a sphere. Due to the fact that the model levels gradually change into levels of constant height as the distance from the lower boundary increases, top and bottom triangle faces of a grid box are also slightly inclined to each other. The exact altitude of each grid box depends on the geographical position on the globe. The top and bottom faces are called *half levels* of the vertical grid, the center of the box is said to be at the *full level* of the vertical grid, see Fig. 3.2 for an illustration. Note that the numbering of full and half levels is top-down, starting with $k = 1$ for the top half- and full level. A Lorenz-type staggering is used in the vertical, which means that horizontal velocity, virtual potential temperature and density are defined at full levels, whereas vertical velocity is defined at half levels.

num_lev (namelist **run_nml**, list of integer value)
Comma-separated list of integer values giving the number of vertical full levels for each domain.

If the number of vertical levels is desired to vary between domains, setting the namelist parameter `lvert_nest (run_nml)` to `.TRUE.` is required. See Section 3.9.1 for more information on vertical nesting.

Two variants of a height-based terrain-following vertical coordinate are available in ICON. Both of which are briefly described in the following section.

General vertical height coordinate. It will become clear from the description of the terrain-following coordinate below that the exact vertical axis definition depends on a multitude of parameter settings. This makes it virtually impossible to encode the exact vertical coordinate parameters themselves in the appropriate section of the GRIB code. The data sets which are produced by the ICON model therefore contain only a *reference* to a vertical grid. Apart from very basic information like the number of vertical levels, only a number identifying the special vertical grid used is provided. The actual vertical height coordinate is then specified by providing a 3D (GRIB2) field which defines the height of every grid point.

This indirect *reference grid* approach raises the same questions that played a role in the handling of the horizontal grid, see Section 2.1.7: In order to find out if identical vertical coordinate options were used for two given data sets, the GRIB2 data records contain special meta-data items, namely `numberOfVGridUsed` and `uuidOfVGrid`.

3.4.1. Terrain-following Hybrid Gal-Chen Coordinate

The *terrain-following hybrid Gal-Chen coordinate* (Simmons and Burridge, 1981) is an extension of the classic terrain-following coordinate introduced by Gal-Chen and Somerville (1975). As shown by Klemp (2011), it can be expressed in the form

$$\begin{aligned} z(x, y, \eta) &= \frac{(H - B'(\eta) h(x, y))}{H} \eta + B'(\eta) h(x, y) \\ &= \eta + B'(\eta) \left(1 - \frac{\eta}{H}\right) h(x, y), \end{aligned} \quad (3.25)$$

where z represents the height of the coordinate surfaces defined by η , $h(x, y)$ is the terrain height, and H denotes the domain height. With $B'(\eta) = 1$ the coordinate reverts to the classic formulation by Gal-Chen and Somerville (1975), i.e. the coordinate is terrain-following at the surface ($\eta = 0$) and becomes flat at model top ($\eta = H$). By choosing B' appropriately, a more rapid transition from terrain-following at the surface toward constant height can be achieved. One popular choice is to set

$$B'(\eta) \left(1 - \frac{\eta}{H}\right) = 1 - \frac{\eta}{z_{flat}}, \quad \text{with } z_{flat} < H$$

such that coordinate surfaces become constant height surfaces above $z = z_{flat}$. Oftentimes, Equation (3.25) is also written in the discretized form

$$z_h(x, y, k) = A(k) + B(k) h(x, y), \quad k = 1, \dots, \text{num_lev} + 1 \quad (3.26)$$

where k denotes the vertical level index and z_h is the half level height.

Configuring the Hybrid Gal-Chen Coordinate

The main switch for selecting the Gal-Chen hybrid coordinate is

```
ivctype = 1 (namelist nonhydrostatic_nml, integer value)
```

The user has to provide the vertical coordinate table (vct) as an input file, using the namelist variable `vct_filename` in the namelist `grid_nml`. The table consists of the A and B values (see Equation (3.26)) from which the half level heights $z_h(x, y, k)$ can be deduced. $A(k)[m]$ contains fixed height values, with $A(1)$ defining the model top height H and $A(\text{num_lev} + 1) = 0m$. The dimensionless values $B(k)$ control the vertical decay of the topography signal, with $B(1) = 0$ and $B(\text{num_lev} + 1) = 1$. Thus, $z_h(x, y, 1)$ is equivalent to the model top height, while $z_h(x, y, \text{num_lev} + 1)$ is the surface height.

The structure of the expected input file is depicted in Table 3.2. Example files can be found in `icon/vertical_coord_tables`.

Please note that for idealized runs (i.e. `ltestcase=TRUE` (namelist `run_nml`)) with equidistant vertical levels, it is possible to create the vertical coordinate table on the fly during the initialization phase of ICON by specifying the layer thickness `layer_thickness` and number of flat levels `n_flat_lev` in the namelist `nh_testcases_nml`.

```
# File structure
# -----
# A and B values are stored in arrays vct_a(k) and vct_b(k).
# The files in text format are structured as follows:
#
# -----
# |   k      vct_a(k) [m]   vct_b(k) [] | <- first line of file = header line
# |   1      A(1)          B(1)      | <- first line of A and B values
# |   2      A(2)          B(2)      |
# |   3      A(3)          B(3)      |
# |   .      .            .          |
# |   .      .            .          |
# | nlev+1    A(nlev+1)      B(nlev+1) | <- last line of A and B values
# |=====| <- lines from here on are ignored
# |Source:                                     | by mo_hyb_params:read_hyb_params
# |<some lines of text>                         |
# |Comments:                                   |
# |<some lines of text>                         |
# |References:                                 |
# |<some lines of text>                         |
# -----
```

Table 3.2.: Structure of vertical coordinate table as expected by the ICON model.

3.4.2. SLEVE Coordinate

In the case of a terrain-following hybrid Gal-Chen coordinate the influence of terrain on the coordinate surfaces decays only linearly with height. The basic idea of the *Smooth Level*

Vertical SLEVE coordinate (Schär et al., 2002, Leuenberger et al., 2010) is to increase the decay rate, by allowing smaller-scale terrain features to be removed more rapidly with height. To this end, the topography $h(x, y)$ is divided into two components

$$h(x, y) = h_1(x, y) + h_2(x, y),$$

where $h_1(x, y)$ denotes a smoothed representation of $h(x, y)$, and $h_2(x, y) = h(x, y) - h_1(x, y)$ contains the smaller-scale contributions. The coordinate is then defined as

$$z(x, y, \eta) = \eta + B_1(\eta) h_1(x, y) + B_2(\eta) h_2(x, y).$$

Different decay functions B_1 and B_2 are chosen for the decay of the large- and small-scale terrain features, respectively. These functions are selected such that the influence of small-scale terrain features on the coordinate surfaces decays much faster with height than their large-scale (well-resolved) counterparts. The squeezing of the model layers above steep mountains is limited automatically in order to prevent (nearly) intersecting layers that would cause numerical instabilities.

Configuring the SLEVE Coordinate

The main switch for selecting the SLEVE vertical coordinate is

```
ivctype = 2 (namelist nonhydrostatic_nml, integer value)
```

This is the default and recommended setting. The vertical grid is constructed during the initialization phase of ICON, based on additional parameters defined in `sleve_nml`. Here we will only discuss the most relevant parameters. For a full list, the reader is referred to the namelist documentation.

Namelist `sleve_nml`:

top_height (namelist `sleve_nml`, real value)

Height of model top.

flat_height (namelist `sleve_nml`, real value)

Height above which the coordinate surfaces become constant height surfaces.

min_lay_thckn (namelist `sleve_nml`, real value)

Layer thickness of lowermost layer.



Note for advanced users: On default, a vertical stretching is applied such that coordinate surfaces become non-equally distributed along the vertical, starting with a minimum thickness of `min_lay_thckn` between the lowermost and second lowermost half-level. If constant layer thicknesses are desired, `min_lay_thckn` must be set to a value ≤ 0 . The layer thickness is then determined as `top_height/num_lev`. Control output of the vertical layer distribution is written to `stderr`.

Similar to the Gal-Chen coordinate, it is possible to read the vertical coordinate table from file, by specifying a file name and path via `vct_filename` (`grid_nml`). Please note that the SLEVE coordinate only requires the height values $A(k)$. It is recommended to set the unused $B(k)$ values to zero. Reading the vertical grid information from file becomes handy if the same vertical level distribution is desired as used e.g. by the operational model suite at DWD. In particular, the reproduction of the level distribution used by the vertically nested ICON-EU domain is not possible with the available `sleve_nml` namelist variables. This is due to the fact that the ICON-EU level distribution is generated by removing the uppermost 30 levels from the level distribution used by the global model domain.

3.5. Temporal Discretization

In this section we will focus on time differencing in isolation and will neglect any complexity due to space differencing. Before we dive into the (nasty) details of ICON's time discretization, let us take a step back and try to grasp the basic idea.

3.5.1. Basic Idea

Consider an arbitrary first-order ordinary differential equation of the form

$$\frac{d\psi(\vec{x}, t)}{dt} = F(\psi(\vec{x}, t), t). \quad (3.27)$$

Here, $\psi : \mathbb{R}^3 \times \mathbb{R} \rightarrow \mathbb{R}^q$ is the q -dimensional vector of state variables. In real applications, the vector-valued flux function $F : \mathbb{R}^q \times \mathbb{R} \rightarrow \mathbb{R}^q$ might be very complex.

Let t_{n-m} denote some time in the past and t_{n+1} some time in the future. Now we integrate Eq. (3.27) from time t_{n-m} to t_{n+1} , which gives

$$\psi(\vec{x}, t_{n+1}) - \psi(\vec{x}, t_{n-m}) = \int_{t_{n-m}}^{t_{n+1}} F(\psi(\vec{x}, t), t) dt.$$

We can try to approximate the integral on the r.h.s. using some weighted average of F at known discrete time levels. In the following we will restrict ourselves to simple two-level time-stepping schemes, i.e. we set $m = 0$ and only make use of F at the discrete times t_n and t_{n+1} (for a general survey of time-differencing schemes, see [Randall \(2017\)](#)). With this restriction we get

$$\frac{\psi(\vec{x}, t_{n+1}) - \psi(\vec{x}, t_n)}{\Delta t} = \alpha F_{n+1} + \beta F_n. \quad (3.28)$$

The coefficients α and β must satisfy the so called *consistency condition*

$$\alpha + \beta = 1,$$

such that the r.h.s. of Eq. (3.28) represents some averaged F . Equation (3.28) represents a whole family of simple two-step schemes. For example, by choosing $\alpha = 0$, $\beta = 1$ we arrive at the simple Euler forward scheme, while for $\alpha = 1$, $\beta = 0$ we get the (implicit) Euler backward scheme, both of which are first order accurate. Choosing $\alpha = \beta = 0.5$ leads to the implicit *trapezoidal* scheme, which is of second order accuracy.

One way to avoid the implicitness of the trapezoidal scheme while retaining higher order is to switch to iterative schemes, also known as *predictor-corrector* schemes. This is the way pursued in ICON. The key idea of the predictor-corrector scheme is to replace the unwieldy F_{n+1} by an estimate $F_{n+1}^* = F(\psi_{n+1}^*, t_{n+1})$ with ψ_{n+1}^* computed by an explicit scheme, e.g. a forward Euler scheme. The full predictor-corrector scheme reads

$$\begin{aligned} \text{predictor :} \quad & \psi^*(\vec{x}, t_{n+1}) = \psi(\vec{x}, t_n) + \Delta t F_n \\ \text{corrector :} \quad & \psi(\vec{x}, t_{n+1}) = \psi(\vec{x}, t_n) + \Delta t \{F_{n+1}^*, F_n\}_\alpha \end{aligned} \quad (3.29)$$

Here we have introduced the notation

$$\{x, y\}_\alpha := \alpha x + (1 - \alpha) y \quad .$$

Note that for $\alpha = 1$, Equation (3.29) is an imitation of the Euler backward scheme (termed Matsuno scheme, (Matsuno, 1966)), while for $\alpha = 0.5$, it is an imitation of the *trapezoidal* scheme (termed *Heun's method*). The Matsuno scheme has first-order accuracy, and Heun's method has second-order accuracy.



In simplified terms, the time integration scheme of ICON can be regarded as a mixture of the Matsuno scheme ($\alpha = 1$) and the Heun scheme ($\alpha = 0.5$), as the coefficient α used by ICON varies between these two extremes.

3.5.2. Implementation Details

Now, in the terminology of the previous section, in ICON we have

$$\psi(\vec{x}, t) = [v_n(\vec{x}, t), w(\vec{x}, t), \rho(\vec{x}, t), \pi(\vec{x}, t)]^\top \quad .$$

Here, we have omitted the partial densities ρq_k from the vector of state variables. They are treated with a different time discretization scheme, as will be explained in Section 3.6.

For the sake of brevity we omit the notation for Reynolds- and Hesselberg averages. The superscripts n , $n+1^*$ and $n+1$ will be used to denote the current time level, the resulting time level of the predictor step and the new time level, respectively. They should not be confused with the subscript n used to denote the normal velocity component v_n , or the horizontal derivative in edge-normal direction $\partial/\partial n$.

The time discretization complies with the explicit two-time level predictor-corrector scheme which was described in the previous section, except for those terms describing vertical sound-wave propagation. These terms, i.e. vertical derivatives of w and π , are treated implicitly for reasons of numerical stability and efficiency. Below, the implicit terms are marked in blue .

Predictor step:

$$\frac{v_n^{n+1*} - v_n^n}{\Delta t} = -\text{adv}(v_n^n) - c_{pd}\theta_v^n \frac{\partial \pi'^{n,n}}{\partial n} + F(v_n^n) \quad (3.30)$$

$$\frac{w^{n+1*} - w^n}{\Delta t} = -\text{adv}(w^n) - c_{pd}\theta_v'^{n,n} \frac{d\pi_0}{dz} - c_{pd}\theta_v^n \left\{ \frac{\partial \pi'^{n,n+1*}}{\partial z}, \frac{\partial \pi'^{n,n}}{\partial z} \right\}_\eta \quad (3.31)$$

$$\begin{aligned} \frac{\rho^{n+1*} - \rho^n}{\Delta t} &= -\nabla_h \cdot (v_n^{n+1*} \rho^n) - \frac{\partial}{\partial z} \left[\{w^{n+1*}, w^n\}_\eta \rho^n \right] \\ \frac{\pi^{n+1*} - \pi^n}{\Delta t} &= -\frac{R_d}{c_{vd}} \left(\frac{\pi^n}{\rho^n \theta_v^n} \right) \left[\nabla_h \cdot (v_n^{n+1*} \rho^n \theta_v^n) \right. \\ &\quad \left. + \frac{\partial}{\partial z} \left[\{w^{n+1*}, w^n\}_\eta \rho^n \theta_v^n \right] \right] + Q^n \end{aligned} \quad (3.32)$$

with

$$\begin{aligned} \text{adv}(v_n^n) &= \frac{\partial K_h^n}{\partial n} + (\zeta^n + f) v_t^n + w^n \frac{\partial v_n^n}{\partial z} \\ \text{adv}(w^n) &= v_h^n \cdot \nabla w^n + w^n \frac{\partial w^n}{\partial z}. \end{aligned}$$

The terms $F(v_n^n)$ and Q^n denote diabatic momentum and Exner pressure tendencies due to *slow physics processes*, i.e. parameterized convection, orographic and non-orographic gravity waves and radiation. See Section 3.7.1 for more details on the distinction between fast and slow physics processes.

The implicitness parameter η (with $0 \leq \eta \leq 1$) for the vertically implicit sound wave solver has a default value of $\eta = 0.65$ which is usually sufficient to ensure numerical stability in real-case applications. If required, this value can be modified with the namelist parameter `vwind_offctr (nonhydrostatic_nml)`. Note that `vwind_offctr` presents the off-centering from 0.5, i.e. `vwind_offctr` = $\eta - 0.5$. Its permissible range is given by

$$0 \leq \text{vwind_offctr} \leq 0.5,$$

and the default value is `vwind_offctr` = 0.15.

Corrector step:

$$\begin{aligned}
\frac{v_n^{n+1} - v_n^n}{\Delta t} &= - \left\{ \text{adv}(v_n^{n+1*}), \text{adv}(v_n^n) \right\}_\alpha \\
&\quad - c_{pd} \left\{ \theta_v^{n+1*}, \theta_v^n \right\}_\delta \left\{ \frac{\partial \pi'^{n+1*}}{\partial n}, \frac{\partial \pi'^n}{\partial n} \right\}_\delta \\
&\quad - F_d(v_n^{n+1*}) + F(v_n^n) \\
\frac{w^{n+1} - w^n}{\Delta t} &= - \left\{ \text{adv}(w^{n+1*}), \text{adv}(w^n) \right\}_\alpha \\
&\quad - c_{pd} \left\{ \theta_v'^{n+1*}, \theta_v'^n \right\}_\delta \frac{d\pi_0}{dz} \\
&\quad - c_{pd} \left\{ \theta_v^{n+1*}, \theta_v^n \right\}_\delta \left\{ \frac{\partial \pi'^{n+1}}{\partial z}, \frac{\partial \pi'^n}{\partial z} \right\}_\eta \\
\frac{\rho^{n+1} - \rho^n}{\Delta t} &= - \nabla_h \cdot (v_n^{n+1} \rho^n) - \frac{\partial}{\partial z} \left[\{w^{n+1}, w^n\}_\eta \{ \rho^{n+1*}, \rho^n \}_\delta \right] \\
\frac{\pi^{n+1} - \pi^n}{\Delta t} &= - \frac{R_d}{c_{vd}} \left(\frac{\pi^n}{\rho^n \theta_v^n} \right) \left[\nabla_h \cdot (v_n^{n+1} \rho^n \theta_v^n) \right. \\
&\quad \left. + \frac{\partial}{\partial z} \left[\{w^{n+1}, w^n\}_\eta \{ \rho^{n+1*}, \rho^n \}_\delta \{ \theta_v^{n+1*}, \theta_v^n \}_\delta \right] \right] \\
&\quad + Q^n
\end{aligned} \tag{3.33}$$

The term $F_d(v_n^{n+1*})$ represents 4th order divergence damping which has been introduced in order to control checkerboard noise.

The parameter α denotes the Matsuno parameter which was introduced in Section 3.5.1. With respect to the velocity, it can be used to make the explicit part of the corrector step resemble either a Matsuno-type scheme ($\alpha \rightarrow 1$) or Heun's method ($\alpha \rightarrow 0.5$). The default value of the Matsuno-parameter for velocity is given by $\alpha = 0.75$.

A second Matsuno parameter $0 \leq \delta \leq 1$ exists for the thermodynamic variables ρ and θ_v , where the default value of the coefficient is $\delta = 0.4$. The corresponding namelist parameters are named `veladv_offctr` and `rhotheta_offctr` in the namelist `nonhydrostatic_nml`. Note again that both define the off-centering from 0.5 rather than the absolute value.

In order to close the system of equations, the quantity θ_v must be calculated from the state variables π and ρ for both substeps (predictor and corrector). We explain this in the following, using the notation $\pi^{\text{new}} \equiv \pi^{n+1*}$ or $\pi^{\text{new}} \equiv \pi^{n+1}$ for the predictor and corrector step.

Once the updated state vector

$$\psi^{\text{new}}(\vec{x}, t) = [v_n^{\text{new}}(\vec{x}, t), w^{\text{new}}(\vec{x}, t), \rho^{\text{new}}(\vec{x}, t), \pi^{\text{new}}(\vec{x}, t)]^T$$

is known, the virtual potential temperature θ_v^{new} is computed from the linearized equation of state (3.20), recast in the form

$$\theta_v^{\text{new}} = \theta_v^n \frac{\rho^n}{\rho^{\text{new}}} \left[1 + \frac{c_{vd}}{R_d} \left(\frac{\pi^{\text{new}}}{\pi^n} - 1 \right) \right].$$

See Section 3.1.1 for details and the characterization of this procedure.

Pragmatic Simplifications

A potential disadvantage of predictor-corrector schemes as compared to non-iterative schemes is its computational expense. This is because the forcing terms (r.h.s.) must be evaluated twice per time step. Therefore, several terms have been simplified provided that the simplification proved to not degrade the quality of the results significantly. The following pragmatic simplifications have been performed:

- The horizontal and vertical momentum advection at the predictor step is re-used from the corrector step of the preceding time step. With time level n^* denoting $n + 1^*$ from the preceding time step this can be written as

$$\text{Equation (3.30):} \quad \text{adv}(v_n^n) \simeq \text{adv}(v_n^{n^*})$$

$$\text{Equation (3.31):} \quad \text{adv}(w^n) \simeq \text{adv}(w^{n^*})$$

By this, in each time step the momentum advection needs to be computed only once.

The first predictor step following the physics step represents an exception. At this time level the momentum advection from the preceding corrector step does not provide a suitable estimate, as the physics step might have changed v_n considerably.

- Another simplification relates to the horizontal pressure gradient term which occurs in the predictor and corrector step of v_n . Using time level n in the predictor and an interpolated value between n and $n + 1^*$ in the corrector provides an effective damping mechanism for horizontally propagating sound waves, without significantly impacting gravity waves (Klemp et al., 2007). A very similar effect can be achieved by using the same horizontal pressure gradient in the predictor and corrector, however, with the pressure being extrapolated in time.

$$\text{Equation (3.30):} \quad c_{pd}\theta_v^n \frac{\partial \pi'^{n,n}}{\partial n} \simeq c_{pd}\theta_v^n \frac{\partial \pi'^{n,\tilde{n}}}{\partial n}$$

$$\text{Equation (3.33):} \quad c_{pd} \left\{ \theta_v^{n+1^*}, \theta_v^n \right\}_\delta \left\{ \frac{\partial \pi'^{n,n+1^*}}{\partial n}, \frac{\partial \pi'^{n,n}}{\partial n} \right\}_\delta \simeq c_{pd}\theta_v^n \frac{\partial \pi'^{n,\tilde{n}}}{\partial n}$$

By this, in each time step the horizontal pressure gradient needs to be computed only once. The time level \tilde{n} at which the horizontal gradient is taken is an extrapolated time level using the levels n and $n - 1$:

$$\pi'^{n,\tilde{n}} = (1 + \gamma)\pi'^{n,n} - \gamma\pi'^{n-1,n}$$

The temporal extrapolation factor is chosen from the range $\gamma \in [\frac{1}{3}, \frac{2}{3}]$, with the default being $\gamma = 1/3$. The corresponding namelist parameter is named `exner_expol` (`nonhydrostatic_nml`).

Vertically Implicit Solver

The solution to the predictor-corrector scheme described above is mostly straightforward, as the majority of terms are treated in an explicit manner. This, however does not hold for the prognostic equation for vertical wind, since the solution for w^{n+1^*} depends on π^{n+1^*} , which itself depends on w^{n+1^*} (see Equations (3.31) and (3.32)).

The overall solution strategy is as follows: First, π is eliminated from Eq. (3.31) by inserting the prognostic equation (3.32). This results in a linear system of equations for the unknown w 's in the vertical direction. Once w^{n+1*} is known, the Exner equation (3.32) can be solved. The derivation for the corrector step is basically identical and differs only w.r.t. the time levels that are used for ρ and θ_v .

We start with the vertical discretization of (3.31) and (3.32). When using basic centered differences, and noting that $\partial\pi/\partial t = \partial\pi'/\partial t$ this leads to

$$w_{k+1/2}^{n+1*} = Z_{k+1/2}^{w \text{ expl}} - \Delta t c_{pd} \theta_{v,k+1/2}^n \eta \frac{\pi_k'^{n+1*} - \pi_{k+1}^{n+1*}}{\Delta z_{k+1/2}} \quad (3.34)$$

$$\pi_k'^{n+1*} = Z_k^{\pi \text{ expl}} - \Delta t \frac{R_d}{c_{vd}} \left(\frac{\pi_k^n}{\rho_k^n \theta_{v,k}^n} \right) \eta \frac{(w^{n+1*} \rho^n \theta_v^n)_{k-1/2} - (w^{n+1*} \rho^n \theta_v^n)_{k+1/2}}{\Delta z_k}, \quad (3.35)$$

with the shorthand notations $Z_{k+1/2}^{w \text{ expl}}$ and $Z_k^{\pi \text{ expl}}$ for the explicit parts

$$\begin{aligned} Z_{k+1/2}^{w \text{ expl}} = & w_{k+1/2}^n - \Delta t \left[\text{adv}(w_n^*)_{k+1/2} + c_{pd} \theta_{v,k+1/2}^{n,n} \frac{d\pi_0}{dz} \Big|_{k+1/2} \right. \\ & \left. + c_{pd} \theta_{v,k+1/2}^n (1 - \eta) \frac{\pi_k'^{n,n} - \pi_{k+1}^{n,n}}{\Delta z_{k+1/2}} \right] \end{aligned}$$

and

$$\begin{aligned} Z_k^{\pi \text{ expl}} = & \pi_k'^{n,n} - \Delta t \frac{R_d}{c_{vd}} \left(\frac{\pi_k^n}{\rho_k^n \theta_{v,k}^n} \right) \left[\nabla_h \cdot (v^{n+1*} \rho^n \theta_v^n)_k \right. \\ & \left. + (1 - \eta) \frac{(w^n \rho^n \theta_v^n)_{k-1/2} - (w^n \rho^n \theta_v^n)_{k+1/2}}{\Delta z_k} \right] + \Delta t Q_n^k. \end{aligned}$$

$\Delta z_k = z_{k-1/2} - z_{k+1/2}$ denotes the thickness of the k^{th} cell which is bounded by the half levels $k \pm 1/2$, whereas $\Delta z_{k+1/2} = z_k - z_{k+1}$ denotes the thickness of the layer bounded by the full levels k and $k+1$ (see also Figure 3.2).

As an intermediate step, we compute $\pi_k'^{n+1*} - \pi_{k+1}^{n+1*}$ from Eq. (3.35):

$$\begin{aligned} \pi_k'^{n+1*} - \pi_{k+1}^{n+1*} = & Z_k^{\pi \text{ expl}} - Z_{k+1}^{\pi \text{ expl}} - \frac{\Delta t R_d}{c_{vd}} \eta \left[(w^{n+1*} \rho^n \theta_v^n)_{k-1/2} \frac{\Gamma_k^n}{\Delta z_k} \right. \\ & - (w^{n+1*} \rho^n \theta_v^n)_{k+1/2} \left(\frac{\Gamma_k^n}{\Delta z_k} + \frac{\Gamma_{k+1}^n}{\Delta z_{k+1}} \right) \\ & \left. + (w^{n+1*} \rho^n \theta_v^n)_{k+3/2} \frac{\Gamma_{k+1}^n}{\Delta z_{k+1}} \right]. \end{aligned} \quad (3.36)$$

Here we have introduced Γ_k^n as an abbreviation for

$$\Gamma_k^n = \frac{\pi_k^n}{\rho_k^n \theta_{v,k}^n}.$$

Inserting Eq. (3.36) into Eq. (3.34) and collecting terms proportional to $w_{k-1/2}^{n+1*}$, $w_{k+1/2}^{n+1*}$, $w_{k+3/2}^{n+1*}$ on the left hand side leads to:

$$\begin{aligned}
 & - w_{k-1/2}^{n+1*} \underbrace{\left[\Delta t \eta \frac{c_{pd} \theta_{v,k+1/2}^n}{\Delta z_{k+1/2}} \Delta t \frac{R_d}{c_{vd}} \frac{\Gamma_k^n}{\Delta z_k} \rho_{k-1/2}^n \theta_{v,k-1/2}^n \eta \right]}_{\text{sub-diagonal}} \\
 & + w_{k+1/2}^{n+1*} \underbrace{\left[1 + \Delta t \eta \frac{c_{pd} \theta_{v,k+1/2}^n}{\Delta z_{k+1/2}} \Delta t \frac{R_d}{c_{vd}} \left(\frac{\Gamma_k^n}{\Delta z_k} + \frac{\Gamma_{k+1}^n}{\Delta z_{k+1}} \right) \rho_{k+1/2}^n \theta_{v,k+1/2}^n \eta \right]}_{\text{main diagonal}} \\
 & - w_{k+3/2}^{n+1*} \underbrace{\left[\Delta t \eta \frac{c_{pd} \theta_{v,k+1/2}^n}{\Delta z_{k+1/2}} \Delta t \frac{R_d}{c_{vd}} \frac{\Gamma_{k+1}^n}{\Delta z_{k+1}} \rho_{k+3/2}^n \theta_{v,k+3/2}^n \eta \right]}_{\text{sup-diagonal}} \\
 & = Z_{k+1/2}^{w \text{ expl}} - \Delta t \eta \frac{c_{pd} \theta_{v,k+1/2}^n}{\Delta z_{k+1/2}} \left(Z_k^{\pi \text{ expl}} - Z_{k+1}^{\pi \text{ expl}} \right) \tag{3.37}
 \end{aligned}$$

Equation (3.37) defines a linear system of equations from which the unknown vertical velocities $w_{k-1/2}^{n+1*}$ can be computed. The system is tridiagonal and can be solved with the Thomas algorithm (Press et al., 2007, p. 57), given that suitable boundary conditions are provided. So far it is assumed that the upper and lower boundary are impermeable w.r.t. to mass, i.e. we set $w_{1/2}^{n+1*} = w_{\text{nlev}+1/2}^{n+1*} = 0$. In case of vertical nesting, w at the upper boundary is usually nonzero and is interpolated from the parent domain (see Section 3.9.1 for details).

In order to be consistent with the implementation in the ICON code, we introduce the following abbreviations:

$$\begin{aligned}
 \alpha_{k+1/2} &= \rho_{k+1/2}^n \theta_{v,k+1/2}^n \eta \\
 \beta_k &= \Delta t \frac{R_d}{c_{vd}} \frac{\Gamma_k^n}{\Delta z_k} \\
 \gamma_{k+1/2} &= \Delta t \eta \frac{c_{pd} \theta_{v,k+1/2}^n}{\Delta z_{k+1/2}}
 \end{aligned}$$

Note that α and γ are defined on half levels, while β is defined on full levels. The tridiagonal system (3.37) can now be written in the form ($a_{1/2} = 0$, $c_{\text{nlev}+1/2} = 0$)

$$a_{k+1/2} w_{k-1/2}^{n+1*} + b_{k+1/2} w_{k+1/2}^{n+1*} + c_{k+1/2} w_{k+3/2}^{n+1*} = d_{k+1/2}, \quad k = 0, \dots, \text{nlev},$$

or in matrix form

$$\begin{pmatrix} b_{1/2} & c_{1/2} & & & \\ a_{3/2} & b_{3/2} & c_{3/2} & & \\ & & \ddots & & \\ & & & a_{\text{nlev}-1/2} & b_{\text{nlev}-1/2} & c_{\text{nlev}-1/2} \\ & & & a_{\text{nlev}+1/2} & b_{\text{nlev}+1/2} & c_{\text{nlev}+1/2} \end{pmatrix} \begin{pmatrix} w_{1/2}^{n+1*} \\ w_{3/2}^{n+1*} \\ \vdots \\ w_{\text{nlev}-1/2}^{n+1*} \\ w_{\text{nlev}+1/2}^{n+1*} \end{pmatrix} = \begin{pmatrix} d_{1/2} \\ d_{3/2} \\ \vdots \\ d_{\text{nlev}-1/2} \\ d_{\text{nlev}+1/2} \end{pmatrix}$$

with the coefficients

$$\begin{aligned} a_{k+1/2} &= -\gamma_{k+1/2} \beta_k \alpha_{k-1/2} \\ b_{k+1/2} &= 1 + \gamma_{k+1/2} (\beta_k + \beta_{k+1}) \alpha_{k+1/2} \\ c_{k+1/2} &= -\gamma_{k+1/2} \beta_{k+1} \alpha_{k+3/2} \\ d_{k+1/2} &= Z_{k+1/2}^{w \text{ expl}} - \gamma_{k+1/2} \left(Z_k^{\pi \text{ expl}} - Z_{k+1}^{\pi \text{ expl}} \right) \end{aligned}$$

for $k = 1, \dots, \text{nlev} - 1$, and the Dirichlet boundary conditions

$$\begin{array}{llll} \text{top:} & b_{1/2} = 1 & c_{1/2} = 0 & d_{1/2} = w_{top} \\ \text{bottom:} & a_{\text{nlev}+1/2} = 0 & b_{\text{nlev}+1/2} = 1 & d_{\text{nlev}+1/2} = w_{bot} . \end{array}$$

3.6. Tracer Transport

The transport module is an important building block of any numerical weather prediction (NWP) or climate model, as it predicts the large-scale redistribution of water substances, chemical constituents or aerosols in the atmosphere due to air motion. Mathematically, it solves one of the fundamental laws of physics, namely the equation of tracer mass continuity (3.8). The transport module itself does not take into account tracer sources or sinks. It only predicts its large scale redistribution. Hence, for each tracer the transport module solves the simplified continuity equation

$$\frac{\partial \bar{\rho} \hat{q}_k}{\partial t} + \nabla \cdot (\bar{\rho} \hat{q}_k \hat{\mathbf{v}}) = 0 \quad (3.38)$$

(compare with Eq. (3.8)). Additional sources and sinks as well as turbulent diffusion are accounted for in the physics interface with a fractional step approach (see Section 3.7).

The numerical solution of Eq. (3.38) is based on so-called space-time finite volume methods. By space-time methods we refer to methods where the temporal and spacial discretizations are combined rather than separated. Space-time methods are also known as *cell-integrated semi-Lagrangian* schemes. As will become clear, such schemes are neither purely semi-Lagrangian, nor Eulerian in the classical sense. They are Eulerian in the sense that the flux of mass through the stationary walls of grid cells is considered. They are, however, semi-Lagrangian in the sense that trajectory calculations are needed for flux computation. In the literature such schemes are sometimes termed *Flux Form Semi-Lagrangian (FFSL)*. The specific implementation in ICON partly builds upon work by Lauritzen et al. (2010, 2011a), Harris and Lauritzen (2010), Skamarock and Menchaca (2010), Miura (2007) for the horizontal and Colella and Woodward (1984), Zerroukat et al. (2006) for the vertical.

As we are dealing with a Finite Volume (FV) discretization, it is worth noting that in the following all scalar variables ψ , whose storage location is at the triangle cell circumcenter, are interpreted as cell averages rather than point values, i.e.

$$\bar{\psi}_i^n = \frac{1}{\Delta V_i} \iiint_{V_i} \psi(x, y, z, t^n) dV ,$$

with ΔV_i denoting the volume of the i^{th} prismatic cell (the so-called control volume). Here and in the reminder of this Section, the overbar denotes volume averages rather than Reynolds averages.

3.6.1. Directional Splitting

By integrating the continuity equation (3.38) in space over a prismatic grid cell and in time over the time step Δt , a solution to (3.38) can formally be written as

$$\overline{\rho q}_{i,k}^{n+1} = \overline{\rho q}_{i,k}^n + \Delta t [\mathcal{H}(\overline{q}^n) + \mathcal{V}(\overline{q}^n)] , \quad (3.39)$$

where \mathcal{H} and \mathcal{V} denote the horizontal and vertical transport operators acting on q^n , and $\overline{\rho q}_{i,k}^{n+1}$ denoting the updated cell averaged partial density of constituent k at the time t^{n+1} (see Reinert, 2020).

Instead of solving this somewhat unwieldy equation in one sweep, a fractional step approach is taken in ICON such that separate equations for horizontal and vertical transport are solved consecutively. Of course, replacing equation (3.39) by some approximation involving the two subproblems

$$\begin{aligned} \overline{\rho q}_{i,k}^* &= \overline{\rho q}_{i,k}^\alpha + \Delta t \mathcal{V}(\overline{q}^\beta) \\ \overline{\rho q}_{i,k}^{**} &= \overline{\rho q}_{i,k}^\gamma + \Delta t \mathcal{H}(\overline{q}^\delta) \end{aligned}$$

will inevitably result in a residual error. This error is known as the *splitting error*. On default, the following approximation to (3.39) is used:

$$\overline{\rho q}_{i,k}^* = \overline{\rho q}_{i,k}^n + \Delta t \mathcal{V}(\overline{q}^n) \quad (3.40)$$

$$\overline{\rho q}_{i,k}^{n+1} = \overline{\rho q}_{i,k}^* + \Delta t \mathcal{H}(\overline{q}^*) \quad (3.41)$$

In order to maintain $\mathcal{O}[\Delta t^2]$ accuracy, the order of the operators is reversed on alternate time steps. This might be regarded as a poor man's *Strang splitting* (Strang, 1968). Full Strang-splitting of the form $[\mathcal{V}(\Delta t/2)][\mathcal{H}(\Delta t)][\mathcal{V}(\Delta t/2)]$ has also been tested during the implementation phase. Except for being more expensive (the vertical operator is called twice per time step) no significant impact on the model results has been noted.

A shortcoming of the splitting (3.40), (3.41) is that it does not preserve an initially uniform tracer field (e.g. $q(x, y, z, t_0) = 1$) in a deformational flow since there is not enough information available to correctly do the conversion $\overline{\rho q}^* \rightarrow \overline{q}^*$. Tempting candidates for this conversion might be $\overline{\rho}^n$ or $\overline{\rho}^{n+1}$ as they are readily available from ICON's dynamical core. Any such attempt, however, will not preserve an initially uniform tracer field. In order to do so, it is necessary to keep track of the changes in partial density ρq that are solely a result of mass convergence/divergence in the directions of splitting. Therefore we follow the method of Easter (1993) wherein the air mass continuity equation (3.8) is simultaneously reintegrated in the same split manner as the continuity equation for tracer mass.

$$\begin{aligned} \overline{\rho q}_{i,k}^* &= \overline{\rho q}_{i,k}^n + \Delta t \mathcal{V}(\overline{q}^n) \\ \overline{\rho}_{i,k}^* &= \overline{\rho}_{i,k}^n + \Delta t \mathcal{V}(1) \\ \overline{q}_{i,k}^* &= \frac{\overline{\rho q}_{i,k}^*}{\overline{\rho}_{i,k}^*} \end{aligned} \quad (3.42)$$

$$\begin{aligned}
\overline{\rho q}_{i,k}^{n+1} &= \overline{\rho q}_{i,k}^* + \Delta t \mathcal{H}(\bar{q}^*) \\
\bar{\rho}_{i,k}^{n+1} &= \bar{\rho}_{i,k}^* + \Delta t \mathcal{H}(1) \\
\bar{q}_{i,k}^{n+1} &= \frac{\overline{\rho q}_{i,k}^{n+1}}{\bar{\rho}_{i,k}^{n+1}}
\end{aligned} \tag{3.43}$$

Changes in partial density solely due to mass convergence/divergence are corrected for in equations (3.42) and (3.43). The key point here is that the intermediate density $\bar{\rho}^*$ rather than $\bar{\rho}^{n+1}$ or $\bar{\rho}^n$ is used to recover the mass fraction \bar{q}^* in (3.42). The re-integration of (3.7) (second and fifth equation above) is rather straightforward, as it relies on pre-computed mass fluxes provided by the dynamical core.

3.6.2. Horizontal Transport

A rigorous derivation of the horizontal transport operator $\mathcal{H}(\bar{q})$ is beyond the scope of this document. We will merely concentrate on the general concept and illustrate graphically how the scheme works. The horizontal transport scheme belongs to the class of so-called **Flux Form Semi-Lagrangian** (FFSL) schemes (Harris and Lauritzen, 2010). In the literature such schemes are sometimes alternatively termed *Incremental remapping schemes* (Lipscomb and Ringler, 2005) or *streamline subgrid integration method* (Yeh, 2007).

Graphical Interpretation

Figure 3.3 provides a graphical interpretation of the FFSL-scheme. Black solid lines show the triangular grid, with thick solid lines highlighting an arbitrary cell with area ΔA_i for which the scheme will be explained. In the following we will refer to this cell as the Eulerian control volume (CV). The basic task is to compute the updated value $\overline{\rho q}_i^{n+1}$ for that cell on the basis of the old values $\overline{\rho q}_i^n$, the cell averages \bar{q}_i , and the velocity fields \mathbf{v}^n and \mathbf{v}^{n+1} .

In order to set the stage, let us first take the Lagrangian viewpoint: Assume that the time dependent velocity field is known analytically such that the trajectories for all the air parcels are known, which terminate at the walls of the Eulerian CV at the new time t^{n+1} . As an example, trajectories for air parcels terminating at the CV vertices at t^{n+1} are depicted as gray lines. Accordingly we know the position of these air parcels at time t^n which we will denote as the departure points. By connecting the departure points we can construct the gray shaded area known as the Lagrangian CV. The latter encompasses all air parcels that are transported into the Eulerian CV (i.e. the grid cell) during the time interval $[t^n, t^{n+1}]$. In a standard semi-Lagrangian scheme the key task is to compute an estimate of the Lagrangian CV (gray shaded), followed by a computation of the total tracer mass contained. Then, the solution $\overline{\rho q}_i^{n+1}$ can easily be deduced from the Lagrangian finite-volume form of the continuity equation (3.38)

$$\overline{\rho q}_i^{n+1} \Delta A_i = \overline{\rho q}_i^n \Delta a_i,$$

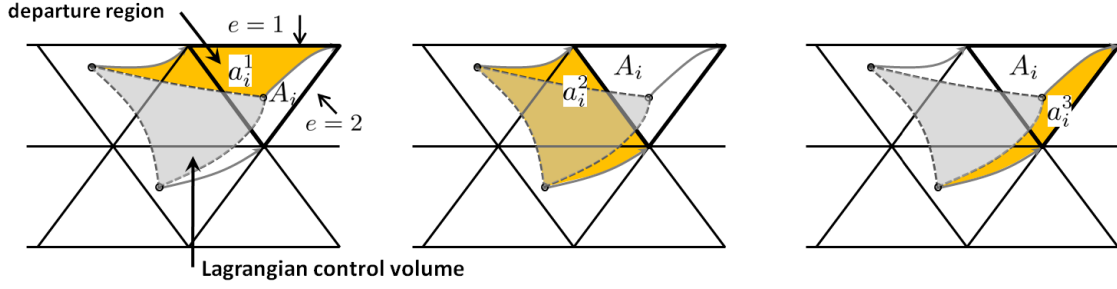


Figure 3.3.: Graphical illustration of the FFSL scheme. Black solid lines show the triangular grid, with thick solid lines highlighting the Eulerian control volume under consideration. Gray area shows the Lagrangian control volume and yellow areas show the flux areas (departure region) for each cell wall.

where ΔA_i and Δa_i denote the area of the Eulerian and Lagrangian CV, respectively (see e.g. [Lauritzen et al., 2011b](#)). $\overline{\rho q}_i^n$ is the average tracer mass over the Lagrangian CV area a_i

$$\overline{\rho q}_i^n = \frac{1}{\Delta a_i} \iint_{a_i} \rho^n(x, y) q^n(x, y) dA.$$

As mentioned previously, the Eulerian rather than Lagrangian viewpoint is taken in ICON. Here we keep track of the flux of mass crossing the Eulerian cell walls. This is where the yellow areas in Figure 3.3 enter the game. We will refer to these as *flux areas* a_e . Since the individual edges of the Lagrangian CV pass through the flux areas during $[t^n, t^{n+1}]$, it is the mass inside the flux areas that is swept across the Eulerian CV walls during one time step. Thus, starting from the tracer mass $\overline{\rho q}_i^n$ in the Eulerian CV and assuming that we know the tracer mass in every flux area, we can compute the updated cell value $\overline{\rho q}_i^{n+1}$.

Mathematically the scheme can be cast into the flux form

$$\overline{\rho q}_i^{n+1} = \overline{\rho q}_i^n - \frac{\Delta t}{\Delta A_i} \sum_{e=1}^{N_e} s_{ie} \Delta l_e \overline{q}^{a_e} \widetilde{F}_e, \quad (3.44)$$

with the flux area average

$$\overline{q}^{a_e} = \frac{1}{\Delta a_e} \iint_{a_e^e} q^n(x, y) da, \quad (3.45)$$

the size of the flux area Δa_e , the edge length Δl_e , and the mass flux crossing the e^{th} cell wall \widetilde{F}_e averaged over Δt . In addition, $s_i^e = \pm 1$ distinguishes inward and outward directed fluxes. The mass flux \widetilde{F}_e is assumed to be provided by the dynamical core.

We point out that the Eulerian viewpoint is fully equivalent to the Lagrangian viewpoint. It can be shown ([Lauritzen et al., 2011b](#)) that all areas involved in our quasi-Eulerian approach (i.e. the Eulerian CV and the flux areas) sum up to the Lagrangian CV.

Basic Algorithm

The numerical algorithm which solves Eq. (3.44) for a single Eulerian CV proceeds in 4 major stages:

1. The flux area a_i^e for each cell wall is computed by means of backward trajectories.
2. For each Eulerian CV the unknown tracer subgrid distribution $q(x, y)^n$ is reconstructed from the known cell averages \bar{q}_i^n of the CV itself and surrounding cells. Several polynomial reconstructions, from linear to cubic, are available.
3. The flux area average \bar{q}^{ae} is estimated by numerical evaluation of the integral (3.45). In particular, Gauss-Legendre quadrature is applied to integrate the subgrid distribution $q(x, y)^n$ over the flux area a_e .
4. Under the assumption that the mass flux \tilde{F}_e is known, the sum on the r.h.s. of Eq. (3.44) can be evaluated, which leads to the solution $\bar{\rho q}_i^{n+1}$.

3.6.3. Vertical Transport

A rigorous derivation of the vertical transport operator $\mathcal{V}(q)$ is beyond the scope of this document. As for the horizontal operator $\mathcal{H}(q)$ we will concentrate on the basic concept. For more details we refer to Reinert (2021).

Piecewise Parabolic Method (PPM)

The Piecewise Parabolic Method (PPM) (Colella and Woodward, 1984) is a mass conserving finite volume method which is widely used in geophysical fluid dynamics. The unknown subgrid distribution of any 1D scalar field $q(z)$ is approximated cell-wise by a piecewise parabolic function, which is constructed in a way that it is continuous at cell interfaces. The construction of the parabolas is based on the known cell averages

$$\bar{q}_k = \frac{1}{\Delta z_k} \int_{z_{k+1/2}}^{z_{k-1/2}} q(z) \, dz, \quad (3.46)$$

with Δz_k denoting the vertical cell thickness. The PPM scheme bears some conceptual resemblance to the horizontal FFSL transport scheme. The basic concept is depicted in Figure 3.4 and described below.

Step 1: The subgrid distribution $q(\zeta, k)$ is reconstructed cell-wise in a vertical column by using the parabolic interpolant

$$q(\zeta, k) = a_0 + a_1\zeta + a_2\zeta^2, \quad \text{with} \quad \zeta = \frac{z - z_{k+1/2}}{\Delta z_k}. \quad (3.47)$$

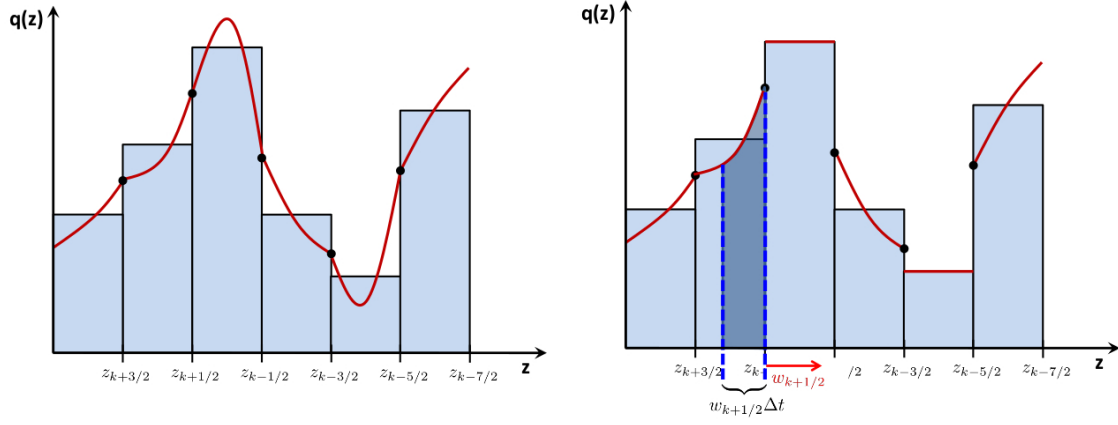


Figure 3.4.: The Piecewise Parabolic Method (PPM). Left: The unknown subgrid distribution $q(z)$ is approximated by piecewise parabolic interpolants, which are \mathcal{C}^0 continuous at cell faces. Right: The polynomial reconstruction is filtered (optional) to render the scheme monotonous. Integration step: The sub-grid distribution is integrated over the “area” $w\Delta t$ (dark blue) in order to determine the mass which enters the k^{th} cell during Δt .

ζ is a dimensionless coordinate which is 1 at the grid cell top and 0 at its bottom. The unknown coefficients a_i in (3.47) are derived from the three constraints

$$\begin{aligned} \int_0^1 q(\zeta, k) d\zeta &= \bar{q}_k, \\ q(\zeta = 1, k) &= q_u = q_{k-1/2}, \\ q(\zeta = 0, k) &= q_l = q_{k+1/2}, \end{aligned} \quad (3.48)$$

which, expressed in words, state that the polynomial must be mass conserving, and that the polynomial equals q_u and q_l at the upper and lower cell face, respectively. q_u and q_l are shared between vertical adjacent cells, by which continuity of the reconstruction across cells is enforced.

The parabolic interpolant (3.47) can finally be written as a function of the grid scale variables \bar{q}_k , $q_{k+1/2}$, $q_{k-1/2}$

$$q(\zeta, k) = \bar{q}_k - \Delta q_k \left(\frac{1}{2} - \zeta \right) - a_{6,k} \left(\frac{1}{6} - \zeta + \zeta^2 \right), \quad (3.49)$$

with

$$\begin{aligned} \Delta q_k &= q_{k-1/2} - q_{k+1/2} \\ a_{6,k} &= 6\bar{q}_k - 3q_{k-1/2} - 3q_{k+1/2}. \end{aligned}$$

The accuracy of the parabolic interpolant (3.49) strongly depends on the accuracy of the edge values $q_{k\pm 1/2}$. In order for the interpolant to reconstruct a parabola exactly, edge values must be at least third-order accurate. Here we follow [Colella and Woodward \(1984\)](#) and compute fourth-order edge value estimates, as this turned out to be beneficial to the overall accuracy of the scheme ([Lauritzen et al. \(2011b\)](#), p. 228)).

An in-depth derivation of the edge values $q_{k\pm 1/2}$ is beyond the scope of this tutorial. We note that $q_{k+1/2}$ is computed from a cubic polynomial $c(z)$ evaluated at $z_{k+1/2}$. The cubic polynomial is constructed from the constraint that it must be mass conserving in each of the four cells surrounding half level $z_{k+1/2}$ (likewise for $q_{k-1/2}$, see e.g. [Zerroukat et al. \(2002\)](#)).

Step 2: As can be seen from Figure 3.4, it is not guaranteed that the reconstruction preserves monotonicity or positive definiteness, especially near strong gradients. The scheme can optionally be made (semi-) monotonic or positive definite by filtering the polynomial reconstruction. The effect of a monotonic filter on the reconstructed parabolas is schematically depicted in Figure 3.4b. The filtering is controlled by the namelist switch `itype_vlimit (transport_nml)`.

Step 3: In a last step, the mass that is swept across the cell wall during Δt is computed by integrating the subgrid distribution $q(\zeta, k)$ over the (upwind) flux area.

$$F_{k-1/2} = \frac{1}{\Delta t} \int_{z_{k-1/2} - w_{k-1/2}^{n+1/2} \Delta t}^{z_{k-1/2}} \rho(z) q(z) dz, \quad \text{for } w > 0 \quad (3.50)$$

Vividly speaking, an estimate of the flux area can be gained by launching a backward trajectory at the given cell wall. In this 1D scheme, the flux area for the cell wall at $z_{k-1/2}$ is given as $-w_{k-1/2}^{n+1/2} \Delta t$, where w is the time-centered vertical velocity provided by the dynamical core.

For $w > 0$ integration of (3.50) leads to the time-averaged vertical flux

$$F_{k-1/2} = \rho_{k-1/2} w_{k-1/2} \left[\bar{q}_k + \frac{1}{2} \Delta q_k (1 - C_{k-1/2}) - \frac{1}{6} a_{6,k} (1 - 3C_{k-1/2} + 2C_{k-1/2}^2) \right],$$

with the Courant number $C_{k-1/2} = w_{k-1/2} \Delta t / \Delta z_k$.

In its standard version, the PPM scheme has a Courant number limitation of $|C| \leq 1$. It can, however, be extended to larger Courant numbers by splitting the computation of the mass fluxes (3.50) into so-called integer and fractional fluxes ([Lin and Rood, 1996](#)). The PPM implementation in ICON is stable for $|C| \leq 5$.

The method is third-order accurate in space on arbitrary grids and third-order accurate in time for the spacial case of constant velocity w . For the general case of space and time dependent velocity fields $w(z, t)$, the temporal accuracy reduces to first order ([Strassmann et al., 2025](#)).

Parabolic Spline Method (PSM)

As an alternative to PPM, the Parabolic Spline Method ([Zerroukat et al., 2006](#)) is available as well. PSM is similar to PPM, as both rely on piecewise parabolic functions for reconstructing the unknown sub-grid distribution in each cell. PSM, however, differs from

PPM in terms of the edge-value estimate. The edge values are determined by imposing the following additional constraint on the parabola $q(\zeta, k)$ in (3.47)

$$\frac{1}{\Delta z_{k+1}} \frac{dq_{k+1}}{d\zeta} \Big|_{\zeta=1} = \frac{1}{\Delta z_k} \frac{dq_k}{d\zeta} \Big|_{\zeta=0}, \quad (3.51)$$

which states that the parabola's first derivative must be continuous at cell edges. Hence, while the PPM parabolas are continuous at cell edges, the PSM parabolas are even continuously differentiable. The latter turns the piecewise parabolic function into a parabolic spline. From the condition (3.51) an implicit equation system for the unknown edge values $q_{k+1/2}$ can be deduced, which is however beyond the scope of this tutorial (see [Zerroukat et al., 2006](#), [Reinert, 2021](#)).

A comparison of PSM and PPM reconstructions for an arbitrary irregular signal is depicted in Figure 3.5a. The signal is taken from [Zerroukat et al. \(2005\)](#) and is defined on the unit interval $z \in [0, 1]$. It is given in terms of cell averages (black dots) on a 1D grid with constant grid spacing. The solid red line depicts the PSM reconstruction, with red circles showing the reconstructed edge values. The PPM reconstruction is shown in blue.

Both reconstructions result in a third-order accurate and smooth representation of the underlying irregular signal. As expected from the previous discussion, both reconstructions are continuous at cell faces, but exhibit unphysical over and undershoots in the vicinity of strong gradients. Available methods for dealing with spurious over- and undershoots are briefly discussed in Section 3.6.5. While both reconstructions behave similarly in large parts of the domain, the effect of PPM being only continuous becomes apparent at some points. The fact that PPM slopes exhibit discontinuities at cell faces is clearly visible e.g. at $z = 0.15$ and $z = 0.35$. The PSM reconstruction, on the other hand, has continuous slopes throughout the domain, which gives it a more “natural” appearance. The absolute difference between the PSM and PPM reconstruction is shown in Figure 3.5b. Largest differences occur close to cell edges.

3.6.4. Reduced Calling Frequency

The prognostic equations for momentum, energy and mass (3.4)–(3.7) are strongly coupled and support the propagation of fast-moving waves, such as sound waves. In particular, the density ρ resulting from the continuity equation for air (3.7) impacts air pressure (via the equation of state (3.10)) and feeds back on the velocity field, which in turn is needed to solve (3.7) for ρ . The fast-moving sound waves impose strict numerical stability limits in terms of a maximum Courant number (assuming that explicit time-stepping is applied). It follows that the equations for momentum, energy and mass must be solved with a sufficiently small time step, obeying the maximum Courant number constraint for sound wave propagation.

On the other hand, the prognostic equation for tracer mass (3.8) is only very weakly coupled to the momentum, energy and mass equations. The evolution of tracers depends on the air velocity \mathbf{v} and density ρ , but tracers do not influence ρ and \mathbf{v} in any significant way. It follows that the tracer mass continuity equation is lacking fast wave modes, which translates to less severe time step restrictions.

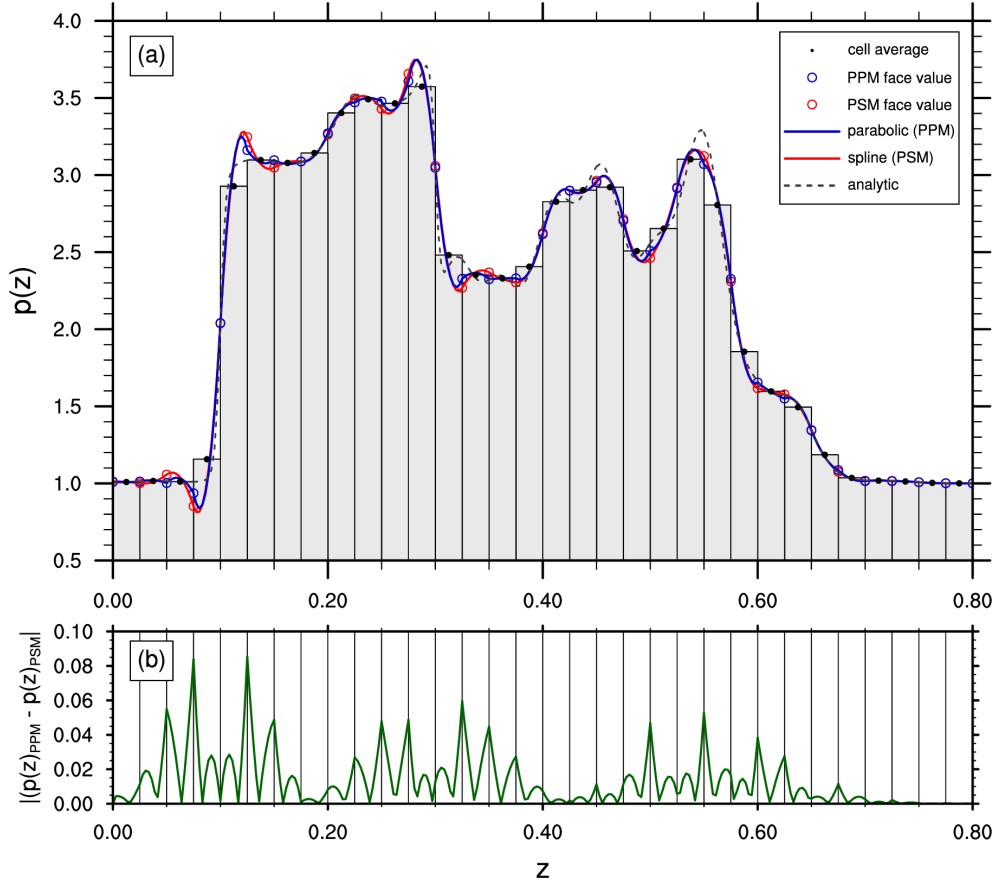


Figure 3.5.: (a) Reconstruction of an irregular 1D signal (gray-dashed) with piecewise parabolics (blue) and piecewise parabolic splines (red) from known cell averages (black dots) on an equidistant grid. The interpolated edge values are shown by blue and red circles, respectively. (b) Absolute difference between the piecewise parabolics and piecewise parabolic splines.

Given the large number of tracers in state of the art climate and NWP models, significant computational cost can be saved by sub-cycling the equations for momentum, energy and air mass with respect to the tracer equations. Stated in another way, the tracer equations can be integrated with a much larger time step. In doing so, care must be taken to maintain tracer-mass consistency.

In ICON, the continuity equation for air (as well as the equations for momentum and energy) are by default sub-cycled five times within one time step of the tracer mass continuity equations (see also Section 3.7.1). In order to maintain tracer-mass consistency, the mass flux \tilde{F}_e entering (3.44) is averaged over the sub-cycling steps according to

$$\tilde{F}_e = \sum_{k=1}^m \frac{1}{m} \tilde{F}_e^{k/m},$$

where m is the number of substeps, and $\tilde{F}_e^{k/m}$ is the average mass flux for the k th substep from $t^{n+(k-1)/m}$ to $t^{n+k/m}$ which follows from the solution of the continuity equation for air mass (3.7).

3.6.5. Some Practical Advice

Here we give some practical guidance on how to configure the tracer transport for standard NWP runs. The most important namelist parameters are discussed along with recommended settings.

The main switch for activating tracer transport is `ltransport` (`run_nml`). Except for specific idealized test cases (see Chapter 4) this switch should generally be set to `.TRUE..` The namelist `run_nml` contains a second relevant parameter termed `ntracer` which is meant for specifying the total number of tracers that shall be advected. We note, however, that this parameter is important for idealized cases only. In real case runs, ICON takes care of initializing the correct number of tracers based on the selected physics packages. E.g. when selecting the one-moment microphysics scheme without graupel (`inwp_gscp=1`), the number of tracers is automatically set to `ntracer=5`.

The namelist `transport_nml` contains additional parameters for selecting the transport scheme and the type of limiter. This can be done individually for each tracer, for horizontal and vertical directions.

`ihadv_tracer` (namelist `transport_nml`, list of Integer values)

Comma separated list of integer values, specifying the type of the horizontal transport scheme. The i^{th} entry corresponds to the i^{th} tracer in ICON's internal tracer list. Most relevant options are

- 1 1st order upwind
- 2 MIURA (Miura (2007))-type with linear reconstruction)
- 3 MIURA3 (Miura (2007))-type with cubic reconstruction)
- 4 FFSL with quadratic or cubic reconstruction (depends on `lsq_high_ord` (`interpol_nml`))
- 5 hybrid MIURA3/FFSL with quadratic or cubic reconstruction
- x2 Sub-cycling versions of MIURA ($x = 2$), MIURA3 ($x = 3$), FFSL ($x = 4$) and hybrid MIURA3/FFSL ($x = 5$).

Sub-cycling means that the integration from t^n to t^{n+1} is split into substeps to meet the stability requirements. By default 3 substeps are used, see `nadv_substeps` (`transport_nml`). Sub-cycling is only applied above a certain height defined by `hbot_qvsubstep` (`nonhydrostatic_nml`), see Section 3.8.12. Above that height the MIURA scheme (linear reconstruction) is used, irrespective of the settings for `ihadv_tracer`.

FFSL and MIURA3 differ w.r.t. the way the integration over the flux area is performed. FFSL can cope with slightly larger Courant numbers while being somewhat more expensive. Option 5 tries to combine the improved stability of FFSL with the speed of MIURA3 by calling FFSL only for those edges for which the horizontal Courant number exceeds a threshold.

`ivadv_tracer` (namelist `transport_nml`, list of Integer values)

Comma separated list of integer values, specifying the type of the vertical transport scheme. The i^{th} entry corresponds to the i^{th} tracer in ICON's internal tracer list. Most relevant options are

- 1 1st order upwind
- 2 Parabolic Spline Method (PSM)
- 3 Piecewise Parabolic Method (PPM)

itype_hlimit (namelist transport_nml, list of Integer values)

Comma separated list of integer values, specifying the type of the horizontal limiter. The i^{th} entry corresponds to the i^{th} tracer in ICON's internal tracer list. Most relevant options are

- 0 no limiter
- 3 monotonic Flux Corrected Transport ([Zalesak, 1979](#))
- 4 positive definite Flux Corrected Transport

itype_vlimit (namelist transport_nml, list of Integer values)

Comma separated list of integer values, specifying the type of the vertical limiter. The i^{th} entry corresponds to the i^{th} tracer in ICON's internal tracer list. Most relevant options are

- 0 no limiter
- 1 semi-monotonic reconstruction filter
- 2 monotonic reconstruction filter
- 3 positive definite Flux Corrected Transport

ivlimit_selective (namelist transport_nml, integer value)

Reduces detrimental effect of the vertical limiter by applying a method for identifying and avoiding spurious limiting of smooth extrema ([Reinert, 2021](#)).

0/1 off/on

Example Settings for a Standard NWP Run

Valid settings for a standard NWP run with one-moment microphysics (5 prognostic water tracers) are depicted in Figure 3.6. The (hardcoded) ordering of tracers in ICON and their tracer IDs are listed in Table 3.3. In order to set the transport scheme and limiter for a tracer with ID = i , the i^{th} entry in the respective namelist parameters must be modified.

This procedure can become unwieldy and error prone if more than a handful of tracers is used. The ART-package ([Schröter et al., 2018](#)) alleviates this problem by providing a more elaborate way of configuring tracers based on XML files. Note, however that the configuration via XML files is restricted to ART-specific tracers, only.

ivadv_tracer, ihadv_tracer: PPM is the method of choice in the vertical direction for all tracers. In horizontal directions, MIURA is used for all tracers except vapor q_v . A somewhat more accurate (and more expensive) scheme is selected for q_v (hybrid MIURA3/FFSL).

If time to solution is not absolutely critical (e.g. for non-operational, scientific, applications), we recommend to select PSM for vertical transport, together with selective vertical

```

! transport_nml: tracer transport -----
&transport_nml
  ihadv_tracer      = 52, 2, 2, 2, 2  ! hor. transport selector

  ivadv_tracer      = 3, 3, 3, 3, 3  ! vert. transport selector

  itype_hlimit      = 3, 4, 4, 4, 4  ! hor. limiter

  itype_vlimit      = 1, 1, 1, 1, 1  ! vert. limiter

  ivlimit_selective = 0                ! selective limiting switched off
/

```

Figure 3.6.: Example namelist settings for tracer transport in a standard NWP run with one-moment microphysics without graupel (i.e. 5 tracers q_v , q_c , q_i , q_r , q_s).

limiting (`ivadv_tracer=2`, `ivlimit_selective=1`), as this will slightly improve the accuracy of vertical transport with only marginal computational overhead.

One might wonder why sub-cycling is activated only for q_v . The reason is that with standard NWP settings q_v is the only tracer which gets transported all the way up to the model top, where the highest wind speeds are typically encountered. For all other water tracers transport is switched off above a certain height defined by `htop_moist_proc(nonhydrostatic_nml)` (typically around 18 km) such that sub-cycling is not strictly required (see Section 3.8.12).

Note that sub-cycling must be activated for q_v . This is crucial for numerical stability!

Furthermore, note that if additional non-water tracers are added (e.g. purely diagnostic passive tracers or chemical tracers), sub-cycling must be activated since `htop_moist_proc` is only effective for water tracers.

itype_hlimit, itype_vlimit: In terms of limiters, the rule of thumb is that at least a positive definite limiter should be used for all water tracers. Otherwise numerical instabilities will occur due to negative water concentrations. For q_v it is advisable to use a more stringent (albeit more expensive) monotonic limiter in order to reduce spurious condensation/evaporation emerging from nonphysical over-/undershoots in q_v .

3.7. Physics-Dynamics Coupling

Models of the atmosphere are assemblies of individual components based on mathematical conservation laws, and describe important atmospheric processes, such as advection, mixing, radiation and clouds. It is common among modelers to separate these components into ‘*the dynamical core*’ (in short ‘*dycore*’) and ‘*the physics*’ (or ‘parameterizations’), noting thereby that this separation might not be unique. A widely accepted definition of these terms is provided by Thuburn (2008):

Tracer ID	1	2	3	4	5	6	7
Tracer Name	water vapour q_v	cloud water q_c	cloud ice q_i	rain water q_r	snow q_s	graupel q_g	hail q_h

Table 3.3.: Ordering of water tracers in ICON. The tracer ID indicates the position within ICON’s internal tracer data structure. In order to specify transport settings for a tracer with $ID = i$, the i^{th} entry in the respective namelist parameter must be set (see `ihadv_tracer`, `ivadv_tracer`, `itype_hlimit`, `itype_vlimit`). Note that this table is incomplete in the sense that additional water tracers for two-moment microphysics schemes (like number concentrations) are omitted. These indices can be taken directly from the source code if required.

The formulation of a numerical model of the atmosphere is usually considered to be made up of a dynamical core, and some parameterizations. Roughly speaking, the dynamical core solves the governing fluid and thermodynamic equations on resolved scales, while the parameterizations represent subgrid scale processes and other processes not included in the dynamical core such as radiative transfer. Here, no attempt is made to give a precise definition of ‘dynamical core’ because, as discussed below, there are some open questions concerning exactly which terms and which processes should be included in a dynamical core.

We follow [Rood \(2010\)](#), in order to exemplify the component separation. The conservation equation for a scalar Ψ (such as temperature or water vapor) may be written in the schematic form

$$\frac{\partial \Psi}{\partial t} + \nabla \cdot (\mathbf{u}\Psi) + F = M + P - L\Psi ,$$

where \mathbf{u} is the velocity vector, M represents sub-grid scale mixing, and P , L represent production and loss rates, respectively. The term F accounts for numerical filters and fixers, which are needed to stabilize the numerical discretization method. The divergence term on the left is identified with the dynamical core, as it is related to the resolved flow. The same holds for the set of filters and fixers F , as these are tightly coupled to the numerical methods and grids chosen. On the other hand, the terms P and L are identified with the ‘the physics’, since the underlying physical processes act on scales that are usually smaller than the resolved scales. For the mixing term M the attribution is less clear. It represents adiabatic dynamical processes, such as boundary layer turbulence and drag due to breaking gravity waves, on scales that are not explicitly resolvable by the dynamical core. Based on their nature (dynamical processes) these processes may be attributed to the dynamical core. However, based on their spatial and temporal scales, they might be attributed to the physics. The separation of the dynamical core and the physics is becoming even less clear for high resolution models. With increasing spatio-temporal resolution more and more gravity waves become explicitly resolved by the dynamical core, which are already accounted for by the gravity wave parameterization. Similar *grey zones*,

$$\begin{aligned}
\frac{\partial \hat{v}_n}{\partial t} + \frac{\partial \hat{K}_h}{\partial n} + (\hat{\zeta} + f)\hat{v}_t + \hat{w} \frac{\partial \hat{v}_n}{\partial z} + c_{pd} \hat{\theta}_v \frac{\partial \pi'}{\partial n} + F_{v_n} &= -\frac{1}{\bar{\rho}} (\nabla_h \cdot \overline{\rho \mathbf{v}'' \mathbf{v}''}) \cdot \mathbf{e}_n \\
\frac{\partial \hat{w}}{\partial t} + \hat{\mathbf{v}}_h \cdot \nabla \hat{w} + \hat{w} \frac{\partial \hat{w}}{\partial z} + c_{pd} \left(\hat{\theta}_v \frac{\partial \pi'}{\partial z} + \theta'_v \frac{d\pi_0}{dz} \right) + F_w &= -\frac{1}{\bar{\rho}} \frac{\partial}{\partial z} \overline{\rho \mathbf{v}'' \mathbf{v}''} \\
\frac{c_{vd} c_{pd}}{R_d} \bar{\rho} \hat{\theta}_v \frac{\partial \bar{\pi}}{\partial t} + c_{pd} \bar{\pi} \nabla \cdot (\bar{\rho} \hat{\mathbf{v}} \hat{\theta}_v) - c_{pd} \bar{\pi} \bar{\rho} \hat{\theta}_v \chi \nabla \cdot \hat{\mathbf{v}} + F_\pi &= c_{pd} \bar{\pi} \bar{\rho} \bar{Q} \\
\frac{\partial \bar{\rho}}{\partial t} + \nabla \cdot (\bar{\rho} \hat{\mathbf{v}}) &= 0 \\
\frac{\partial \bar{\rho} \hat{q}_k}{\partial t} + \nabla \cdot (\bar{\rho} \hat{q}_k \hat{\mathbf{v}}) &= -\nabla \cdot \left(\bar{J}_k^z \mathbf{k} + \overline{\rho q_k'' \mathbf{v}''} \right) + \bar{\sigma}_k
\end{aligned}$$

Figure 3.7.: Separation of the governing equations into ‘the dynamical core’ and ‘the physics’.

in which processes are partly resolved and partly parameterized, exist for the convective scale and the turbulent scale.

Despite these ambiguities, a separation between the dynamical core and the physics is of large practical use, as it allows for a separation of concerns among model developers. From a design perspective, it eases the development of efficient and maintainable code.

In Figure 3.7 the set of governing equations (3.4)-(3.8) is repeated, in order to exemplify the separation into dynamics and physics for the ICON model. Please note that the pressure gradient is reformulated in perturbation form (see Section 3.3), which is the form implemented in the code. The terms F_{v_n} , F_w , F_π have been added to formally account for numerical filters and fixers in the discrete form of the governing equations. In the presented continuous form, these terms are equal to zero. Components that are attributed to the dynamical core (physics) are highlighted in red (blue). The set of NWP parameterizations which account for the blue terms are described in Section 3.8.

3.7.1. ICON Time-Stepping

For efficiency reasons, different integration time steps are applied depending on the process under consideration. In ICON, the following time steps have to be distinguished:

Δt	the basic time step specified via namelist variable <code>dtime</code> , which is used for tracer transport, numerical diffusion and the fast-physics parameterizations.
$\Delta \tau$	the short time step used within the dynamical core; the ratio between Δt and $\Delta \tau$ is specified via the namelist variable <code>ndyn_substeps</code> (namelist <code>nonhydrostatic_nml</code> , number of dynamics substeps), which has a default value of 5.
$\Delta t_{i,slow_physics}$	the process dependent slow physics time steps; they should be integer multiples of Δt and are rounded up automatically if they are not.

An illustration of the relationship between the time steps can be found in Figure 3.8.

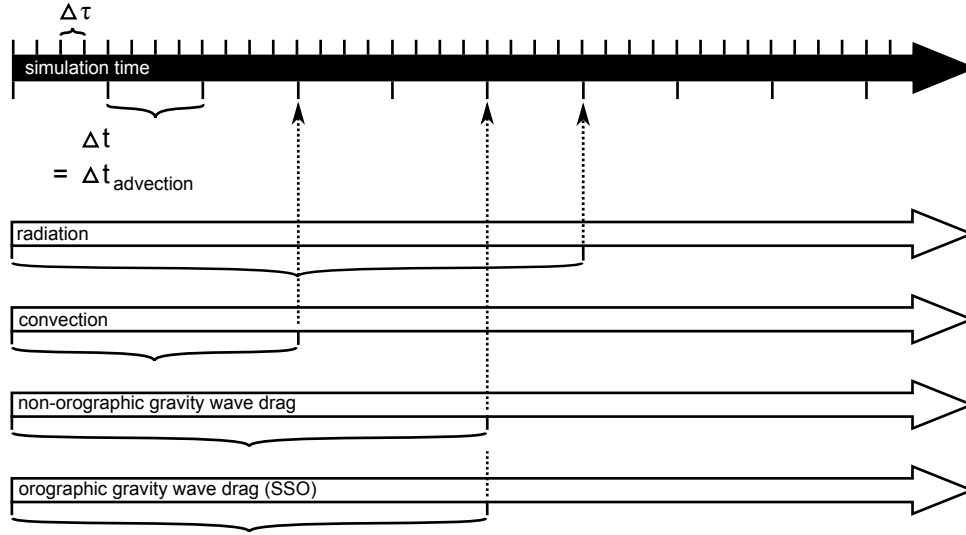


Figure 3.8.: ICON internal time stepping. Sub-cycling of dynamics with respect to transport, fast-physics, and slow-physics. Δt denotes the time step for transport and fast physics and $\Delta\tau$ denotes the short time step of the dynamical core. The time step for slow-physics can be chosen individually for each process.

ICON solves the fully compressible nonhydrostatic Navier-Stokes equations using a time stepping scheme that is explicit except for the terms describing vertical sound wave propagation (see Section 3.5). Thus, the maximum allowable time step $\Delta\tau$ for solving the momentum, continuity and thermodynamic equations is determined by the fastest wave in the system – the sound waves. As a rule of thumb, the maximum dynamics time step can be computed as

$$\Delta\tau = 1.8 \cdot 10^{-3} \overline{\Delta x} \frac{\text{s}}{\text{m}}, \quad (3.52)$$

where $\overline{\Delta x}$ is the effective horizontal mesh size in meters (see Equation (2.1)). This implies that the namelist variable `dtime` should have a value of

$$\Delta t = 9 \cdot 10^{-3} \overline{\Delta x} \frac{\text{s}}{\text{m}},$$

unless `ndyn_substeps` is set to a non-default value.



Historical remark: Note that historically, $\Delta\tau$ rather than Δt was used as basic control variable specified in the namelist, as appears logical from the fact that a universal rule for the length of the time step exists for $\Delta\tau$ only. This was changed shortly before the operational introduction of ICON in 2015 because it turned out that an adaptive reduction of $\Delta\tau$ is needed in rare cases with very large orographic gravity waves in order to avoid numerical instabilities. To avoid interferences with the output time control, the long time step Δt was then taken to be the basic control variable, which always remains unchanged

during a model integration. The adaptive reduction of $\Delta\tau$ is now accomplished by increasing the time step ratio `ndyn_substeps` automatically up to a value of 12, if the Courant number for vertical or horizontal advection grows too large.

Additional time step restrictions for Δt arise from the numerical stability of the horizontal transport scheme and the physics parameterizations, in particular due to the explicit coupling between the turbulent vertical diffusion and the surface scheme. Experience shows that Δt should not significantly exceed 1000 s, which becomes relevant when Δx is larger than about 125 km.

Even longer time steps than Δt can be used for the so-called slow physics parameterizations, i.e. radiation, convection, non-orographic gravity wave drag, and orographic gravity wave drag. These parameterizations provide tendencies to the dynamical core, allowing them to be called individually at user-specified time steps. The related namelist switches are `dt_rad`, `dt_conv`, `dt_gwd` and `dt_sso` in `nwp_phy_nml`. If the slow physics time step is not a multiple of the advective time step, it is automatically rounded up to the next advective time step. A further recommendation is that `dt_rad` should be an integer multiple of `dt_conv`, such that radiation and convection are called at the same time¹. The time-splitting is schematically depicted in Figure 3.8.

3.7.2. Fast and Slow Processes

For efficiency reasons, a distinction is made between so-called *fast* physics processes, whose time scale is comparable or shorter than the model time step, and *slow* physics processes whose time scale is considered slow compared to the model time step.

Fast processes are calculated at every physics time step Δt and are treated with time splitting (also known as sequential-update split) which means that (with exceptions noted below) they act on an atmospheric state that has already been updated by the dynamical core, horizontal diffusion and the tracer transport scheme. Each process then sequentially updates the atmospheric variables and passes a new state to the subsequent parameterization.

The calling sequence is surface transfer scheme \rightarrow saturation adjustment \rightarrow land-surface scheme \rightarrow boundary-layer / turbulent vertical diffusion scheme \rightarrow microphysics scheme, and again saturation adjustment, in order to ensure that vapor and liquid phase are in equilibrium before entering the slow physics parameterizations. The exceptions from the above-mentioned sequential splitting are the surface transfer scheme and the land-surface scheme. Both take the input at the ‘old’ time level because the surface variables are not updated in the dynamical core and the surface transfer coefficients and fluxes would be calculated from inconsistent time levels otherwise. The coupling strategy is schematically depicted in Figure 3.9.

¹This behavior is automatically enforced in the current model version.

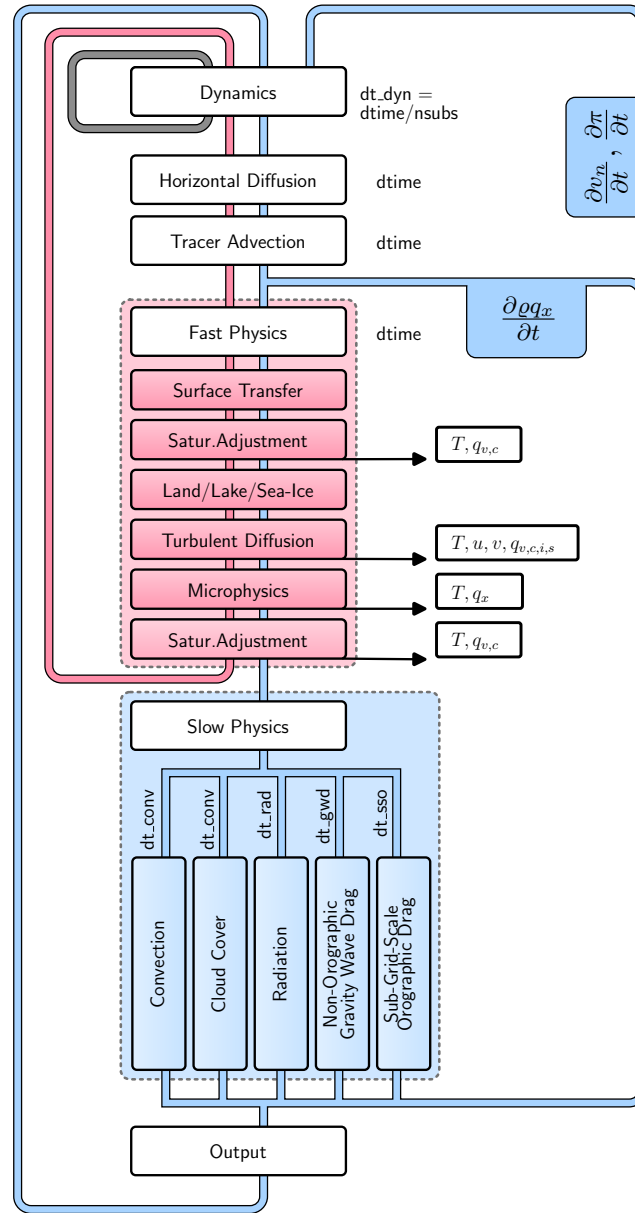


Figure 3.9.: Coupling of the dynamical core and the NWP physics package. Processes declared as fast (slow) are treated in a time-split (process-split) manner.



Note that in the actual ICON code, the surface transfer scheme is called at the very end of the sequence of fast physics processes rather than at the beginning, as depicted in Figure 3.9. In compensation, the input to the surface transfer scheme is the ‘new’ time level rather than the ‘old’ one claimed in the text. Without proof, we claim that this is an equivalent implementation of the more natural sequence shown in Figure 3.9. The natural sequence would necessitate the allocation of additional memory for storing the ‘old’ atmospheric state, as this state is no longer available after the dynamics update due to the inherent substepping. By calling the surface transfer scheme at the end of the previous

time step, the allocation of additional memory and related data transfer can be avoided.

Slow physics processes are treated in a parallel-split manner, which means that they are stepped forward in time independently of each other, starting from the model state provided by the latest fast physics process. In ICON the processes convection, subgrid-scale cloud cover, radiation, non-orographic gravity wave drag and orographic gravity wave drag are considered being slow. Typically, these processes are integrated with time steps longer than the (fast) physics time step Δt . The slow physics time steps can be specified by the user. The resulting slow physics tendencies $\partial v_n / \partial t$, $\partial T / \partial t$ and $\partial \rho q_x / \partial t$, with $x \in [v, c, i]$, are kept constant between two successive calls of the parameterization, and act as forcing terms in the prognostic equations for v_n (3.4), π (3.6), and ρq_x (3.8). The time integration of (3.4) and (3.6) takes place in the dynamical core, while the integration of the water mass continuity equations (3.8) takes place in the tracer transport module. One reason for this technically motivated distinction is the fact that the continuity equation for tracers can be integrated with a much larger time step (Δt) compared to the velocity and thermodynamic equation ($\Delta \tau$). The update of v_n and π due to the slow physics tendencies $\partial v_n / \partial t$, $\partial \pi / \partial t$ takes place inside the dynamical core, while the update of ρq_x due to the slow physics tendency $\partial \rho q_x / \partial t$ happens at a later stage, right after the water mass fractions have been transported (Figure 3.9).

3.7.3. Isobaric vs. Isochoric Coupling Strategies

The physics-dynamics coupling in ICON differs from many existing atmospheric models in that it is performed at constant density (volume) rather than constant pressure. This is related to the fact that the total air density ρ is one of the prognostic variables, whereas pressure is only diagnosed for parameterizations needing pressure as input variable. Thus, it is natural to keep ρ constant in the physics-dynamics interface. As a consequence, heating rates arising from latent heat release or radiative flux divergences have to be converted into temperature changes using c_v , the specific heat capacity at constant volume of moist air. Some physics parameterizations inherited from hydrostatic models, in which the physics-dynamics coupling always assumes constant pressure, therefore had to be adapted appropriately.

Moreover, it is important to note that the diagnosed pressure entering into a variety of parameterizations is a hydrostatically integrated pressure rather than a nonhydrostatic pressure derived directly from the prognostic model variables². This is motivated by the fact that the pressure is generally used in physics schemes to calculate the air mass represented by a model layer, and necessitated by the fact that sound waves generated by the saturation adjustment can lead to a local pressure increase with height in very extreme cases, particularly between the lowest and the second lowest model level.

The hydrostatic pressure field p is diagnosed as follows: In a first step, the surface pressure p_s is diagnosed by integrating the hydrostatic equation from the third-lowest full level

²Note that the (surface) pressure available for output is as well the hydrostatically integrated pressure rather than a nonhydrostatic pressure derived directly from the prognostic model variables.

zm_{nlev-2} down to the surface, under the assumption that the virtual temperature T_v is constant within a grid cell

$$\int_{p_{nlev-2}}^{p_s} \frac{1}{p} dp = - \int_{zm_{nlev-2}}^{z_{nlev+1}} \frac{g}{R_d T_v(z)} dz ,$$

leading to

$$p_s \approx p_{nlev-2} \cdot \exp \left[\frac{g}{R_d} \left(\frac{\Delta z_{nlev}}{T_{v, nlev}} + \frac{\Delta z_{nlev-1}}{T_{v, nlev-1}} + \frac{0.5 \Delta z_{nlev-2}}{T_{v, nlev-2}} \right) \right] .$$

In a second step, the pressure at interface levels $p_{ifc, k}$ is derived by integrating the hydrostatic equation bottom-up, again assuming that T_v is cell-wise constant.

$$p_{ifc, k} = p_{ifc, k+1} \exp \left[-\frac{g}{R_d} \frac{\Delta z_k}{T_{v, k}} \right] , \text{ for } k \in [nlev, \dots, 1] \quad (3.53)$$

Thus, the full level pressure p_k is given by

$$\begin{aligned} p_k &= p_{ifc, k+1} \exp \left[-\frac{g}{R_d} \frac{0.5 \Delta z_k}{T_{v, k}} \right] \\ &= p_{ifc, k+1} \exp \left[-\frac{g}{R_d} \frac{\Delta z_k}{T_{v, k}} \right]^{0.5} \\ &= p_{ifc, k+1} \cdot \left(\frac{p_{ifc, k}}{p_{ifc, k+1}} \right)^{0.5} \\ &= \sqrt{p_{ifc, k+1} \cdot p_{ifc, k}} , \text{ for } k \in [nlev, \dots, 1] \end{aligned}$$

where we made use of equation (3.53) for replacing the exponential function.

Another important aspect is related to the fact that physics parameterizations traditionally work on mass points (except for three-dimensional turbulence schemes). While the conversion between different sets of thermodynamic variables is reversible except for numerical truncation errors, the interpolation between velocity points and mass points potentially induces errors. To minimize them, the velocity increments, rather than the full velocities, coming from the turbulence scheme are interpolated back to the velocity points and then added to the prognostic variable v_n .

3.8. ICON NWP-Physics in a Nutshell

An in-depth description of the physical parameterization package for NWP is beyond the scope of this document. However, the following section provides a short introduction to the available parameterizations and references for further reading.

Table 3.4 contains a summary of physical parameterizations available in ICON (NWP-mode). In what follows, an outline (executive summary) of the parameterization schemes is given.

3.8.1. Radiation

Section author

S. Schäfer, DWD Physical Processes Division

Radiation is a crucial component that drives weather and climate from microscopical to global scales: Heating by absorption of radiation and cooling by emission determine local and global temperature and gradients, which in turn drive dynamics and physical processes. While some other physical effects occur locally, radiation travels throughout the depth of the atmosphere, therefore the entire atmospheric column has to be considered when calculating local radiative fluxes. Both visible or shortwave radiation from the sun and thermal or longwave radiation emitted within the Earth system interact with atmospheric gases, aerosols, clouds and the surface.

Radiative transfer models for the atmosphere consist of multiple components: optical property parameterizations for each atmospheric component and the surface, and a radiation solver that calculates how radiation travels through the optical medium. The new radiation scheme ecRad (Hogan and Bozzo, 2018; implementation in ICON: Rieger et al., 2019) allows choices for each component individually, while the radiation scheme itself is chosen with `inwp_radiation=1` for ICON's RRTM radiation scheme and `inwp_radiation=4` for ecRad (Namelist `nwp_phy_nml`). Using ecRad requires the Compiler-Flag `-enable-ecrad` in the config command before compiling, and setting the namelist parameter `ecrad_data_path` (Namelist `radiation_nml`). The necessary files are contained in '`<ICON-directory>/externals/ecrad/data`'. Since both the whole column and multiple spectral wavelengths have to be considered, the radiation calculation has to be simplified to be practical. Thus, radiation schemes for global weather and climate models neglect horizontal radiative transfer and treat only the vertical dimension. Clouds have a particularly strong radiative effect, and can vary on scales smaller than the model grid-boxes, therefore radiation is calculated once for the clear-sky part of each grid-box and once for the cloudy part.

In the spectral dimension, the strongest variability is due to atmospheric gases, which can absorb and emit radiation of particular, sharply defined wavelengths, according to their molecular properties. The optical properties of cloud particles, aerosol and the surface also depend on the wavelength, but vary more slowly. ICON's RRTM radiation scheme and ecRad both use the RRTM (Rapid Radiative Transfer Model Mlawer et al., 1997) gas optics scheme. The spectral range is divided into 30 spectral bands (16 bands in the longwave spectrum and 14 bands in the shortwave spectrum), and cloud, surface












Process	Scheme	Settings
Radiation	RRTM (<u>R</u> apid <u>R</u> adiative <u>T</u> ransfer <u>M</u> odel) Mlawer et al. (1997) , Barker et al. (2003)	inwp_radiation=1
	 ecRad Hogan and Bozzo (2018)	inwp_radiation=4
Non-orographic gravity wave drag	 Wave dissipation at critical level Orr et al. (2010)	inwp_gwd=1
Sub-grid scale orographic drag	 Lott and Miller scheme Lott and Miller (1997)	inwp_sso=1
Microphysics	 Single-moment scheme Doms et al. (2011) , Seifert (2008)	inwp_gscp=1, 2
	Double-moment scheme Seifert and Beheng (2006)	inwp_gscp=4
	Mixed-phase Spectral Bin Microphysics (SBM) Khain and Sednev (1996) , Khain et al. (2004)	inwp_gscp=8
Convection	 Mass-flux shallow and deep Tiedtke (1989) , Bechtold et al. (2008)	inwp_convection=1
Cloud cover	 Diagnostic PDF <i>M. Köhler et al. (DWD)</i>	inwp_cldcover=1
	All-or-nothing scheme (grid-scale clouds)	inwp_cldcover=5
Turbulent diffusion	 Prognostic TKE (COSMO) Raschendorfer (2001)	inwp_turb=1
	3D Smagorinsky diffusion (for LES) Smagorinsky (1963) , Lilly (1962)	inwp_turb=5
Land	 Tiled TERRA Schrodin and Heise (2001) , Schulz et al. (2016)	inwp_surface=1
	 Flake: Mironov (2008)	llake=.TRUE.
	 Sea-ice: Mironov et al. (2012)	lseaiice=.TRUE.

Table 3.4.: Summary of ICON’s physics parameterizations for NWP, together with the related namelist settings (namelist `nwp_phy_nml`). Parameterizations which are used operationally (at 13 km horizontal grid spacing) are indicated by .

and aerosol optical properties are described within each band. The bands are subdivided into sub-intervals with similar gas properties, termed g-points. This so-called correlated-k method strongly reduces computational costs with an accuracy comparable to spectrally more detailed line-by-line models. Since we only consider up- and downwelling radiation, the optical properties are integrated for all angles within a hemisphere, reducing the optical parameters that are needed to optical depth, single scattering albedo and asymmetry factor (of scattering).

Cloud particle optical properties depend on the amount of water or ice, on the particle size and particle shape. Since cloud particles often have sizes similar to the visible or thermal wavelength ranges, scattering by particles varies strongly according to scattering angle and to the ratio of particle size to wavelength (Mie scattering). For given particle shape and size, this complex function can be approximated numerically. Cloud optical property parameterizations have to make an assumption on cloud particle shape. While liquid water particles are spherical, real ice particles can have a variety of shapes, so that ice shape assumptions are uncertain, and vary between parameterizations. Using these assumptions, the cloud optics parameterization provides optical properties depending on particle size (which is parameterized within ICON) and wavelength. In ecRad, several cloud water and ice optics parameterizations are available (namelist parameters `iliquid_scatter` and `iice_scatter` in `radiation_nml`).

For aerosol, optical properties are directly provided as an input to the radiation scheme. In operational settings, we use fixed global aerosol distributions from climatologies. Similarly, surface optical properties are provided to ICON in an external parameter file, based on satellite observations of the surface.

All of these optical properties are provided to the radiation solver, which calculates reflection, transmission and internal radiation sources in each grid-box and model layer, and the resulting amount of up- and downwelling radiation at each height, for cloudy and clear sky. Vertical overlap of cloudy and clear regions between neighboring layers is parameterized according to overlap assumptions (Hogan and Illingworth, 2000, chosen by namelist parameter `icld_overlap` in `radiation_nml`). Operationally, ICON uses the exponential-random overlap assumption, meaning clouds with clear layers in-between are uncorrelated, while the overlap in continuous clouds decreases exponentially with vertical distance. Since cloud absorption and reflection depend non-linearly on cloud optical depth, the variability of thick and thin cloud in a grid-box also has an effect. Highly variable clouds interact less strongly with radiation than homogeneous clouds that contain the same amount of water. This effect can be parameterized roughly by reducing cloud optical depth to compensate (ICON's RRTM radiation uses a factor of 0.8). In ecRad, the variability is captured by dividing the cloudy region into two or more sub-regions with different cloud optical depths. One method to do this in a numerically efficient way is the Monte Carlo Independent Column Method (McICA, Pincus et al., 2003), which only calculates the radiative transfer for one spectral band in each sub-column. The distribution of the bands over the sub-columns introduces random noise, but no bias. Since this McICA method is comparatively cheap, it is the default method used. However, ecRad also provides a choice of other solvers without random noise: Tripleclouds, and SPARTACUS, which also approximately accounts for sub-grid horizontal transfer. The solver and further namelist parameters specific to ecRad are set in the module `/src/atm_phy_nwp/mo_nwp_ecrad_init` and are described in the ecRad documentation (<https://confluence.ecmwf.int/display/ECRAD>).

From the radiative fluxes, radiative heating and cooling is calculated, which feeds back into dynamics and physics. Despite the simplifying assumptions, radiation is still one of the most expensive parts of the model. Hence, radiation is calculated only on a coarser radiation grid (see Section 3.10) and at a coarse radiation time step `dt_rad` (see Section 3.7.1). However, radiative heating rates are updated more frequently, so that they better represent the diurnal cycle of incoming solar radiation.

3.8.2. Non-orographic gravity wave drag

Section authors

M. Köhler, DWD Physical Processes Division
S. Borchert, DWD Numerical Models Division

All kinds of (synoptic-scale) atmospheric flow structures (e.g., fronts, convection, jet streams) can develop imbalances that force air parcels to oscillate (vertically) and radiate gravity waves (GWs). The interaction of these non-orographically forced GWs with the atmospheric background flow is assumed to be significant in the middle and upper atmosphere, and is consequently of interest for models that cover this region (e.g., the operational global ICON configuration with its model top at 75 km). If the synoptic scale flow (be it resolved or unresolved by the model) would force GWs, whose horizontal and vertical wave lengths would be smaller than the horizontal and vertical grid mesh sizes, they cannot be resolved by the model. But yet this unresolved part of the GW spectrum could have a significant impact on the resolved flow, and is therefore parameterized (see Figure 3.10).

The parameterization implemented in ICON follows the ansatz of Scinocca (2003) and McLandress and Scinocca (2005), which in turn is based on the work of Warner and McIntyre (1996) to which the simplifying assumption of a hydrostatic, non-rotational atmosphere has been applied. The parameters of this scheme have been optimized following Ern et al. (2006). Details can be found in Orr et al. (2010).

The mechanisms of non-orographic GW forcing can be relatively complex and are not completely understood. For this reason the parameterization assumes a constant source of GWs. The amount of GWs radiated by this idealized source is directly proportional to the namelist parameter `tune_gfluxlaun` (`nwp_tuning_nml`). Its default value is 0.0025 Pa.

The scheme computes tendencies of the horizontal wind and the temperature. A detailed description can be found in ECMWF (2018a).

3.8.3. Sub-grid scale orographic drag

Section authors

J.-P. Schulz, DWD Numerical Models Division
M. Köhler, DWD Physical Processes Division
S. Borchert, DWD Numerical Models Division

ICON treats the entire sub-grid scale orographic drag with one model based on the work of Lott and Miller (1997). Other NWP models treat scales smaller than ~ 5 km with different

ansatzes, for example with the help of turbulent orographic form drag formulations of the use of an effective roughness length that determines the conductivity of the surface for momentum fluxes between the atmosphere and the Earth.

The motivation for the implementation of a sub-grid scale orographic drag model into ICON traces back to experience with the COSMO model. When the 7-km EU domain of the operational COSMO model at DWD was expanded in order to cover almost all Europe (see [Schulz, 2006](#)), it turned out that the surface pressure in the model forecasts became systematically biased. In particular, in wintertime high pressure systems of the model atmosphere tended to develop a positive pressure bias, by 1 to 2 hPa after 48 h, low pressure systems a negative bias (“highs too high, lows too low”). At the same time the wind speed tended to be overestimated by up to 1 m s^{-1} throughout the entire troposphere. The wind direction near the surface showed a positive bias.

The combination of these deficiencies led to the hypothesis that in the model there is too little surface drag, causing an underestimation of the cross-isobar flow in the planetary boundary layer. Consequently, the solution would be to increase the surface drag in the model. This may be accomplished, for instance, by introducing an envelope orography ([Wallace et al., 1983](#), [Tibaldi, 1986](#)), but this has unfavorable effects, e.g., for the simulated precipitation. Another option is the inclusion of sub-grid scale orographic (SSO) effects, which were neglected in the COSMO model before. The SSO scheme by [Lott and Miller \(1997\)](#) was selected for this purpose. Its implementation in the COSMO-EU model is described in [Schulz \(2008\)](#), and was later transferred to the ICON model.

The SSO scheme by [Lott and Miller \(1997\)](#) deals explicitly with a low-level flow which is blocked when the sub-grid scale orography is sufficiently high. For this blocked flow separation occurs at the mountain flanks, resulting in a form drag. The upper part of the low-level flow is lead over the orography, while generating gravity waves³. In order to describe the low-level flow behavior in the SSO scheme a non-dimensional height H_n of the sub-grid scale mountain is introduced

$$H_n = \frac{NH}{|U|} = Fr^{-1},$$

where H is the maximum height of the mountain, $|U|$ is the wind speed and N is the Brunt-Väisälä frequency of the incident flow. The latter is defined by

$$N = \sqrt{\frac{g}{\theta} \frac{\partial \theta}{\partial z}},$$

where θ is the potential temperature, g the acceleration of gravity and z the height coordinate. H_n may be also regarded as an inverse Froude number Fr^{-1} of the system “flow round a mountain”.

A small H_n means that there is an unblocked regime, all the flow goes over the mountain and gravity waves (GWs) are forced by the vertical motion of the fluid. A large H_n means that there is a blocked regime, the vertical motion of the fluid is limited and part of the low-level flow goes around the mountain. The SSO scheme requires four external parameters, which are the standard deviation `SSO_STDH`, the anisotropy `SSO_GAMMA`, the slope

³Propagating, coherent structures consisting of buoyancy and inertial oscillations.

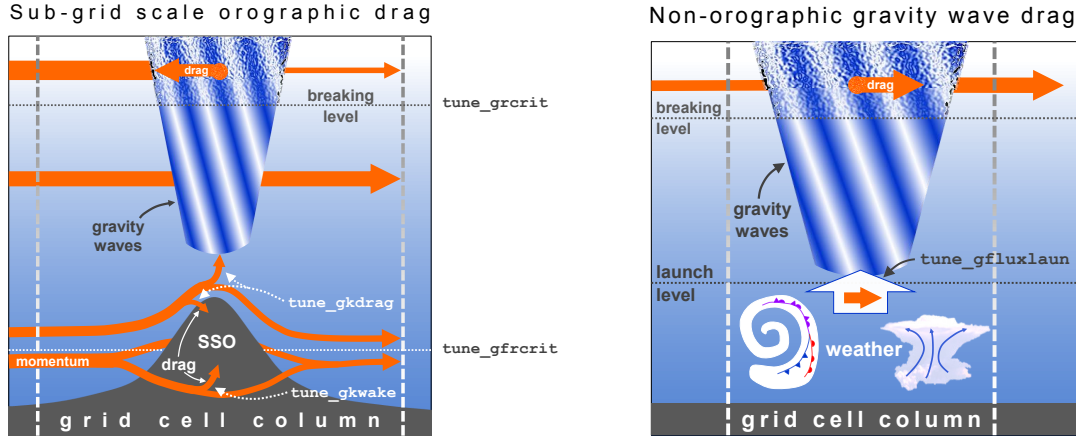


Figure 3.10.: Left: Schematic illustration of the two elements of sub-grid scale orographic drag: First, the low-level blocking, where the horizontal flow (orange) is forced to flow around the sub-grid scale orography (SSO, colored gray). Second, the gravity wave drag on the flow forced to overflow the SSO, and on the flow at higher altitudes, where the radiated gravity waves (white-blue) break. (Following Figure 1 in [Lott and Miller \(1997\)](#).) Right: Schematic illustration of the non-orographic gravity wave drag.

SSO_SIGMA and the geographical orientation SSO_THETA of the sub-grid scale orography. Following [Baines and Palmer \(1990\)](#) these are computed by ExtPar (see Section 2.4.1) from the GLOBE data set ([GLOBE-Task-Team, 1999](#)), which has a resolution of approximately 1 km.

Four tuning parameters in the namelist `nwp_tuning_nml` control the SSO scheme (see Figure 3.10). First, the critical Froude number $Fr_c = \text{tune_gfrcrit}$, which has a twofold effect. The larger its value the higher the likelihood for low-level blocking to occur, since it is activated only where $Fr < Fr_c$. Conversely, where $Fr > Fr_c$ all flow goes over the mountain and the entire stress is associated with GW radiation. In addition, if blocking is active, Fr_c controls the thickness of the blocking layer relative to the mountain height (the larger Fr_c the larger the layer thickness). Its default value is 0.4. Operational simulations with horizontal mesh sizes of 13 km and 40 km use the values 0.333 and 0.425, respectively. Second and third, the magnitudes of the SSO drag and the GW drag are directly proportional to the parameters `tune_gkwake` and `tune_gkdrag`, respectively. The default values are 1.5 and 0.075. Different from this, operational simulations with 13 km horizontal mesh size use `tune_gkwake` = 1.8 and `tune_gkdrag` = 0.09. Finally, the critical Richardson number $Ri_c = \text{tune_grcrit}$ is a control parameter for the onset of GW breaking. If the Richardson number of the resolved atmospheric state plus the unresolved GWs

$$Ri = \frac{N_{\text{tot}}^2}{|\partial \mathbf{u}_{\text{tot}} / \partial z|^2},$$

where \mathbf{u} denotes the horizontal wind vector, falls below Ri_c , the flow configuration becomes unstable and the GWs break (partly). This process is accompanied by a drag effect on the resolved horizontal flow. Combined with the rule of thumb that Ri decreases with height if GWs are present, this means that the larger the value of Ri_c the lower the altitude where the radiated GWs tend to break and exert a drag. The default value is 0.25.

The scheme computes tendencies of the horizontal wind and the temperature. A detailed description can be found in [ECMWF \(2018b\)](#).

3.8.4. Saturation Adjustment

Section author

A. Seifert, DWD Physical Processes Division

In ICON, the atmospheric state which enters the fast physics parameterizations has already been updated by the dynamical core (see Figure 3.9). As a consequence it is no longer guaranteed that vapor and liquid phase are in equilibrium. Hence supersaturated but cloud-free regions might exist, as well as cloudy regions, which are sub- or supersaturated.

Classic saturation adjustment

The aim of a saturation adjustment scheme is to adjust the temperature and water vapor mixing ratio to perfect saturation in supersaturated regions. In subsaturated but cloudy regions, cloud water is evaporated until either saturation is reached or all cloud water is evaporated. From a cloud microphysical point of view the saturation adjustment scheme describes the processes of condensation and evaporation of cloud droplets.

In atmospheric models this adjustment process is usually treated isobarically. In ICON, however, it is treated isochorically, which is a consequence of ICON's physics-dynamics coupling strategy.

We start the description of the saturation adjustment scheme with a short derivation of the temperature equation emphasizing the difference between the enthalpy (constant pressure) and internal energy (constant volume) formulation and including water vapor and liquid water. For a more complete derivation of the prognostic temperature equation see, e.g., [Doms and Baldauf \(2018\)](#).

To derive the prognostic temperature equation one uses the specific internal energy u or the specific enthalpy $h = u + pv$ as a starting point. While the internal energy is appropriate for processes at constant volume, enthalpy would be chosen for processes at constant pressure. The prognostic temperature equation is derived as an expansion of either h or u as a function of temperature T , mass fractions of dry air q_d , water vapor q_v and liquid water q_ℓ and either specific volume v or pressure p .

$$\begin{aligned}\frac{dh}{dt} &= \left. \frac{\partial h}{\partial p} \right|_{T, q_k} \frac{dp}{dt} + \left. \frac{\partial h}{\partial T} \right|_{p, q_k} \frac{dT}{dt} + \sum_{k=d,v,\ell} \left. \frac{\partial h}{\partial q_k} \right|_{p, T} \frac{dq_k}{dt} \\ \frac{du}{dt} &= \left. \frac{\partial u}{\partial v} \right|_{T, q_k} \frac{dv}{dt} + \left. \frac{\partial u}{\partial T} \right|_{v, q_k} \frac{dT}{dt} + \sum_{k=d,v,\ell} \left. \frac{\partial u}{\partial q_k} \right|_{v, T} \frac{dq_k}{dt}\end{aligned}$$

With

$$\begin{aligned}c_p &= \left. \frac{\partial h}{\partial T} \right|_{p, q_k} \\ c_v &= \left. \frac{\partial u}{\partial T} \right|_{v, q_k}\end{aligned}$$

and $\sum q_k = 1$ and no phase transitions involving dry air, and therefore $dq_v/dt = -dq_\ell/dt$, we find:

$$\begin{aligned}\frac{dh}{dt} &= \left. \frac{\partial h}{\partial p} \right|_{T, q_k} \frac{dp}{dt} + c_p \frac{dT}{dt} + (h_\ell - h_v) \frac{dq_\ell}{dt} \\ \frac{du}{dt} &= \left. \frac{\partial u}{\partial v} \right|_{T, q_k} \frac{dv}{dt} + c_v \frac{dT}{dt} + (u_\ell - u_v) \frac{dq_\ell}{dt}\end{aligned}$$

Now we use conservation of energy, i.e. $dh = 0$ or $du = 0$, and constant pressure or constant volume respectively, and find:

$$c_p \frac{dT}{dt} = -(h_\ell - h_v) \frac{dq_\ell}{dt} = L_{\ell v} I_\ell, \quad p = \text{const.} \quad (3.54)$$

$$c_v \frac{dT}{dt} = -(u_\ell - u_v) \frac{dq_\ell}{dt} = \hat{L}_{\ell v} I_\ell, \quad V = \text{const.} \quad (3.55)$$

The $L_{\ell v}$ or $\hat{L}_{\ell v}$ are the latent heats of vaporization and $I_\ell = dq_\ell/dt$ is the condensation rate and also known as the 'Phasenfluss' (in German). The usual latent heat of vaporization given in most textbooks is the $L_{\ell v}$ or the *enthalpy of vaporization*. At 0°C it has a value of $L_{\ell v,0} = 2.501 \times 10^6 \text{ J kg}^{-1}$. The corresponding *internal energy of vaporization* $\hat{L}_{\ell v}$ can be derived from $h_k = u_k + pv_k$ and with $v_\ell \ll v_d$ and $pv_v = R_v T$ we find

$$\hat{L}_{\ell v} = L_{\ell v} - pv_v = L_{\ell v} - R_v T$$

Saturation adjustment is a parameterization of condensation which assumes that vapor and liquid phase are in equilibrium, i.e., by saturation adjustment we want to ensure or realize this equilibrium. Therefore we simply assume that the final state with temperature T_1 has a vapor mass fraction of $q_{v,1} = q_{\text{sat}}(T_1)$ inside the cloud, i.e., whenever $q_\ell > 0$. From the temperature equation at constant volume (3.55) we find

$$c_{vd}(T_1 - T_0) + \hat{L}_{\ell v}(q_{\text{sat}}(T_1) - q_{v,0}) = 0.$$

Note that we have now replaced c_v by the specific heat of dry air at constant volume c_{vd} and, hence, made the usual approximation to neglect the differences in the specific heats between dry air, vapor and liquid water in the temperature term. This is justified because q_v and q_l are usually small.

This equation is solved for T_1 , e.g., using a Newton iteration

$$T_1^{n+1} = T_1^n - \frac{F(T_1^n)}{F'(T_1^n)},$$

with

$$\begin{aligned}F(T_1) &= T_1 - T_0 + \hat{L}_{\ell v} \frac{q_{\text{sat}}(T_1) - q_{v,0}}{c_{vd}} \\ F'(T_1) &= 1 + \frac{\hat{L}_{\ell v}}{c_{vd}} \left. \frac{dq_{\text{sat}}}{dT} \right|_{T=T_1}.\end{aligned}$$

In ICON the iteration is stopped if either $|T_1^{n+1} - T_1^n| < 1\text{E} - 3\text{ K}$, or if the number of iterations exceeds a hard-coded maximum $maxiter = 10$. Regarding the *internal energy of vaporization* \hat{L}_{lv} we actually use

$$\hat{L}_{lv}(T) = L_{lv,0} + (c_{pv} - c_l)(T - T_0) - R_v T.$$

Hence, in addition to the $R_v T$ term arising from the Legendre transformation to internal energy as discussed above, we take into account the linear temperature dependency of the latent heat of vaporization according to Kirchhoff's equation⁴

$$\left. \frac{\partial L_{lv}}{\partial T} \right|_{p, q_k} = \left. \frac{\partial h_v}{\partial T} \right|_{p, q_k} - \left. \frac{\partial h_l}{\partial T} \right|_{p, q_k} = c_{pv} - c_l.$$

This is consistent with the assumption of water vapor as an ideal gas with constant specific heat capacity.

Quasi-equilibrium saturation adjustment

In the previous section, we assumed that water vapor and cloud liquid water are in thermodynamic equilibrium. This classic assumption is made in most cloud-resolving and large-eddy simulation (LES) models. In LES models using moist conserved variables, this assumption is, in fact, implicit in the model equations. In-situ observations confirm that this is a reasonable assumption even in convective clouds, as measured supersaturations rarely exceed 2% (Politovich and Cooper, 1988, Pruppacher and Klett, 1998, Siebert and Shaw, 2017). Kogan and Martin (1994) show that for high cloud condensation nuclei (CCN) concentrations of several hundred per cubic centimeter, the saturation equilibrium assumption is well justified due to shorter phase relaxation times associated with smaller droplet sizes and larger surface area. In this regime, the classic saturation adjustment provides a good approximation for condensation. Significant biases in the condensation rate occur in marine cumulus congestus clouds with low CCN concentrations, which are not explicitly resolved by kilometer-scale NWP models.

While Lebo et al. (2012) argue that supersaturation levels exceeding 10% may occur in convective updrafts, such values have not been confirmed by in-situ observations and are likely limited to idealized or extreme dynamical conditions, such as strong updraft cores prior to substantial droplet activation. Grabowski and Morrison (2021) find supersaturation values around 5% in convective updrafts of their semi-idealized simulations. In a steady-state updraft, the cloud does not approach zero supersaturation but rather a quasi-equilibrium supersaturation, which is a linear function of vertical velocity (Korolev and Mazin, 2003). Thermodynamically, this is known as flux equilibrium.

In ICON, a generalized saturation adjustment has recently been implemented as an adjustment toward a predefined quasi-equilibrium supersaturation, which is an explicit function of vertical velocity, w . This reduces the condensation rate and the corresponding latent heat release, leading to slightly weaker convective updrafts. As a result, the instantaneous precipitation rates of convective systems are reduced and better agree with surface observations (e.g., when using the one-moment graupel scheme, `inwp_gscp=2`). Currently,

⁴see e.g. http://glossary.ametsoc.org/wiki/Kirchhoff's_equation

the quasi-equilibrium supersaturation is hardcoded as $S_{eq} = 0.005 w$, corresponding to 5% supersaturation in a 10 m/s updraft. To allow clouds to form at zero supersaturation initially, S_{eq} is limited by $c_{sat,lim} q_c/q_v$, with a tuning parameter `tune_supsat_limfac` (`nwp_tuning_nml`) that sets $c_{sat,lim}$. The recommended value is $c_{sat,lim} = 2$ for a 2 km grid (R19B07). The parameter $c_{sat,lim}$ is also used to deactivate the generalization to quasi-equilibrium; with the default value of $c_{sat,lim} = 0$, the classic saturation adjustment to zero supersaturation is recovered.

Unfortunately, this implementation remains incomplete. For the two-moment microphysics schemes, `inwp_gscp=4,5,6,7`, which predict both cloud droplet number and mass, the phase relaxation time scale should be a function of the cloud droplet (integral) radius (Politovich and Cooper, 1988, Korolev and Mazin, 2003). Then, the two-moment scheme could, in principle, resolve the differing behavior of low-CCN marine clouds with significant in-cloud supersaturation and high-CCN continental clouds with negligible supersaturation. This capability will be implemented in the future.

3.8.5. Cloud Microphysics

Section author

A. Seifert, DWD Physical Processes Division

Microphysical schemes provide a closed set of equations to calculate the formation and evolution of condensed water in the atmosphere. The most simple schemes predict only the specific mass content of certain hydrometeor categories like cloud water, rain water, cloud ice and snow. This is often adequate, because it is sufficient to describe the hydrological cycle and the surface rain rate, which is the vertical flux of the mass content. Microphysical schemes of this category are called *single-moment schemes*.

In ICON two single-moment schemes are available, one that predicts the categories cloud water, rain water, cloud ice and snow (`inwp_gscp=1` in the namelist `nwp_phy_nml`), and the other that predicts in addition also a graupel category (`inwp_gscp=2`). Graupel forms through the collision of ice or snow particles with supercooled liquid drops, a process called *riming*.

Most microphysical processes depend strongly on particle size and although the mean size is usually correlated with mass content this is not always the case. Schemes that predict also the number concentrations have the advantage that they provide a size information, which is independent of the mass content. Such schemes are called *double-moment schemes*, because both, mass content and number concentration, are statistical moments of the particles size distribution.

ICON does also provide a double-moment microphysics scheme (`inwp_gscp=4,5,6,7`), which predicts the specific mass and number concentrations of cloud water, rain water, cloud ice, snow, graupel and hail. This scheme is most suitable at convection-permitting or convection-resolving scales, i.e., mesh sizes of 3 km and finer. Only on such fine meshes the dynamics is able to resolve the convective updrafts in which graupel and hail form. On coarser grids the use of the double-moment scheme is not recommended.

To predict the evolution of the number concentrations the double-moment scheme includes various parameterizations of nucleation processes and all relevant microphysical

interactions between these hydrometeor categories. Many choices regarding, e.g., cloud condensation and ice nuclei, particle geometries and fall speeds etc. can be set through the namelist `twomom_mcrph_nml`.

To provide a double-moment microphysics scheme for the global ICON model, an extension of the cloud ice scheme (`inwp_gscp=1`) with a two-moment representation of cloud ice is available as `inwp_gscp=3`. This scheme explicitly treats heterogeneous and homogeneous ice nucleation, thereby improving the prediction of ice supersaturated regions. The two-moment cloud ice scheme is also coupled with ART and can make use of prognostic mineral dust as ice nucleating particles. Consequently, it can explicitly represent aerosol-cloud-interaction of ice clouds within the global ICON model. To expose this information to the radiation scheme, coupling of the effective radius is implemented via `icpl_rad_reff=1`.

Last but not least, ICON does include a spectral bin microphysics for mixed-phase clouds (Khain and Sednev, 1996, Khain et al., 2004). It is available as `inwp_gscp=8`. Computationally, this scheme is an order of magnitude more expensive than the two-moment bulk scheme, but it provides a very detailed representation of microphysical processes.

3.8.6. Cumulus Convection

Section author

D. Klocke, DWD Physical Processes Division

Convection is an important process in the atmosphere by contributing to forming the large-scale circulation to local heavy precipitation through thunderstorms. Parameterizations of atmospheric moist convection provide the effect of an ensemble of sub-grid convective clouds on the model column as a function of grid-scale variables. The schemes vertically mix heat, moisture and momentum. They convert available potential energy into kinetic energy and produce precipitation as a result of atmospheric instability.

Three steps are taken. First, it is determined if the grid-scale conditions allow for the occurrence of convection in the column, and a decision is taken if convection is triggered. In the second step, the tendencies of heat, moisture and momentum changes are determined with a *cloud model*, which represents an ascending parcel and its interactions with the environment. Finally the closure decides on the strength of the convection by determining the amount of energy to be converted, which is linked to precipitation amount generated by the convection scheme.

In ICON a bulk mass flux convection scheme is available `inwp_convection=1` (in the namelist `nwp_phy_nml`), which treats three convective cloud types. Only one type - shallow, mid-level or deep convection - can exist at a time in a column, which is decided upon by the trigger function. All three types of convection use a cloud model representing an ascending plume mixing with its environmental air. The cloud base mass flux closure differs between the three convection types, with a CAPE (convective available potential energy) based closure for deep convection, a boundary layer equilibrium closure for shallow convection, and a large-scale omega (vertical velocity in pressure coordinates) based closure for mid-level convection.

The full convection scheme can generally be used for horizontal grids coarser than 5 km, as some resolution dependent adjustments are implemented for grid spacings smaller than

20 km. For convection permitting simulations (1 – 3 km horizontal grid spacing) the largest convective clouds can be resolved by the model and the parameterization parts treating deep and mid-level convection can be switched off (`lshallowconv_only=.TRUE.` (namelist `nwp_phy_nml`)).

The implemented scheme represents a branch of the Tiedtke-Bechtold convection scheme used in the IFS model. For further reading we refer to [Bechtold et al. \(2008\)](#), [Tiedtke \(1989\)](#), [Bechtold \(2017\)](#), [ECMWF \(2017\)](#). Similar to the operational IFS scheme it contains an improved CAPE closure for deep convection ([Bechtold et al., 2014](#)) in order to improve the representation of the diurnal cycle of convection over land (`icapdcycle>0` in the namelist `nwp_phy_nml`). Note that in the operational ICON scheme the modified closure is only applied over land and latitudinally restricted to the tropics (`icapdcycle=3`).



ICON-D2 specifics - grayzone convection:

The option `lgrayzone_deepconv=.TRUE.` (namelist `nwp_phy_nml`) provides a NWP-specific tuning of the deep convection scheme for DWD's ICON-D2 configuration in order to reduce the activity of the convection scheme to just a bit more than pure shallow convection. It includes an increased entrainment rate for parcel ascent calculations and a modified CAPE closure in order to suppress widespread convective drizzle. Recommended for limited area applications in midlatitudes if the model overpredicts high convective precipitation intensities with pure shallow convection. Requires `lshallowconv_only=.FALSE.`

Don't use in the tropics!

3.8.7. Cloud Cover

Section author

M. Köhler, DWD Physical Processes Division

To prepare optical properties of clouds for the radiative transfer it is necessary to determine the best estimate of cloud cover, cloud water and cloud ice as well as the precipitation quantities, such as snow, rain and graupel if those are required by the radiation calculation. Note that there are various assumptions in the ICON model on the subgrid distribution of water, such as the up/down/subsidence regions in convection, a uniform distribution in microphysics, and a Gaussian distribution in turbulence.

The aim of the diagnostic cloud cover scheme is to combine information from the different parameterizations mentioned above: turbulence, convection and microphysics. The turbulence scheme provides the sub-grid variability of water due to turbulent motions, the convection scheme detrains cloud into the anvil and the microphysics scheme describes the supersaturation due to ice, in other words, the distribution between ice and vapor in cold situations.

The turbulent variability of water is at the moment prescribed by using a top-hat total water (the sum of water vapor q_v , cloud water q_c , and cloud ice q_i) distribution with a

fixed width of 10% of total water. Work is in progress to replace this crude assumption with the total water variance from the turbulence scheme.

The split between water vapor and cloud ice as determined from the microphysics scheme is replicated in the diagnostic cloud scheme for the turbulent component of ice clouds.

The convective anvil is calculated by writing an equation for the evolution of cloud cover that depends on the detrainment of volume (from the convection scheme) and a decay term with a fixed decay time-scale (taken as 30 min). The diagnostic assumption means that we can neglect the tendency term on cloud cover so that we arrive simply at the diagnostic anvil cloud cover that is purely a function of detrainment and decay time-scale. The liquid and ice cloud water is taken also from the convective updraft properties.

In the end the turbulent and convective clouds are combined with a simple maximum function.

To emphasize, the cloud cover scheme takes into account the subgrid variability of water and therefore the associated distribution in water vapor, cloud liquid water and cloud ice (optional 3D diagnostic output variables `tot_qv_dia`, `tot_qc_dia`, `tot_qi_dia` and their vertically integrated counterparts `tqv_dia`, `tqc_dia`, `tqi_dia`). They are not equal to their prognostic grid-scale equivalents (standard output variables `qv`, `qc`, `qi`), yet the sum of all three water quantities is kept the same. This cloud information is then passed to the radiation, where additional assumptions are made on the vertical overlap of clouds.

3.8.8. Turbulent Diffusion

Section authors

M. Köhler and M. Raschendorfer,
DWD Physical Processes Division

TKE Scheme for Turbulence.

The TKE turbulence scheme consists of two components: one describing the free troposphere, and the other for the surface layer.

TURBDIFF. The turbulence scheme TURBDIFF developed by Raschendorfer (2001) is based on a 2^{nd} -order closure on level 2.5 according to Mellor and Yamada (1982) (MY-scheme). In this scheme, all pressure correlation terms and dissipation terms, being present in the system of 2^{nd} -order equations for all the turbulent (co-)variances that can be built from the dynamically active prognostic model variables, are expressed by the standard closure assumptions according to Rotta (1951a,b) and Kolmogorov (1968) valid for quasi-isotropic turbulence. These dynamically active model variables are the horizontal wind components u and v , vertical wind speed w , a variable related to inner energy (like absolute temperature T , potential temperature θ or virtual potential temperature θ_v), specific humidity q_v and at least one cloud water variable q_c (which is a mass fraction and may be split into liquid q_l and frozen q_i cloud water).

Among these 2^{nd} -order moments, only the trace of the turbulent stress tensor, which is twice the Turbulent Kinetic Energy (TKE), is described by a prognostic equation. Each of the remaining 2^{nd} -order equations (for the elements of the remaining trace-less stress tensor and for the other 2^{nd} -order moments) is simplified as diagnostic source term equilibrium being a linear equation in terms of the governing statistical moments.

Further, correlations between any model variable and source terms of scalar model variables are neglected in these equations. However, by choosing quasi-conserved scalar variables (total water content $q_t = q_v + q_c$ and liquid-water potential temperature $\theta_l = \theta - \frac{L_c}{c_{p,d}} q_c$), these correlations are taken into account implicitly as far as local condensation and evaporation within liquid non-precipitating clouds is concerned. Hence, TURBDIFF is a moist turbulence scheme, which includes the effect of these sub-grid scale phase transitions. The required conversion of turbulent fluxes of these conserved variables into those of absolute temperature, specific humidity and liquid water content is performed by means of turbulent saturation adjustment, assuming a Gaussian distribution-function for local saturation-deficiency according to [Sommeria and Deardorff \(1977\)](#).

Finally, application of the horizontal boundary layer approximation reduces the linear system of diagnostic 2nd-order equilibrium equations to a single column scheme with only two equations for two diffusion coefficients (one for horizontal wind components and another for scalar variables), which both are proportional to the square root of TKE and an integral turbulent length scale. This length-scale rises with height above ground according to [Blackadar \(1962\)](#) with a further limitation related to the horizontal grid scale. The desired vertical turbulent fluxes of any prognostic variable can then be calculated by multiplying the (negative) vertical gradient of the latter with the associated diffusion coefficient.

One main extension of TURBDIFF (compared with a moist MY-scheme) is the formal separation of turbulence from a possible non-turbulent part of the subgrid-scale energy spectrum. This separation is related to additional scale-interaction terms in the prognostic TKE equation, which describe additional shear production of TKE by the action of other non-turbulent sub-grid-scale flow patterns (such as wakes generated by sub-grid-scale orography, convective currents or separated horizontal circulations). Through this formalism, the scheme describes Separated Turbulence Interacting with non-turbulent Circulations (STIC), which allows for a consistent application of turbulence closure assumptions, even though other sub-grid-scale processes may be dominant within a grid cell. Due to this extension, the scheme is applicable also above the boundary layer and for very stable stratification. The Eddy Dissipation Rate (EDR) calculated by TURBDIFF can even be used to forecast the intensity of Clear Air Turbulence (CAT).

TURBTRAN. The turbulence scheme TURBDIFF is closely related to the scheme TURBTRAN developed by [Raschendorfer \(2001\)](#) for the surface-to-atmosphere transfer (SAT), which calculates transport resistances for fluxes of prognostic model variables at the surface of the Earth. Figure 3.11 illustrates the corresponding sub layers of the surface layer. In TURBTRAN, a constant flux approximation is applied to the sum of turbulent and laminar vertical fluxes within the transfer layer (between the rigid surface and the lowest atmospheric main level of the model). By application of the turbulence scheme at the top of the lowest atmospheric layer as well as the bottom of this layer (which is the top

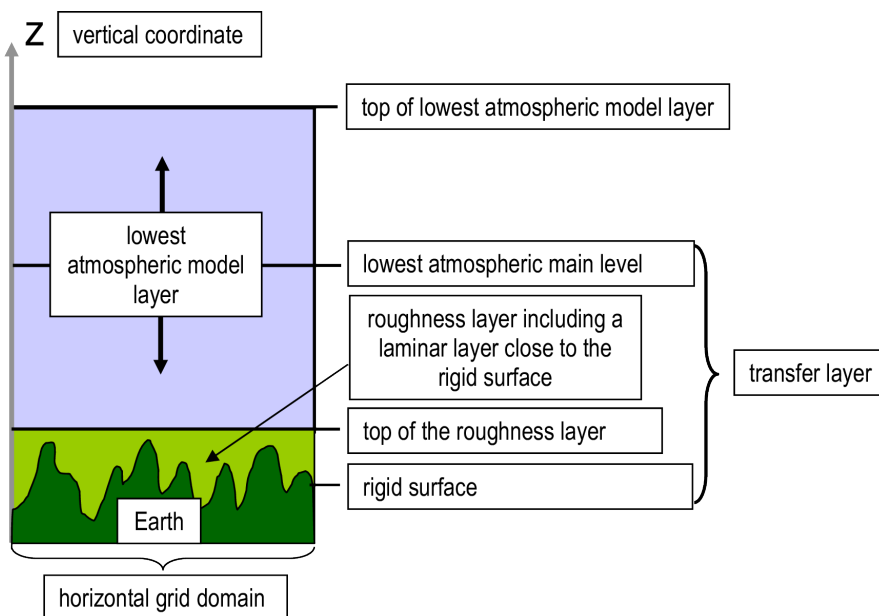


Figure 3.11.: Surface layer as described in the parameterization TURBTRAN of Raschendorfer (2001).

of the near surface roughness layer being intersected by roughness elements), a vertical interpolation function for the turbulent diffusion coefficient is derived between these two levels and is extrapolated down to the rigid surface. With this preparation, a vertical integration of the flux gradient representation across the transfer layer provides the desired transport resistances for the final bulk representation of SAT fluxes. With this formulation, scale interaction terms considered through STIC in the turbulence scheme also affect the transfer resistances; and hence, some additional mixing is automatically introduced for very large bulk Richardson numbers, as soon as non-turbulent sub-grid-scale motions are present.

Further, with the described procedure, the determination of a specific roughness length for scalars is substituted by a direct calculation of the partial resistance of scalar transfer through the laminar layer and the roughness layer. This partial resistance is dependent on the near-surface model variables, the aerodynamic roughness length and the Surface Area Index (SAI), which is a measure of the surface area enlargement by land use.

TURBDIFF and TURBTRAN are the default schemes for atmospheric turbulence and SAT, respectively, in ICON, and they correspond to `inwp_turb=1` (namelist `nwp_phy_nml`). While TURBTRAN provides the transfer resistances for scalars and horizontal wind components, the final surface fluxes of water vapor and sensible heat are determined in TERRA as a result of updated surface values for q_v and (in case of the upcoming implicit treatment of surface temperature) also for T . Based on these surface fluxes, an implicit equation for vertical diffusion is solved as a final part of TURBDIFF. In this procedure also horizontal momentum and TKE is included, while for these 3 quantities the respective surface concentration is used as a lower boundary condition. Currently, a lower zero-concentration condition is applied for cloud-water and -ice, which is always related to a downward flux for these quantities. For vertical diffusion of passive tracers, the diffusion coefficient for scalars is applied, and the lower boundary condition can be specified individually. Al-

though the effect of local condensation and evaporation is considered in the solution of 2nd –order equations, and hence, can amplify the intensity of turbulent vertical mixing, the direct effect of these additional thermodynamic source-terms in the grid-scale budgets of heat, water vapor and liquid water is not yet considered.

In the namelist `turbdiff_nml` several parameters or selectors for optional calculations related to both schemes can be specified. Through this, TURBDIFF can also be configured as a 3D-turbulence scheme, calculating additionally horizontal shear and providing also horizontal diffusion coefficients. This horizontal shear is the sum of related turbulent shear by the mean flow `itype_sher>0` and (if `ltkeshs=.TRUE.`) of additional shear by larger sub-grid scale non-turbulent horizontal circulations (SHS), which are, for their part, generated by shear of the mean flow, but are formally separated from isotropic turbulence in the framework of STIC. Similarly, each horizontal diffusion coefficient is the sum of the isotropic turbulent diffusion coefficient and an optional additional one related to SHS, provided that this STIC term is active (`ltkeshs=.TRUE.`). However, in ICON, the calculation of horizontal diffusion is not yet connected with these non-isotropic diffusion coefficients from TURBDIFF. Rather, for the time being, isotropic diffusion coefficients calculated by a Smagorinsky formulation are automatically used for 3D diffusion.

Smagorinsky-Lilly for LES Application

The 3D sub-grid model of Smagorinsky (1963) with the stability correction of Lilly (1962) is implemented for LES applications. This scheme writes the eddy viscosity K_m as

$$K_m = (C_s \Delta)^2 |S| C_B,$$

with the Smagorinsky constant C_s , the filter width Δ , the norm of the strain rate tensor $|S|$ and the stability correction factor C_B .

The filter width is taken to be $\Delta = (\Delta x \Delta y \Delta z)^{\frac{1}{3}}$. The strain rate tensor is defined as

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right),$$

whose calculation requires special care due to the triangular grid in ICON. A metric correction has been developed that treats the horizontal gradients over a sloped orography given a terrain-following coordinate correctly. The norm of S_{ij} is

$$|S| = \sqrt{2 S_{ij} S_{ij}}.$$

The stability correction factor C_B is given by

$$C_B = (1 - \text{Ri} / \text{Pr}_t)^{\frac{1}{2}}$$

with the gradient Richardson number $\text{Ri} = N^2 / |S|^2$ and the buoyancy frequency $N^2 = \frac{g}{\theta_0} \frac{\partial \theta_v}{\partial z}$.

The Smagorinsky constant usually has the value of $C_s = 0.1 - 0.2$. In the ICON implementation C_s is set by default to `smag_constant = 0.23` (namelist `les_nml`) and the Prandtl number Pr_t to `turb_prandtl = 0.333` (namelist `les_nml`).

3.8.9. Land-Soil Model TERRA

Section author

J. Helmert, DWD Physical Processes Division
D. Reinert, DWD Numerical Models Division

The soil-vegetation-atmosphere-transfer component TERRA (Schrodin and Heise, 2001, Heise et al., 2006, Schulz et al., 2016) in the ICON model is responsible for the exchange of fluxes of heat, moisture, and momentum between land surface and atmosphere. It establishes the lower boundary-condition for the atmospheric circulation model and considers the energy and water budget at the land surface fractions of grid points. Based on a multi-layer concept for the soil, TERRA considers the following physical processes at each of the tiled land-surface columns, where an uniform soil type with physical properties is assumed:

Radiation

- Photosynthetically active radiation (PAR) is used for plant evapotranspiration
- Solar and thermal radiation budget is considered in the surface energy budget

Biophysical control of evapotranspiration

- Stomatal resistance concept controls the interchange of water between the atmosphere and the plant
- One-layer vegetation intercepts and hold precipitation and dew, which lowers water input to the soil and enhances evaporation
- Roots with root-density profile determines the amount of water available for evapotranspiration in the soil
- Bare-soil evaporation is considered for land-surface fractions without plants.

Heat and soil-water transport

- Implicit numerical methods are used to solve the vertical soil water transport and soil heat transfer between the non-equidistant layers.
- In the operational model version seven layers are used in the soil.
- The lower boundary condition for the heat conduction equation is provided by the climatological mean temperature.
- Surface and sub-surface runoff of water is considered.
- The lower boundary condition is given by a free-drainage formulation.
- A rise of groundwater into the simulated soil column is not represented.
- Soil heat conductivity depends on soil-water content.
- Freezing of soil water and melting of soil ice is considered in hydraulic active soil layers.

Snow

- TERRA offers a one-layer snow model (operational in ICON-NWP) and a multi-layer snow model option (for experiments).
- A prognostic snow density, and snow melting process as well as the time dependent snow albedo are considered

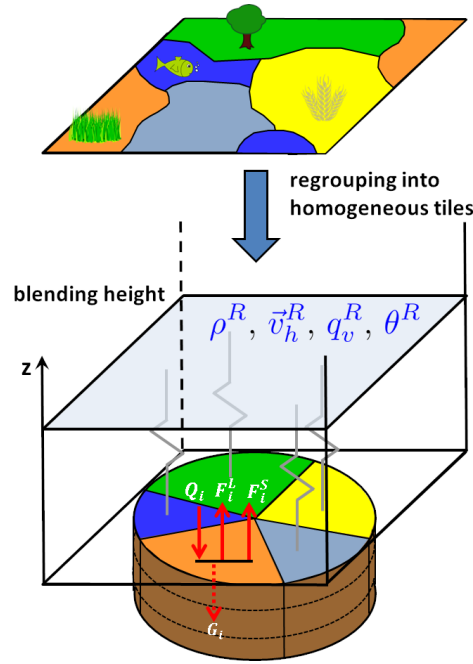


Figure 3.12.: Tile approach for a grid cell containing various surface types. Patches of the same surface type within a grid box are regrouped into homogeneous classes (tiles) for which the soil and surface parameterizations are run separately.

- Surface fractions partly covered with snow are divided in snow-free and snow-covered parts (snow tiles)

Coupling to the atmosphere

- Application of the turbulence scheme at the lower model boundary
- Roughness length for scalars implicitly considered by calculation of an additional transport resistance throughout the turbulent and laminar roughness layer.

TERRA requires a number of external parameter fields, see Section 2.4 for details.

The Tile Approach

The tile approach addresses the problem of calculating proper cell-averaged surface fluxes in the case of large subgrid variations in surface characteristics. The basic idea following [Avisar and Pielke \(1989\)](#) is depicted in Figure 3.12. If patches of the same surface type occur within a grid box, they are regrouped into homogeneous classes (tiles). The surface energy balance and soil physics are then computed separately for each tile, using parameters which are characteristic of each surface type (roughness length, leaf area index, albedo, ...). The atmospheric fields which enter the computations, however, are assumed uniform over the grid cell, i.e. the so-called blending height is located at the lowermost atmospheric model level. The contributions from different tiles are then areally weighted to provide the cell-averaged atmospheric forcing. Note that in this approach the geographical distribution of subgrid heterogeneities is not taken into account.

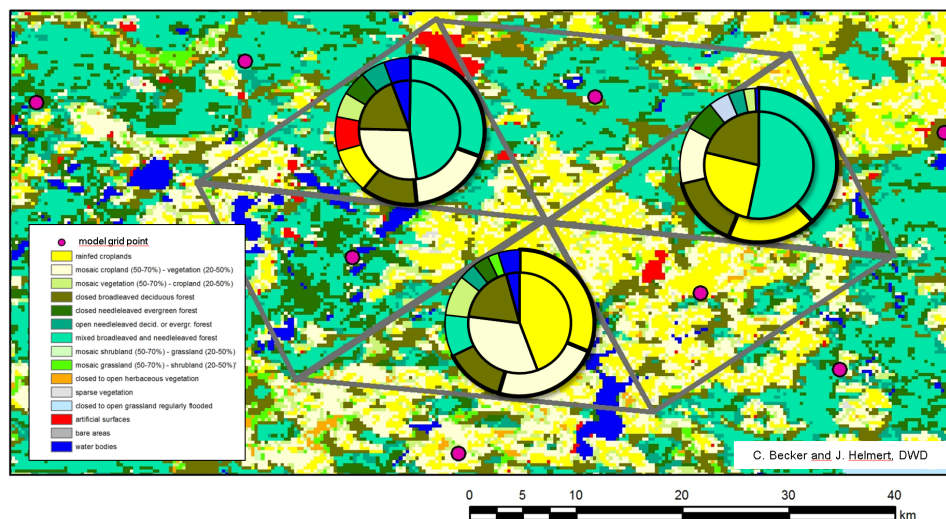


Figure 3.13.: Tile generation for `ntiles=3`, for the case of a heterogeneous land surface. Outer circles show the fractional areas covered by the respective surface-type for a given grid cell. Inner circles show the selected tiles. Please note the re-scaling of fractional areas in the inner circle.

In ICON, the number of surface tiles is specified by the parameter `ntiles` (`lnd_nml`). Setting `ntiles=1` means that the tile approach is switched off, i.e. only the dominant land-surface type in a grid cell is taken into account. Setting `ntiles` to a value $n > 1$, up to n dominant land tiles are considered per grid cell. Note, however, that for $n > 1$ the total number of tiles n_{tot} is implicitly changed to $n_{tot} = n + 3$, with three additional "water" tiles classified as "open water" ($n + 1$), "lake" ($n + 2$), and "sea-ice" ($n + 3$). Additional snow-tiles can be switched on by choosing `lsnowtile=TRUE`. In that case the total number of tiles is further expanded to $n_{tot} = 2 \cdot n + 3$, with the first n tiles denoting the land tiles, the second n tiles denoting the corresponding snow tiles and 3 water tiles as before.

Surface tile generation During the setup phase, all land-surface types within a grid box are ranked according to the fractional area f they cover (see Figure 3.13, outer ring). For efficiency reasons, only the dominating `ntiles` area fractions are represented by tiles (typically about 3), with the others being discarded (inner ring). If a grid cell contains non-negligible water bodies ($f > 5\%$), up to 3 more tiles are created (i.e. open water, lake, and sea-ice) even if they are not among the dominating ones. By this approach, the surface types represented by tiles can differ from grid cell to grid cell such that the full spectrum of surface types provided by the land cover data set is retained.

If the model is initialized from horizontally interpolated initial data and `ntiles > 1`, a tile coldstart becomes necessary. This can be done by setting `ltile_init=TRUE`. and `ltile_coldstart=TRUE`. in the namelist `initicon_nml`. Each tile is then initialized with the same cell averaged value. Note that `ltile_init=TRUE`. is only necessary, if the initial data come from a model run without tiles.

*Important note:*

Naive horizontal interpolation of tile-based variables is incorrect, since the dominant tiles and/or their internal ranking will most likely differ between source and target cell. Only aggregated fields can be interpolated!

3.8.10. Lake Parameterization Scheme FLake

Section author

D. Mironov, DWD Physical Processes Division

Lakes significantly affect the structure and the transport properties of the atmospheric boundary layer. The interaction of the atmosphere with the underlying surface strongly depends on the surface temperature.

In numerical weather prediction (NWP), a simplified approach is often taken that amounts to keeping the water surface temperature constant over the entire forecast period. This approach is to some extent justified for ocean and seas. It is hardly applicable to lakes where diurnal variations of the water surface temperature reach several degrees. The situation is even more grave for frozen lakes as the diurnal variations of the ice surface temperature may exceed ten degrees.

Initialization of the NWP model grid boxes that contain water bodies also presents considerable difficulties. When no observational data for some grid boxes are available, those grid boxes are initialized by means of interpolation between the nearest grid-boxes for which the water surface temperature is known (the interpolation procedure may account for some other variables, e.g., two-meter temperature over land). Such procedure is acceptable for sea points where large horizontal gradients of the water surface temperature are comparatively rare, but it is hardly suitable for lakes. Lakes are enclosed water bodies of a comparatively small horizontal extent. The lake surface temperature has little to do with the surface temperature obtained by means of interpolation between the alien water bodies.

In NWP, the lake surface temperature (i.e., the surface temperature of lake water or lake ice) is a major concern. It is this variable that communicates information between the lake and the atmosphere, whereas details of the vertical temperature distribution (e.g., the temperature near the lake bottom) are of minor importance. Therefore, simplified lake models (parameterization schemes), whose major task is to predict the lake surface temperature, the lake freezing and the ice break-up, should be sufficient for NWP and related applications.

The NWP model ICON (as well as COSMO) utilizes the lake parameterization scheme *FLake* (Mironov, 2008, Mironov et al., 2010, 2012). *FLake* is based on a two-layer parametric representation of the evolving temperature profile. The structure of the stratified layer between the upper mixed layer and the basin bottom, the lake thermocline, is described using the concept of self-similarity (assumed shape) of the temperature-depth curve. The same concept is used to describe the temperature structure of the thermally active upper layer of bottom sediments and of the ice and snow cover. In this way, the problem of

solving partial differential equations (in depth and time) for the temperature and turbulence quantities is reduced to solving ordinary differential equations (in time only) for the time-dependent parameters that specify the temperature profile.

The approach is based on what is actually “verifiable empiricism”. However, it still incorporates much of the essential physics and offers a very good compromise between physical realism and computational economy. FLake incorporates the heat budget equations for the four layers in question, viz., snow, ice, water and bottom sediments, developed with due regard for the volumetric character of the solar radiation heating. An entrainment equation is used to compute the depth of a convectively-mixed layer, and a relaxation-type equation is used to compute the wind-mixed layer depth in stable and neutral stratification. Simple thermodynamic arguments are invoked to develop the evolution equations for the ice and snow depths.

Empirical constants and parameters of FLake are estimated, using independent empirical and numerical data. They should not be re-evaluated when the scheme is applied to a particular lake. In this way, the scheme does not require re-tuning, a procedure that may improve an agreement with a limited amount of data but should generally be avoided. Further information about FLake can be found at <http://lakemodel.net>.

FLake is activated within ICON if the namelist parameter `llake` (`lnd_nml`) is set `.TRUE.`, which is the default operational setting at DWD.

FLake requires two external parameter fields. These are the fields of lake fraction (area fraction of a given numerical-model grid box covered by the lake water) and of lake depth. These external parameter fields are generated with the ExtPar software (see Section 2.4) using the Global Lake Database (Kourzeneva, 2010, Kourzeneva et al., 2012, Choulga et al., 2014).

ICON makes use of a tile approach to compute the grid-box mean values of temperature and humidity (and of other scalars) and the grid-box mean fluxes of momentum and scalars. FLake is applied to the ICON grid boxes whose lake fraction exceeds a threshold value; otherwise the effect of sub-grid scale lakes is ignored. Currently, the value of 0.05 is used (see the namelist variable `frlake_thrld` (`lnd_nml`)).

In the current ICON configuration, the lake depth is limited to 50 m. For deep lakes, the abyssal layer is ignored, a “false bottom” is set at a depth of 50 m, and the bottom heat flux is set to zero. The bottom sediment module is switched off, and the heat flux at the water-bottom sediment interface (or at false bottom) is set to zero. The setting `lflk_botsed_use=.FALSE.` is hard-coded in `mo_data_flake.f90`.

Snow above the lake ice is not considered explicitly. The effect of snow is accounted for in an implicit manner through the temperature dependence of the ice surface albedo with respect to solar radiation Mironov et al. (2012). There is no logical switch to deactivate the snow module of FLake. It is sufficient to set the rate of snow accumulation to zero (hard-coded in `mo_flake.f90`). Without explicit snow layer of the lake ice, the snow depth over lakes is set to zero and the snow surface temperature is set equal to the ice surface temperature.

The attenuation coefficient of lake water with respect to solar radiation is currently set to a default “reference” value for all lakes handled by ICON. It would be advantageous to

specify the attenuation coefficient as a global external parameter field. This can be done in the future as the information about the optical properties of lakes becomes available (not the case at the time being).

Generally, no observational data are assimilated into FLake, i.e., the evolution of the lake temperature, the lake freeze-up, and break-up of ice occur freely during the ICON runs. An exception are the Laurentian Great Lakes of North America. Over the Laurentian Great Lakes, the observation data on the ice fraction (provided by the ICON surface analysis scheme) are used to adjust the ice thickness, the ice surface temperature, and (as needed) the water temperature. See the subroutine `flake_init` in `mo_flake.f90` for details. The use of the ice-fraction data over Great Lakes is controlled by the namelist parameter `use_lakeiceana` (`initicon_nml`).

Finally, a word of caution is in order. Running ICON with the lake parameterization scheme switched off (`llake=.FALSE.`) is not recommended as this configuration has never been comprehensively tested at DWD.

3.8.11. Sea-Ice Parameterization Scheme

Section author

D. Mironov, DWD Physical Processes Division

A major task of the sea-ice parameterization scheme for NWP is to predict the existence of ice within a given atmospheric-model grid box and the ice surface temperature. The sea-ice scheme used within ICON NWP accounts for thermodynamic processes only, i.e., no ice rheology is considered (cf. the sea-ice scheme for climate modeling). The horizontal distribution of the ice cover, i.e., the fractional area coverage of sea ice within a given grid box, is governed by the data assimilation scheme. A detailed description of the sea-ice scheme for ICON NWP is given in [Mironov et al. \(2012\)](#), where a systematic derivation of governing equations, an extensive discussion of various parameterization assumptions and of the scheme disposable parameters, and references to relevant publications can be found. Further comments can be found directly in the code, see the module `src/lnd_phy_schemes/mo_seaice_nwp.f90`.

A distinguishing feature of the ICON NWP sea-ice scheme is the treatment of the heat transfer through the ice. As different from many other sea-ice schemes that solve the heat transfer equation on a finite difference grid, the present scheme uses the integral, or bulk, approach (cf. the lake parameterization scheme FLake, Section 3.8.10). It is based on a parametric representation (assumed shape) of the evolving temperature profile within the ice and on the integral heat budget of the ice slab. Using the integral approach, the problem of solving partial differential equations (in depth and time) is reduced to solving ordinary differential equations (in time only) for the quantities that specify the evolving temperature profile. These quantities are the ice surface temperature and the ice thickness.

In the full-fledged scheme outlined in [Mironov et al. \(2012\)](#), provision is made to account for the snow layer above the ice. Both snow and ice are modeled using the same basic concept, that is a parametric representation of the evolving temperature profile and the integral energy budgets of the ice and snow layers (see [Mironov \(2008\)](#) for a detailed discussion of the concept). In the current ICON configuration, snow over sea ice is not considered

explicitly. The effect of snow is accounted for implicitly (parametrically) through the ice surface albedo with respect to solar radiation.

A prognostic sea-ice albedo parameterization is used. The sea-ice surface albedo is computed from a relaxation-type rate equation, where the equilibrium albedo and the relaxation (e-folding) time scale are computed as functions of the ice surface temperature. In order to account for the increase of the sea-ice albedo after snowfall events, the ice albedo is relaxed to the equilibrium “snow-over-ice” albedo. The equilibrium snow-over-ice albedo is computed as function of the ice surface temperature, and the relaxation time scale is related to the snow precipitation rate.

The horizontal distribution of the ice cover, i.e., the existence of sea ice within a given ICON grid box and the ice fraction, is governed by the data assimilation scheme (cf. the treatment of lake ice). If an ICON grid box has been set ice-free during the initialization, no ice is created over the forecast period. If observational data indicate open water conditions for a given grid box but there was ice in that grid box at the end of the previous ICON run, ice is removed and the grid box is initialized as ice-free. The new ice is formed instantaneously if the data assimilation scheme indicates that there is sea ice in a given grid box, but there was no ice in that grid box in the previous model run. The newly formed ice has the surface temperature equal to the salt-water freezing point. The thickness of newly formed ice is computed as function of the ice fraction.

ICON utilizes a tile approach to compute surface fluxes of momentum and scalars. For the “sea-water type” grid boxes, the grid-box mean fluxes are computed as a weighted mean of fluxes over ice and over open water, using fractional ice cover f_i and fractional open-water cover $1 - f_i$ as the respective weights. Sea ice in a given ICON grid box is only considered if f_i exceeds its minimum value of 0.015, otherwise the grid box is treated as ice free (see parameter `frsi_min` hard-coded in `mo_seaice_nwp.f90`). Likewise, the open-water fraction less than `frsi_min` are ignored, and the grid box in question is treated as fully ice-covered (f_i is reset to 1). The ice fraction is determined during the model initialization and is kept constant over the entire forecast period. If, however, sea ice melts away during the forecast, f_i is set to zero and the grid box is treated as an open-water water grid box for the rest of the forecast period (prognostic ice thickness is limited from below by a value of 0.05 m, i.e., a thinner ice is removed). The water-surface temperature of that grid box is equal to the observed value from the analysis, or is reset to the salt-water freezing point. The latter situation is encountered when a grid box was entirely covered by ice at the beginning of the forecast, but the ice melts away during the forecast.

In order to run ICON with the sea-ice parameterization scheme switched off, the namelist logical switch `lseaice` (`lnd_nml`) should set equal to `.FALSE..` This configuration has not been comprehensively tested at DWD and is not recommended.

3.8.12. Reduced Model Top for Moist Physics

A notable means for improving the efficiency of ICON is depicted in Figure 3.14. The switch

`htop_moist_proc` (namelist `nonhydrostatic_nml`, floating-point value)

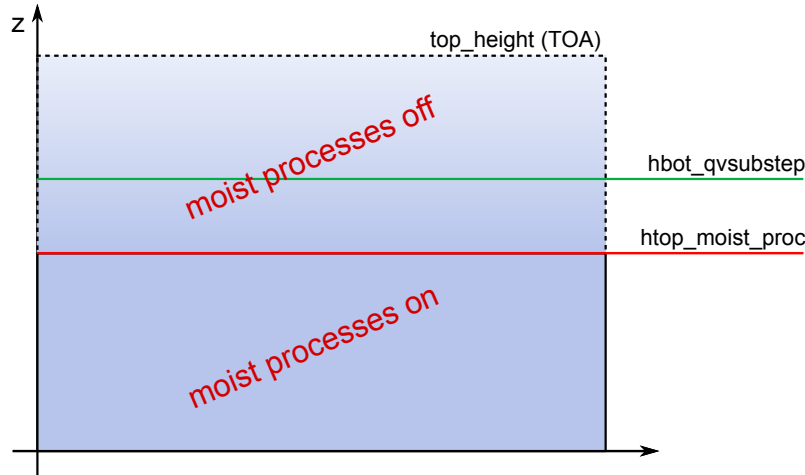


Figure 3.14.: Moist physics are switched off above `htop_moist_proc`, while tracer substepping is switched on above `hbot_qvsubstep`. (Remark: `hbot_qvsubstep` is allowed to be lower than `htop_moist_proc`)

allows to switch off moist physics completely above a certain height. Moist physics include saturation adjustment, grid scale microphysics, convection, cloud cover diagnostic, as well as the transport of all water species but moisture q_v . Of course, moist processes should only be switched off well above the tropopause. The default setting is `htop_moist_proc=22500 m`.

One variant of the implemented horizontal transport scheme for passive scalars is capable of performing internal substepping. This means that the transport time step Δt is split into n (usually 2 or 3) substeps during flux computation. This proves necessary in regions where the horizontal wind speed exceeds a value of about 80 m s^{-1} . In real case applications, this mostly happens in the stratosphere and mesosphere. The recommendation for Δt given in Section 3.7.1 then exceeds the numerical stability range of the horizontal transport scheme. To stabilize the integration without the need to reduce the time step globally, transport schemes with and without internal substepping can be combined. The switch

`hbot_qvsubstep` (namelist `nonhydrostatic_nml`, floating-point value)

indicates the height above which the transport scheme switches from its default version to a version with internal substepping. The default value is `hbot_qvsubstep=22500 m`.

Note that substepping is only performed for a particular tracer if a suitable horizontal transport scheme is chosen. The horizontal transport scheme can be selected individually for each tracer via the namelist switch `ihadv_tracer` (`transport_nml`). Variants of the transport scheme with internal substepping are indicated by a two-digit number (i.e. 22, 32, 42, 52). These variants mostly differ w.r.t. the accuracy of the polynomial reconstruction used for the flux estimation. A linear reconstruction is used by the variant 22, whereas 52 uses a cubic reconstruction. See Section 3.6.5 for additional details regarding the transport algorithm.

If moist physics are switched off above 22.5 km (default for NWP applications), internal substepping only needs to be applied for specific humidity q_v , since the advection of all

other moisture fields is switched off anyway. However, be aware that you must explicitly enable internal substepping if moisture physics are not switched off, or if other (non-microphysical) tracers are added to the simulation.

3.9. Variable Resolution Modeling

ICON has the capability for static mesh refinement in horizontal directions (Zängl et al., 2022). This is realized through a *multi-grid approach* which means that one or more additional higher resolution (child) domains can be overlaid on a coarser base (parent) domain. This base domain can be a regional or a global domain. Each child domain has a defined parent domain providing lateral boundary conditions, but a parent domain can have several child domains. The child domains can be located in different geographical regions and can also be parent domains for further subdomains. Technically, the number of nested domains is arbitrary, but of course not all choices would make sense from a physical point of view.



The multi-grid approach in ICON closely resembles the classical two-way nesting approach known from many mesoscale models, e.g. MM5 (Grell et al., 1994) or WRF (Skamarock et al., 2019) but differs in the fact that the feedback is based on a Newtonian relaxation approach rather than directly replacing the prognostic fields in the parent domain by up-scaled values from the child domain. It also has to be distinguished from recent *uni-grid* approaches, where more cells are added to an existing grid in special areas of interest (*h-refinement*), and where the solver computes a single solution for the whole grid. Atmospheric models capable of static h-refinement are e.g. CAM-SE (Zarzycki et al., 2014) and MPAS-A (Skamarock et al., 2012).

The multi-grid approach easily allows for switching domains on or off at runtime, as well as intertwining one-way and two-way nested domains. Two-way as opposed to one-way nesting means that the solution on the child domain is transferred back to the coarser parent domain every time step by means of a feedback mechanism which is described below.

The basic multi-grid example shown in Figure 3.15 consists of one global domain and one regional domain over Europe. The refinement ratio between the parent domain and the child domain is fixed to a value of 2, i.e. each parent triangle is split into 4 child triangles as shown in Figure 3.16. Consistent with the refinement ratio of 2, the time steps for dynamics and physics are automatically halved for each nesting level. Hence, Δt must be specified for the base (i.e. coarsest) domain only.

The coupling time step Δt between successive nesting levels is the large (fast physics) time step described in Zängl et al. (2015), which is usually five times the so-called dynamics



Figure 3.15.: Basic example of a multi-grid setup, consisting of a global ICON domain and a child domain over Europe with half grid spacing. This is similar to the deterministic forecast setup that is operationally used at DWD with a horizontal grid spacing of 13 km globally and 6.5 km in the child domain.

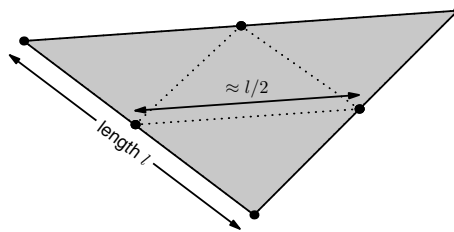


Figure 3.16.: Nested grids are constructed by splitting each parent cell into 4 child cells.

time step limited by the Courant-Friedrichs-Lewy (CFL) stability criterion for horizontal sound-wave propagation, see Section 3.7.1.

The grids corresponding to the different refinement levels are stored in separate files. The usual way to establish a parent-child relationship between these grids is to read the header information from the list of provided grid files:

dynamics_grid_filename (namelist **grid_nml**, list of string parameters)

This parameter specifies the name(s) of the horizontal grid file(s) and thus implicitly activates the nesting function. For a global simulation with multiple nests a filename must be specified for each domain, see Section 4.1.2 for a practical example. Then, the parent-child relationships can be inferred from the NetCDF attributes `uuidOfHGrid` and `uuidOfParHGrid`, that have been described in Section 2.1.2.

3.9.1. Parent-Child Coupling

This section describes the exchange of information between a single parent and child domain. As shown in Figure 3.17, a nested domain can conceptually be split into three areas: A boundary interpolation zone (red), a nudging zone (light gray) and a feedback zone (blue). Prognostic computations are restricted to the latter two. The nudging zone only exists for one-way nesting or in limited area mode (LAM), whereas for two-way nesting the feedback zone directly borders on the boundary zone.

In the following let n and $n + 1$ denote the begin and end of the current large time step. Once the model state on the parent domain \mathcal{M}_p has been advanced from n to $n + 1$, the states \mathcal{M}_p^n and \mathcal{M}_p^{n+1} are used to update the boundary interpolation zone on the child domain. Hence, the boundary interpolation zone provides the necessary lateral boundary (forcing) data in order to advance the model state on the child domain from \mathcal{M}_c^n to \mathcal{M}_c^{n+1} . It has a fixed width of 4 cell rows (see Figure 3.17), which is motivated by the technical constraint that the boundary zone needs to match with parent cell rows (i.e. an odd number of cell rows is not allowed), combined with the fact that 2 cell rows would not be sufficient to cover all stencil operations performed in the dynamical core. For example, the ∇^4 -diffusion operator (Zängl et al., 2015) requires information from three adjacent cell rows.

In the feedback zone, the updated model state on the child domain \mathcal{M}_c^{n+1} is transferred (interpolated) back to the parent domain. By this, the parent and child domain remain closely coupled, and the simulation on the parent domain benefits from the higher-resolution results of the child domain.

In the nudging zone, the model state on the child domain \mathcal{M}_c^{n+1} is nudged towards the corresponding parent domain state \mathcal{M}_p^{n+1} , in order to accommodate possible inconsistencies between the two domains. The nudging is essentially a relaxation of the prognostic variables towards the lateral boundary data following Davies (1976). The same method is applied in limited area mode (see Section 6.2).

Further details on the different zones and the boundary update and feedback mechanism are given in the following.

Lateral Boundary Update: Parent \rightarrow Child

The boundary update mechanism provides the child domain with up-to-date lateral boundary conditions for the prognostic variables $v_n, w, \rho, \theta_v, q_k$. In order to avoid that parent-to-child interpolated values of ρ enter the solution of the mass continuity equation, the above set of variables is extended by the horizontal mass flux ρv_n . This will allow for parent-child mass flux consistency, as described below.

In general, the boundary update works as follows: Let ψ_p^n, ψ_p^{n+1} denote any of the above variables on the parent domain at time steps n and $n + 1$, respectively. Once the model state on the parent domain \mathcal{M}_p has been updated from n to $n + 1$, the time tendency

$$\frac{\partial \psi_p}{\partial t} = \frac{\psi_p^{n+1} - \psi_p^n}{\Delta t_p}$$

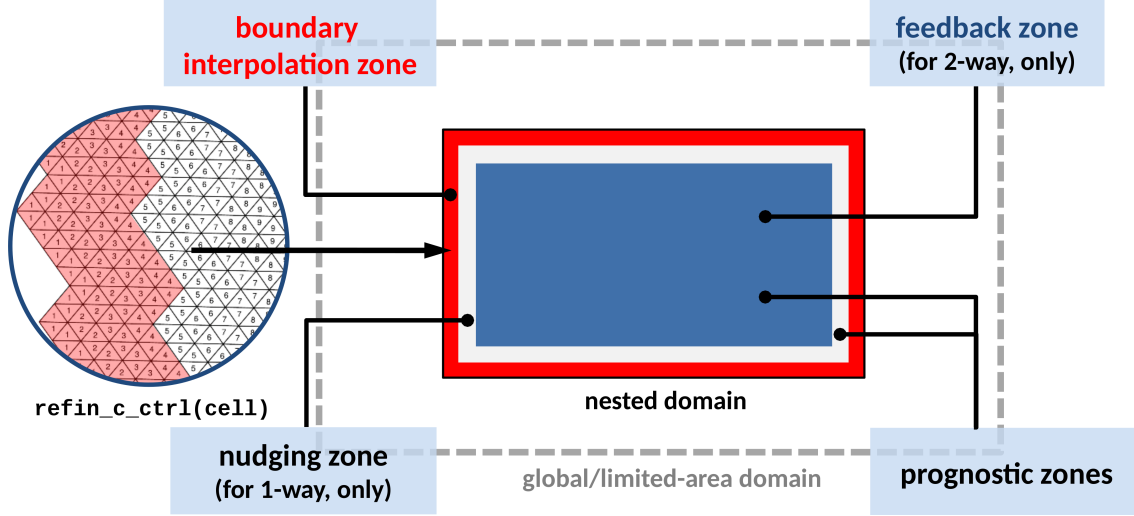


Figure 3.17.: General structure of a nested domain. **Red:** boundary interpolation zone consisting of 4 cell rows (see enlarged view). **Light-gray:** nudging zone with adjustable width, which is only active for one-way nesting and in limited area mode. **Blue:** feedback zone. Prognostic computations are performed in the feedback and nudging zone.

is diagnosed. Both, the field ψ_p^n at time level n and the tendency $\frac{\partial \psi_p}{\partial t}$ are then interpolated (downscaled) from the parent grid cells/edges to the corresponding cells/edges of the child's boundary zone. With $\mathcal{I}_{p \rightarrow c}$ denoting the interpolation operator, we get

$$\begin{aligned}\psi_c^n &= \mathcal{I}_{p \rightarrow c}(\psi_p^n) \\ \frac{\partial \psi_c}{\partial t} &= \mathcal{I}_{p \rightarrow c}\left(\frac{\partial \psi_p}{\partial t}\right)\end{aligned}$$

The interpolated tendencies are generally needed in order to provide the lateral boundary conditions at the right time levels, since two integration steps are necessary on the child domain in order to reach the model state \mathcal{M}_c^{n+1} , with each step consisting of `ndyn_substeps` (`nonhydrostatic_nml`) dynamics sub-steps. E.g. for the first and second (large) integration step on the child domain the boundary conditions read ψ_c^n and $\psi_c^n + 0.5 \Delta t_p \partial \psi_c / \partial t$, respectively.

Regarding the interpolation operator $\mathcal{I}_{p \rightarrow c}$ we distinguish between cell based variables (i.e. scalars) and edge-based variables (v_n and ρv_n). For cell based variables a 2D horizontal gradient is reconstructed at the parent cell center by first computing edge-normal gradients at edge midpoints, followed by a 9-point reconstruction of the 2D gradient at the cell center based on radial basis functions (RBF, [Narcowich and Ward \(1994\)](#)). The interpolated value at the j th child cell center is then calculated as

$$\psi_{c_j} = \psi_p + \nabla \psi_p \cdot \mathbf{d}(p, c_j), \quad j \in \{1 \dots 4\}, \quad (3.56)$$

with $\nabla \psi_p$ denoting the horizontal gradient at the parent cell center, and $\mathbf{d}(p, c_j)$ the distance vector between the parent and j th child cell center. The same operator is applied to cell based tendencies.

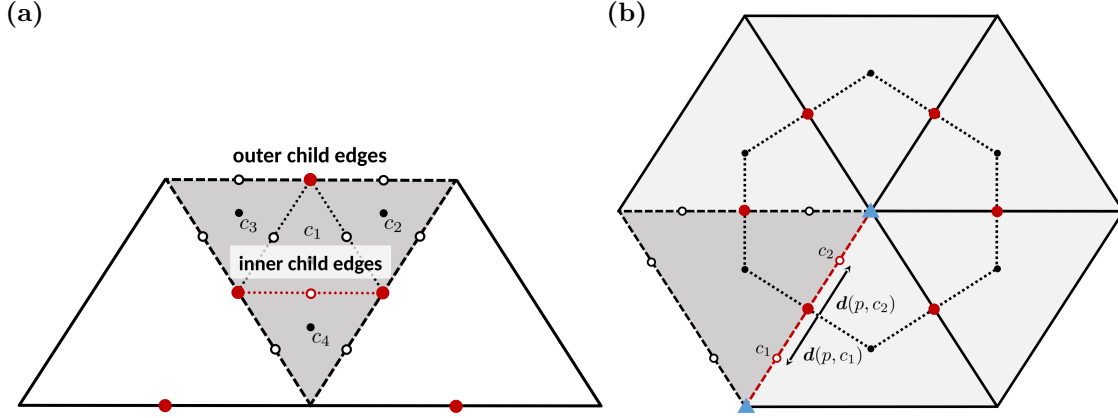


Figure 3.18.: Horizontal reconstruction stencil for edge-normal vector components at (a) inner child edges and (b) outer child edges. The child edge under consideration is highlighted in red. Black open dots indicate child edge midpoints, while black solid dots indicate cell circumcenters. Solid red dots represent the reconstruction stencil, i.e. the location of the parent edge-normal vector components entering the reconstruction, and light-blue triangles in (b) indicate the location of the reconstructed 2D vectors. See the text for details.

To prevent excessive over- and undershoots of ψ_{c_j} in the vicinity of strong gradients, a limiter for $\nabla\psi_p$ is implemented. It ensures that

$$\frac{1}{\beta}\psi_{p,\min} < \psi_{c_j} < \beta\psi_{p,\max} \quad \forall j \in \{1 \dots 4\}$$

on all four child points, where $\psi_{p,\min}$ and $\psi_{p,\max}$ denote the minimum and maximum of ψ_p , respectively, on the above-mentioned reconstruction stencil plus the local cell center, and $\beta = 1.05$ is a tuning parameter.

Regarding the interpolation of edge-based variables (i.e. the edge-normal vector components v_n and ρv_n), we distinguish between *outer child edges* that coincide with the edges of the parent cell, and *inner child edges* (see Figure 3.18a).

Edge-normal vector components at the inner child edges are reconstructed by means of a direct RBF reconstruction using the five-point stencil indicated in Figure 3.18a. For a given inner child edge the stencil comprises the edges of the corresponding parent cell, and the two edges of the neighboring parent cells that (approximately) share the orientation of the inner child edge.

For the outer child edges a more elaborate reconstruction is applied, in order to assure that the mass flux across a parent edge equals the sum of the mass fluxes across the corresponding child edges. We start with an RBF reconstruction of the 2D vector of the respective variable at the triangle vertices, using the six (five at pentagon points) edge points adjacent to a vertex (see Figure 3.18b).

The edge-normal vector component ϕ at the child edge is then computed as

$$\phi_{c_e} = \phi_p + \nabla_t \phi_p \cdot \mathbf{d}(p, c_e), \quad e \in \{1, 2\},$$

with $\mathbf{d}(p, c_e)$ denoting the distance vector between the parent and child edge midpoints for a given parent edge, and $\nabla_t \phi_p$ denoting the gradient of the edge-normal vector component ϕ_p tangent to the parent edge. The latter is computed by projecting the 2D vectors at the two vertices of an edge onto the edge-normal direction and taking the centered difference. Since by construction $\mathbf{d}(p, c_1) = -\mathbf{d}(p, c_2)$ holds on the ICON grid, the above mentioned mass flux consistency is ensured.

It is noted that attempts to use higher-order polynomial interpolation methods, which are the standard in mesoscale models with regular quadrilateral grids, were unsuccessful on the triangular ICON grid, because the ensuing equation system led to the inversion of nearly singular matrices.

In order to minimize interpolation errors, the following modifications from the above interpolation procedure are applied: For the thermodynamic variables ρ and θ_v perturbations from the reference state rather than the full values are interpolated, in order to reduce interpolation errors above steep orography.

Rather than interpolating v_n and its time tendencies, only the time tendencies are interpolated, and then used to update v_n at child level at every dynamics time step. This methodology has been chosen because the comparatively inaccurate interpolation to the interior child edges tends to induce small-scale noise in v_n . To suppress the remaining noise arising from the interpolation of the time tendency, a second-order diffusion operator is applied in the inner half of the boundary interpolation zone on v_n , and the default fourth-order diffusion applied in the prognostic part of the model domain (see Zängl et al. (2015)) is enhanced in the five grid rows adjacent to the interpolation zone. For the other prognostic variables, no special filtering is applied near nest boundaries. In the case of one-way nesting, the second-order velocity diffusion is extended into the nudging zone of the nested domain, replacing the enhanced fourth-order diffusion. More details on the nudging zone are given in Section 3.9.1.

For the horizontal mass flux ρv_n , the time average over the dynamic sub-steps, which is passed to the tracer transport scheme in order to achieve mass consistency, is interpolated instead of time level n . Using the mass flux time tendency that is interpolated as well, the related time shift is corrected for when applying the boundary mass fluxes at child level. In the nested domain, the interpolated mass fluxes valid for the current time step are then prescribed at the interface edges separating the boundary interpolation zone from the prognostic part of the nested domain. Due to the flux-form scheme used for solving the continuity equation (see Zängl et al. (2015)), this implies that the interpolated values of ρ do not enter into any prognostic computations in the dynamical core. They are needed, however, for some computations in the transport scheme. Moreover, no mass fluxes at interior child edges are used, so that the non-conservative interpolation method used for those edges does not affect the model's conservation properties. For θ_v and the tracer variables q_k , the values at the edges are reconstructed in the usual manner (see Zängl et al. (2015)) and then multiplied with the interpolated mass fluxes before computing the flux divergences.

Feedback: Child → Parent

If two-way nesting is activated (`lfeedback=.TRUE.`, namelist `grid_nml`), the model state \mathcal{M}_p^{n+1} on the parent domain is relaxed towards the updated model state \mathcal{M}_c^{n+1} on the child domain at every fast physics time step. In the following we will refer to this as *relaxation-type feedback*. It is restricted to the prognostic variables v_n, w, θ_v, ρ plus specific humidity q_v and the specific contents of cloud water q_c and cloud ice q_i ⁵. Precipitating hydrometeors are excluded because recommended relaxation times (see below) are longer than their typical falling times. Surface variables are excluded as well because they can easily adjust during runtime and a proper treatment of feedback along land-cover inhomogeneities (e.g. coastlines) would be complicated and probably computationally expensive.

Let ψ denote any of the above mentioned variables. Conceptually, the feedback mechanism is based on the following three basic steps:

1. **Upscaling:** The updated field ψ_c^{n+1} is interpolated (upscaled) from the child domain to the parent domain. The upscaling operators for cell based and edge based variables will be denoted by $\mathcal{I}_{c \rightarrow p}$ and $\mathcal{I}_{ce \rightarrow p}$, respectively.
2. **Increment computation:** The difference between the solution on the parent domain ψ_p^{n+1} and the upscaled solution $\mathcal{I}_{c \rightarrow p}(\psi_c^{n+1})$ is computed.
3. **Relaxation:** The solution on the parent domain is relaxed towards the solution on the child domain. The relaxation is proportional to the increment computed in Step 2.

For cell based variables the upscaling consists of a modified barycentric interpolation from the four child cells to the corresponding parent cell:

$$\mathcal{I}_{c \rightarrow p}(\psi_c) = \sum_{j=1}^4 \alpha_j \psi_{c_j}.$$

The weights α_j are derived from the following constraints (3.57)–(3.59). First of all, a desirable property for the value interpolation is that it reproduces constant fields, i.e. the weights are normalized:

$$\sum_{j=1}^4 \alpha_j = 1. \quad (3.57)$$

Moreover, the interpolation is linear: With the four child cell circumcenters \mathbf{x}_j ($j = 1, \dots, 4$), and \mathbf{x}_p denoting the parent cell center, i.e. the interpolation target, we set

$$\sum_{j=1}^4 \alpha_j (\mathbf{x}_j - \mathbf{x}_p) = 0. \quad (3.58)$$

To motivate this constraint, consider the special case of equilateral triangles in which the center point of the inner child cell \mathbf{x}_1 coincides with the parent center such that the term $(\mathbf{x}_1 - \mathbf{x}_p)$ vanishes. Equation (3.58) now defines a barycentric interpolation within the triangle spanned by the mass points of the three outer child cells $\{c_2, c_3, c_4\}$ (see Figure 3.18a), where the weights $\{\alpha_2, \alpha_3, \alpha_4\}$ represent the barycentric coordinates.

⁵Note that when programming ICON the feedback mechanism can easily be switched on for additional tracers by adding the meta-information `lfeedback=.TRUE.` to the corresponding `add_ref`-call, see also Section 9.4.

Of course, the contribution of the point \mathbf{x}_1 closest to the interpolation target is of particular importance. Therefore, the underdetermined system of equations (3.57), (3.58) is closed with a final constraint which reads as

$$\alpha_1 = \frac{a_{c_1}}{a_p}, \quad (3.59)$$

where a_{c_1} and a_p denote the inner child and parent cell areas, respectively. In other words, the inner child cell c_1 containing the parent cell circumcenter is given a pre-defined weight corresponding to its fractional area coverage. This can be interpreted as a conservation constraint for the special case of a very localized signal at the mass point of the inner child cell.

In summary, this method can be regarded as a *modified* barycentric interpolation for the mass points $\{\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, and which accounts for \mathbf{x}_1 as an additional fourth source point. A more stringent barycentric interpolation would require an additional triangulation based on the child mass points. This is done for ICON's model output on regular lat-lon grids, see Section 7.1.2.

For velocity points, a simple arithmetic average of the two child edges lying on the parent edge is taken.

$$\mathcal{I}_{ce \rightarrow p}(v_{n,e}) = \frac{1}{2} [v_{n,e_{\text{child } 1}} + v_{n,e_{\text{child } 2}}]$$

We note that the operator $\mathcal{I}_{c \rightarrow p}$ is not strictly mass conserving and that strict mass conservation would require some means of area-weighted aggregation from the child cells to the parent cells, which is available as an option. The problem with such methods on the ICON grid is related to the fact that the mass points lie in the circumcenter rather than the barycenter of the triangular cells. Using an area-weighted aggregation from the child cells to the parent cells, would map linear horizontal gradients on the child grid into a checkerboard noise pattern between upward and downward oriented triangles on the parent grid.

Another difficulty that was encountered in the context of mass conservation is related to the fact that the density decreases roughly exponentially with height. In the presence of orography, the atmospheric mass resolved on the model grid therefore increases with decreasing mesh size, assuming the usual area-weighted aggregation of the orographic raw data to the model grid. Feeding back ρ is thus intrinsically non-conservative. To keep the related errors small and non-systematic, and to generally reduce the numerical errors over steep mountains, perturbations from the reference state are used for upscaling ρ and θ_v to the parent grid. A closer investigation of the related conservation errors revealed that the differences between modified barycentric and area-weighted averaging are (with real orography) unimportant compared compared to the resolution-dependent conservation error.

When combining the above mentioned steps, the feedback mechanism for ρ can be cast into the following form:

$$\rho_p^* = \rho_p^{n+1} + \frac{\Delta t_p}{\tau_{fb}} (\mathcal{I}_{c \rightarrow p}(\rho_c^{n+1} - \Delta \rho_{corr}) - \rho_p^{n+1}) \quad (3.60)$$

Here ρ_p^{n+1} denotes the density in the parent cell, which has already been updated by dynamics and physics. The superscript “*” indicates the final solution, which includes the increment due to feedback. Δt_p is the fast physics time step on the parent domain, and τ_{fb} is a user-defined relaxation time scale which has a default value of $\tau_{fb} = 10800$ s. This value is motivated by the wish to exclude small scale transient features from the feedback, but to fully capture synoptic-scale features. The relaxation time scale is independent of the relaxed field and can be adjusted by means of the namelist variable `fbk_relax_timescale` (`gridref_nml`). Finally note that the upscaled quantity includes the correction term $\Delta\rho_{corr}$ which has been introduced in order to account for differences in the vertical position of the child and parent cell circumcenters. At locations with noticeable orography, cell circumcenter heights at parent cells can differ significantly from those at child cells. If this is not taken into account, the feedback process will introduce a non-negligible bias in the parent domain’s mass field. The correction term is given by

$$\Delta\rho_{corr} = (1.05 - 0.005 \mathcal{I}_{c \rightarrow p}(\theta'_{v,c}{}^{n+1})) \Delta\rho_{ref,p},$$

with the parent-child difference in the reference density field

$$\Delta\rho_{ref,p} = \mathcal{I}_{c \rightarrow p}(\rho_{ref,c}) - \rho_{ref,p},$$

and the potential temperature perturbation $\theta'_{v,c}{}^{n+1} = \theta_{v,c}^{n+1} - \theta_{vref,c}$. The term $\Delta\rho_{ref,p}$ is purely a function of the parent-child height difference and can be regarded as a first order correction term. In order to minimize the remaining mass drift, the empirically determined factor $(1.05 - 0.005 \mathcal{I}_{c \rightarrow p}(\theta'_{v,c}{}^{n+1}))$ was added, which introduces an additional temperature dependency. Note that the factor 0.005 is close to near surface values of $\frac{\partial \rho}{\partial \theta}$ which can be derived from the equation of state. We further note that a possibly more accurate and less ad hoc approach would require a conservative remapping step in the vertical, prior to the horizontal upscaling.

Care must be taken to ensure that the feedback process retains tracer and air mass consistency. To this end, feedback is not implemented for tracer mass fractions directly, but for partial densities. In accordance with the implementation for ρ , we get

$$(\rho q_k)_p^* = (\rho q_k)_p^{n+1} + \frac{\Delta t_p}{\tau_{fb}} \left[\mathcal{I}_{c \rightarrow p}((\rho_c^{n+1} - \Delta\rho_{corr}) q_{k,c}^{n+1}) - (\rho q_k)_p^{n+1} \right] \quad (3.61)$$

Mass fractions are re-diagnosed thereafter:

$$q_{k,p} = \frac{(\rho q_k)_p^*}{\rho_p^*}$$

When summing Eq. (3.61) over all partial densities, Eq. (3.60) for the total density is recovered.

A very similar approach is used for θ_v . As for ρ , only the increment of θ_v is upscaled from the child- to the parent domain and added to the parent reference profile $\theta_{vref,p}$.

$$\theta_{v,p}^* = \theta_{v,p}^{n+1} + \frac{\Delta t_p}{\tau_{fb}} (\mathcal{I}_{c \rightarrow p}(\theta'_{v,c}{}^{n+1}) + \theta_{vref,p} - \theta_{v,p}^{n+1})$$

The same approach is taken for w , however the full field is upscaled.

$$w^* = w^{n+1} + \frac{\Delta t_p}{\tau_{fb}} (\mathcal{I}_{c \rightarrow p}(w_c^{n+1}) - w_p^{n+1})$$

In the case of v_n some numerical diffusion is added to the resulting feedback increment in order to damp small-scale noise.

$$v_{n,p}^* = v_{n,p}^{n+1} + \frac{\Delta t_p}{\tau_{fb}} (\Delta v_{n,p} + K \nabla^2 (\Delta v_{n,p})) ,$$

with the feedback increment

$$\Delta v_{n,p} = \mathcal{I}_{ce \rightarrow p}(v_{n,c}^{n+1}) - v_{n,p}^{n+1} ,$$

and the diffusion coefficient $K = \frac{1}{12} \frac{a_{p,e}}{\Delta t_p}$, where $a_{p,e}$ is the area of the quadrilateral spanned by the vertices and centers adjacent to the parent's edge.

Lateral Nudging

If the feedback is turned off, i.e. if one-way nesting is chosen, a nudging of the prognostic child grid variables towards the corresponding parent grid values is needed near the lateral nest boundaries in order to accommodate possible inconsistencies between the two grids, particularly near the outflow boundary. Because lateral boundaries are in general not straight lines on the unstructured ICON grid, attempts to make an explicit distinction between inflow and outflow boundaries (e.g. by prescribing v_n at inflow boundaries only) were not successful.

To compute the nudging tendencies, the child grid variables are first upscaled to the parent grid in the same way as for the feedback, followed by taking the differences between the parent-grid variables and the upscaled child-grid variables. The differences are then interpolated to the child grid using the same methods as for the lateral boundary conditions (see above). The relaxation uses weighting factors decreasing exponentially from the inner margin of the boundary interpolation zone towards the interior of the model domain. Its width and the relaxation time scale have default values of 8 cell rows and $0.02 \Delta \tau$ (dynamics time step), respectively, and the nudging coefficients decay with an e-folding width of 2 cell rows. These values can be adjusted by means of the namelist parameters `nudge_zone_width`, `nudge_max_coeff`, and `nudge_efold_width` in the namelist `interp1_nml`. See Section 6.2 for additional details on lateral boundary nudging. As already mentioned, a second-order diffusion on v_n is used near the lateral nest boundaries in order to suppress small-scale noise.

Vertical Nesting

The vertical nesting option allows to set model top heights individually for each domain, with the constraints that the child domain height is lower or at most equal to the parent domain height, and that the child domain extends into heights where the coordinate surfaces are flat (see namelist parameter `flat_height` (`sleve_nml`) for the SLEVE vertical

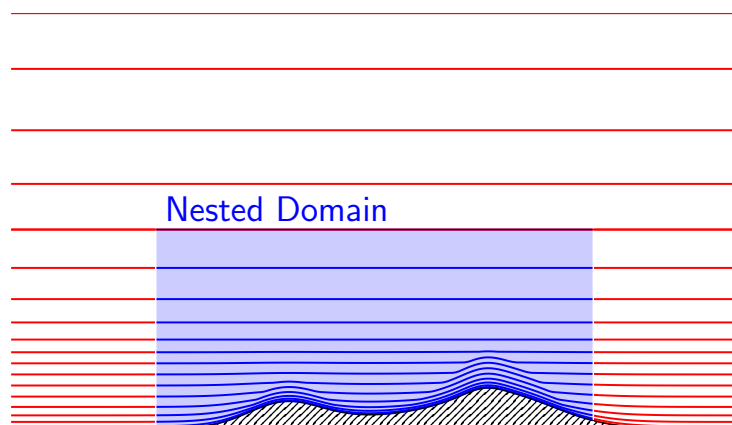


Figure 3.19.: Illustration of ICON’s vertical nesting. Note that the nested domain may have a lower top level height, while the remaining vertical layers must match between the nested and the parent domain. Further note that the parent-nest levels do not coincide in a strict geometrical sense. Differences exist, caused by the resolution-dependent topography, which are omitted in this figure for clarity.

coordinate). This allows, for instance, a global domain extending into the mesosphere to be combined with a child domain that extends only up to the lower stratosphere (see Figure 3.19). However, a vertical refinement in the sense that the vertical resolution in the child domain may differ from that in the parent domain is not available. One possible workaround might be to repeat the model run with the desired vertical resolution in limited area mode (see Chapter 6).

In order to reduce the top height for child domains, the namelist parameter `num_lev` (`run_nml`), which specifies the number of vertical levels in each domain, must be adapted. In addition, vertical nesting must be enabled by setting `lvert_nest=.TRUE.`. See Section 3.4 for details.

Vertical nesting requires appropriate boundary conditions for all prognostic variables to be specified at the vertical nest interface level, i.e. the uppermost half level of the nested domain. This is crucial in order to prevent vertically propagating sound and gravity waves from being spuriously reflected at the nest interface. For details regarding the derivation of these boundary conditions, the reader is referred to Zängl et al. (2022).

3.9.2. Processing Sequence

So far, we have focused on the coupling of an individual parent and child domain. The coupling of multiple and possibly repeatedly nested domains requires a well conceived processing sequence, whose basics will be described in the following.

Figure 3.20 provides a common example where a global domain is combined with two repeatedly nested domains (two-way). The global domain is schematically depicted at the bottom, whereas the nested domains are vertically staggered on top of it. The red and

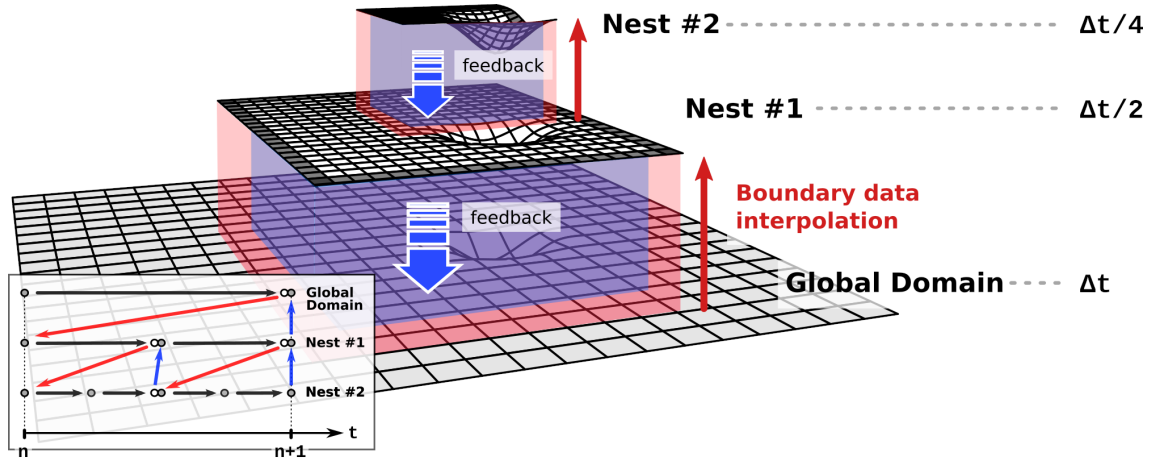


Figure 3.20.: Schematic of a global domain with two repeatedly nested domains (two-way). The processing sequence for the time integration of all domains from time step n to $n + 1$ is shown in the flowchart at the lower left. See the text for details.

blue regions show the boundary interpolation zones and feedback zones of the individual domains, respectively. The integration time step on the global domain is denoted by Δt . It is automatically reduced by a factor of 2 when moving to the next child grid level.

The processing sequence for the integration of all domains from time step n to $n + 1$ is shown in the flowchart at the lower left of Figure 3.20. The domains are ordered top down. Open and filled black dots show model states without and with feedback increments, black arrows indicate time integration, and red and blue arrows indicate lateral boundary data interpolation and feedback, respectively.

From an abstract point of view, the flow control of ICON's hierarchical nesting scheme is handled by a recursive subroutine that cascades from the global domain down to the deepest nesting level and for each domain calls the time stepping and the physics parameterizations in basically the same way as for the global domain. The basic processing sequence is as follows:

1. A single integration step with Δt is performed on the global domain which, results in an updated model state \mathcal{M}_p^{n+1} , indicated by an open black circle.
2. Boundary data are interpolated from the global domain to the first nested domain (red arrow), followed by an integration step on nested domain 1 over the time interval $\Delta t/2$.
3. As there exists another nested domain within nest 1, boundary fields based on the model state $\mathcal{M}_{c1}^{n+1/2}$ are interpolated to the second nested domain. Afterwards, the model is integrated on nested domain 2 over two times the time interval $\Delta t/4$, resulting in the model state $\mathcal{M}_{c2}^{n+1/2}$.
4. Feedback is performed from nest 2 back to nest 1 (blue arrow), which results in an updated model state $\mathcal{M}_{c1}^{n+1/2*}$ on nested domain 1 (black filled dot). Then, on nested domain 1 the model is again integrated in time to reach model state \mathcal{M}_{c1}^{n+1} .

5. This is followed by a second lateral boundary data interpolation from nest 1 to nest 2 based on \mathcal{M}_{c1}^{n+1} . Nest 2 is integrated in time again, to reach its state \mathcal{M}_{c2}^{n+1} .
6. As a final step, feedback is performed from nest 2 to nest 1, followed by feedback from nest 1 to the global domain.

3.9.3. Technical and Performance Aspects

Several measures are taken in order to optimize the computational efficiency of the nesting implementation.

In the model grids, grid points lying at or near the lateral boundary of a nested domain are shifted to the beginning of the index vector, ordered by their distance from the lateral boundary. This allows excluding boundary points from prognostic computations accessing non-existing neighbor points without masking operations. In the present implementation, the four outer cell rows constituting the boundary interpolation zone (see Figure 3.17), and the adjacent fifth one participate in the reordering. For additional information on the grid point reordering see Section 9.1 and in particular Figure 9.2.

The reordering makes use of the grid meta-data field `refin_c_ctrl` which counts the distance in units of cell rows (see Figure 2.12). Correspondingly, there are integer flag arrays for edges and vertices replicating the distance information from the lateral boundary. This distance information is extended to a larger number of grid rows in order to provide the geometric information needed for lateral boundary nudging. Moreover, the flag arrays signify grid points overlapping with a child domain, including a distinction between boundary interpolation points and interior overlap points.

Regarding distributed-memory (MPI) parallelization, the general strategy adopted in ICON is to distribute all model domains among all compute processors. As this implies that child grid points are in general owned by a different processor than the corresponding parent grid point, an intermediate layer having the resolution of the parent grid but the domain decomposition of the child grid is inserted in order to accommodate the data exchange required for boundary interpolation and feedback.

To reduce the amount of MPI communication for complex nested configurations, multiple nested domains at the same nesting level can be merged into one logical domain which is then not geometrically contiguous. This needs to be done during the grid generation process by indicating a list of domains via the grid generator's namelist parameter `merge_domain` (see Prill, 2020). The lateral boundary points belonging to all components of the merged domain are then collected at the beginning of the index vector. For all prognostic calculations, the multiple domains are treated as a single logical entity, and just the output files may be split according to the geometrically contiguous basic domains. As one-way and two-way nesting cannot be mixed within one logical domain, there may still be two logical domains on a given nest level.

To further optimize the amount of MPI communication, a so-called processor splitting is available that allows for executing several nested domains concurrently on processor subsets whose size can be determined by the user in order to minimize the ensuing load imbalance. This option is currently restricted to the step from the global domain to the first nesting level in order to keep the technical complexity at a manageable level.

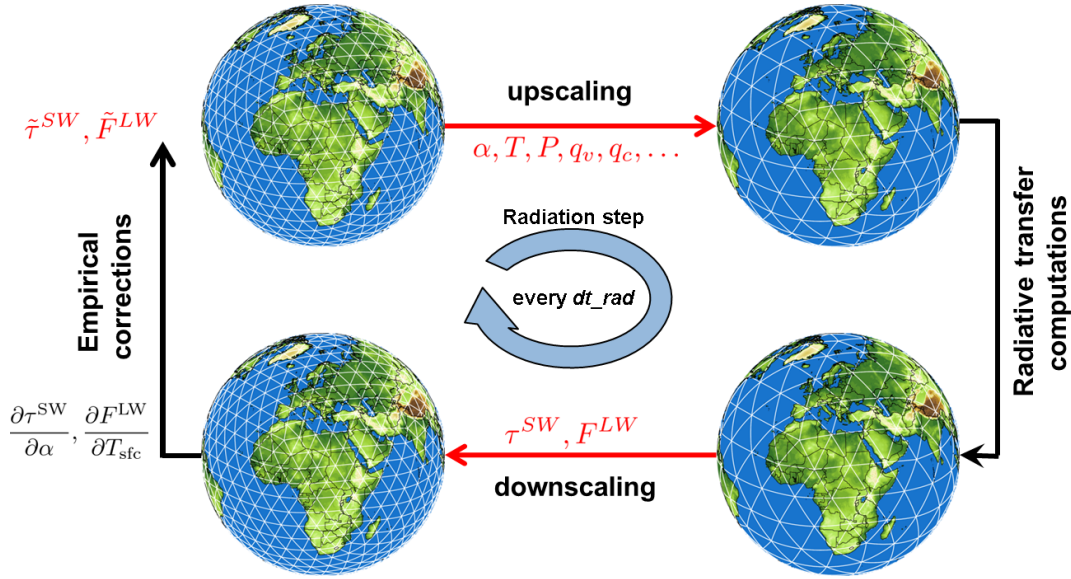


Figure 3.21.: Schematic showing how radiation is computed on a reduced (coarser) grid.

3.10. Reduced Radiation Grid

In real case simulations, radiation is one of the most time consuming physical processes. It is therefore desirable to reduce the computational burden without degrading the results significantly. One possibility is to use a coarser horizontal grid for radiation than for dynamics.

The implementation is schematically depicted in Figure 3.21:

- Step 1.* Radiative transfer computations are usually performed every 30 minutes. Before doing so, all input fields required by the radiation scheme are upscaled to the next coarser grid level.
- Step 2.* Then the radiative transfer computations are performed and the resulting short wave transmissivities τ^{SW} and longwave fluxes F^{LW} are scaled down to the full grid.
- Step 3.* In a last step we apply empirical corrections to τ^{SW} and F^{LW} in order to incorporate the high resolution information about albedo α and surface temperature T_{sfc} again. This is especially important at land-water boundaries and the snow line, since here the gradients in albedo and surface temperature are potentially large.

The reduced radiation grid is controlled with the following namelist switches:

lredgrid_phys = .FALSE./.TRUE. (namelist grid_nml, logical value)

If set to .TRUE. radiation is calculated on a coarser grid (i.e. one grid level coarser)

radiation_grid_filename (namelist grid_nml, string parameter)

Filename of the grid to be used for the radiation model. Must only be specified for the base domain, since for child domains the grid of the respective parent domain serves as radiation grid. An empty string is required, if radiation is computed on the full (non-reduced) grid.

Note that running radiation on a reduced grid is the standard setting for operational runs at DWD. Using the reduced radiation grid is also possible for the limited area mode ICON-LAM. In this case, both the computational grid and the reduced radiation grid are regional grids. Make sure to create the latter during the grid generation process by setting `dom(:)%lwrite_parent = .TRUE.`, see Section [2.1.5](#). Internally, the coarse radiation grid is denoted by the domain index 0.

4. Running Idealized Test Cases

The ability to run idealized model setups serves as a simple means to test the correctness of particular aspects of the model, either by comparison with analytic reference solutions (if they exist), or by comparison with results from other models. Beyond that, idealized test cases may help the scientist to focus on specific atmospheric processes.

ICON provides a set of pre-defined test cases of varying complexity and focus, ranging from pure dynamical core and transport test cases to “moist” cases, including microphysics and potentially other parameterizations. A complete list of available test cases can be found in the namelist documentation, mentioned in Section 1.1.2.

The majority of idealized test cases does not require external parameter or analysis fields for initialization. Initial conditions are usually computed from analytical functions within the ICON model itself. These are either evaluated point-wise at cell centers, edges, or vertices, or are integrated over the triangular control volume to provide cell averages.

Individual test cases can be selected and configured by namelist parameters of the namelist `nh_testcase_nml`. To run one of the implemented test cases, only a horizontal grid file has to be provided as input. A vertical grid file containing the height distribution of vertical model levels is usually not required, since the vertical grid is constructed within the ICON model itself, based on the set of namelist parameters described in Section 3.4.

From the set of available test cases we choose the Jablonowski-Williamson baroclinic wave test and Straka density current test and walk through the procedure of configuring and running these tests in ICON.

4.1. Main Switches for Idealized Test Cases

This section explains several namelist groups and main switches that are necessary for setting up an idealized model run.

4.1.1. Activating/De-activating Main Model Components

Namelist `run_nml`:

`ltestcase = .TRUE./.FALSE.` (namelist `run_nml`, logical value)

This parameter must be set to `.TRUE.` for running idealized test cases.

`ldynamics = .TRUE./.FALSE.` (namelist `run_nml`, logical value)

Main switch for the dynamical core. If set to `.TRUE.`, the dynamical core is

switched on and details of the dynamical core can be controlled via `dynamics_nml`, `nonhydrostatic_nml` and `diffusion_nml`. If set to `.FALSE.`, the dynamical core is switched off completely. This is rarely needed, but can be useful for idealized tests of physical parameterizations with prescribed dynamical forcing.

ltransport = `.TRUE./FALSE.` (namelist `run_nml`, logical value)

Main switch for the transport of scalars such as water constituents and passive tracers. If set to `.TRUE.`, transport is switched on and details of the transport schemes can be controlled via `transport_nml` (see Section 3.6.5 for additional help). If set to `.FALSE.`, transport of tracers is switched off completely.

iforcing = `0/2/3` (namelist `run_nml`, integer value)

Forcing of dynamics and transport by parameterized processes. If set to 0, forcing is switched off completely (pure dynamical core test case). This implies that all physical parameterizations are switched off automatically. If set to 3, dynamics are forced by NWP-specific parameterizations. Individual physical processes can be controlled via `nwp_phy_nml`, see also Table 3.4. If set to 2, the AES parameterization suite is used. In general, the setting of `iforcing` depends on the selected test case.

msg_level (namelist `run_nml`, integer value)

You may increase the model output verbosity by setting this namelist parameter to a higher value (≤ 20). This option can be particularly useful if the ICON model run fails and the cause of the error still does not become clear from the error message.

Namelist `dynamics_nml`:

lcoriolis = `.TRUE./FALSE.` (namelist `dynamics_nml`, logical value)

Main switch for activation/deactivation of the Coriolis force. In general, the setting depends on the selected test case.

Namelist `extpar_nml`:

itopo = `0/1` (namelist `extpar_nml`, integer value)

If set to 1, the model tries to read topography data and external parameters from file. If set to 0, no input file is requested for model initialization. Instead, all initial conditions are computed within the ICON model itself. Usually, `itopo` is set to 0 for idealized test cases.

4.1.2. Specifying the Computational Domain(s)

ICON's computational domain(s) is/are specified via the following namelist:

Namelist `grid_nml`:**dynamics_grid_filename (namelist `grid_nml`, list of string parameters)**

Here, the name(s) of the horizontal grid file(s) must be specified. For a global simulation without nests, of course, only a single filename is required. For a global simulation with multiple nests a filename must be specified for each domain. Note that each name must be enclosed by single quotation marks and that multiple names must be separated by a comma (see the examples below).

Namelist `run_nml`:**num_lev (namelist `run_nml`, list of integer value)**

Comma-separated list of integer values specifying the number of vertical full levels for each domain.

Example: Assuming that a global horizontal grid file is provided (named `icon_grid_0014_R02B05_G.nc`) the settings for a global run with 40 vertical levels are as simple as follows:

```
dynamics_grid_filename = 'icon_grid_0014_R02B05_G.nc'
num_lev = 40
```

The specific placement of vertical levels is determined by the namelist group `sleve_nml` if the SLEVE vertical coordinate is chosen, or by a user-defined vertical coordinate table in case of the hybrid Gal-Chen coordinate (see Section 3.4 for additional help).

4.1.3. Integration Time Step and Simulation Length

The integration time step and simulation length are defined via the following namelist:

Namelist `run_nml`:**dtime (namelist `run_nml`, real value)**

Time step in seconds (for the top-most domain). Note that it is *not* necessary to specify a time step for each domain. For each nesting level, the time step is automatically divided by a factor of two. More details on ICON's time step are given in Section 3.7.1.

nsteps (namelist `run_nml`, integer value)

Number of time steps. An alternative way for setting the simulation length is to specify the simulation start and end date, see Section 5.1.1.

Output is controlled by the namelist group `output_nml`. It is possible to define more than one output namelist and each output namelist has its own output file attached to it. The details of the model output specification are discussed in Section 7.

4.2. Jablonowski-Williamson Baroclinic Wave Test

In order to activate the Jablonowski-Williamson baroclinic wave test, select:

```
nh_test_name = 'jabw' (namelist nh_testcase_nml, string parameter)
```

The Jablonowski-Williamson baroclinic wave test (Jablonowski and Williamson, 2006) has become one of the standard test cases for assessing the quality of dynamical cores. The model is initialized with a balanced initial flow field. It comprises a zonally symmetric base state with a jet in the mid-latitudes of each hemisphere and a quasi realistic temperature distribution. Overall, the conditions resemble the climatic state of a winter hemisphere. This initial state is in hydrostatic and geostrophic balance, but is highly unstable with respect to baroclinic instability mechanisms. It should remain stationary if no perturbation is imposed.

To trigger the evolution of a baroclinic wave in the northern hemisphere, the initial conditions are overlaid with a weak (and unbalanced) zonal wind perturbation. The perturbation is centered at 20°E, 40°N. In general, the baroclinic wave starts growing observably around day 4 and evolves rapidly thereafter with explosive cyclogenesis around model day 8. After day 9, the wave train breaks (see Figure 4.1). If the integration is continued, additional deviations from the purely zonal flow become apparent especially near the pentagon points (see Section 2.1), which is an indication of spurious baroclinic instabilities triggered by numerical discretization errors. In general, this test has the capability to assess

- the diffusivity of a dynamical core,
- the presence of phase speed errors in the advection of poorly resolved waves,
- the strength of grid imprinting.

4.2.1. Recommended Namelist Settings

A complete list of the recommended namelist settings is given in Table 4.1. The three parameters listed below are specific to the Jablonowski-Williamson test case and are therefore explained in more detail. Default values are given in red.

jw_up = 1.0 (namelist nh_testcase_nml, real value)

Amplitude of the u-perturbation in m s^{-1} . If this parameter is set to 0, the model's ability to maintain the initial steady state can be tested.

jw_u0 = 35.0 (namelist nh_testcase_nml, real value)

Maximum zonal wind in m s^{-1}

jw_temp0 = 288.0 (namelist nh_testcase_nml, real value)

Horizontal-mean temperature at the surface in K

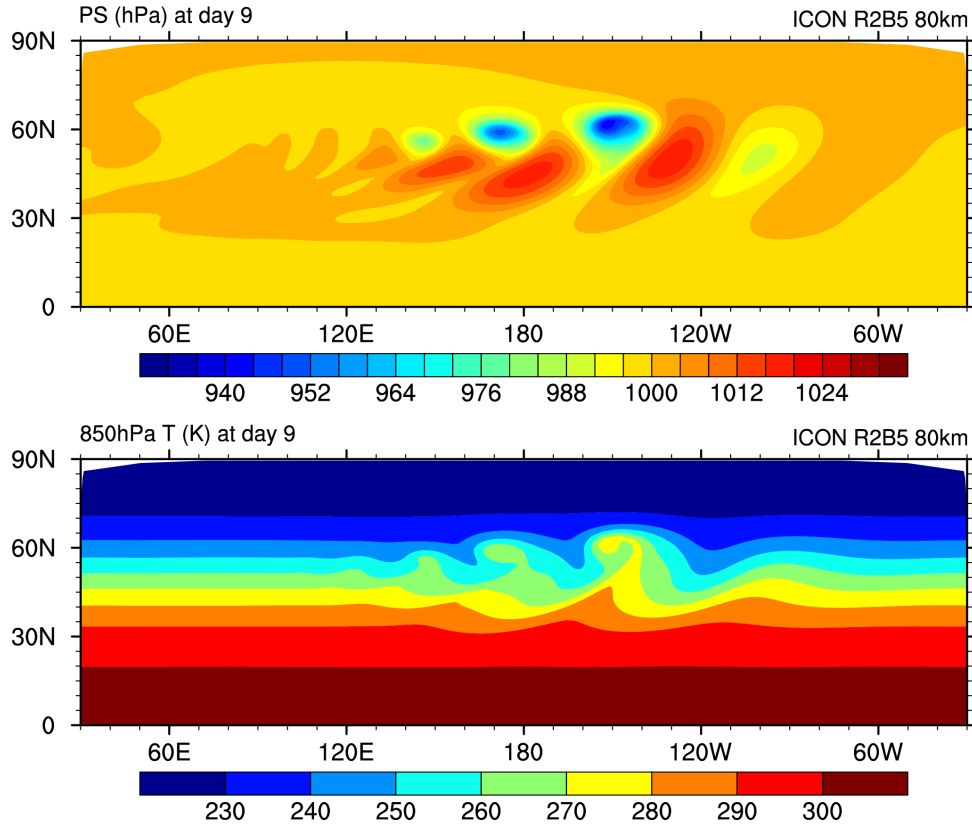


Figure 4.1.: Surface Pressure and 850hPa Temperature at day 9 for the Jablonowski-Williamson test case on a global R2B5 grid.

4.2.2. Enabling Passive Tracers

Jablonowski et al. (2008) suggest to add a variety of passive tracers to the baroclinic wave test, in order to investigate the general behavior of the advection algorithm. Questions that can be addressed are

- whether the advection scheme is monotone or positive-definite,
- how accurate or diffusive the advection scheme is,
- whether an initially constant tracer distribution is preserved (which checks for tracer-air mass consistency).

In the ICON code, four different tracer distributions are implemented, whose initial distributions are depicted in Figure 4.2. See Jablonowski et al. (2008) for further information on the initial distributions.

Please follow the steps below, in order to enable the transport of one or more predefined tracers:

- Enable the transport module by activating the main switch `ltransport=.TRUE.` (namelist `run_nml`).

Namelist	Parameter	Unit	Value
nh_testcase_nml	nh_test_name		'jabw'
	jw_up	m/s	1.0
	jw_u0	m/s	35.0
	jw_temp0	K	288.0
run_nml	ltestcase		.TRUE.
	ldynamics		.TRUE.
	ltransport		.FALSE.
	iforcing		0
	num_lev		40
	dtime	s	576
	nsteps		1500
dynamics_nml	lcoriolis		.TRUE.
extpar_nml	itopo		0
grid_nml	dynamics_grid_filename		'icon_grid014_R02B05_G.nc'
sleve_nml	top_height	m	35000
nonhydrostatic_nml	vwind_offctr		0.2
	exner_expol		0.5
	damp_height	m	25000
	rayleigh_coeff		0.1
diffusion_nml	hdiff_order		5
	itype_vn_diffu		2
	itype_t_diffu		2
	hdiff_efdt_ratio		20

Table 4.1.: Recommended namelist settings for the global Jablonowski-Williamson baroclinic wave test case at ≈ 80 km (R2B05) horizontal resolution.

- Select one or more tracers from the set of pre-defined tracer distributions depicted in Figure 4.2. A specific tracer is selected by adding the respective tracer number (1,2,3, or 4) to the following namelist variable:

tracer_inidist_list(namelist nh_testcase_nml, list of integer values)

Comma-separated list of integer values. A value of 1 selects tracer q_1 , 2 selects q_2 , and so on. If the list is empty, no passive tracer will be transported.

- Set the total number of tracers **ntracer** (namelist **run_nml**) accordingly.

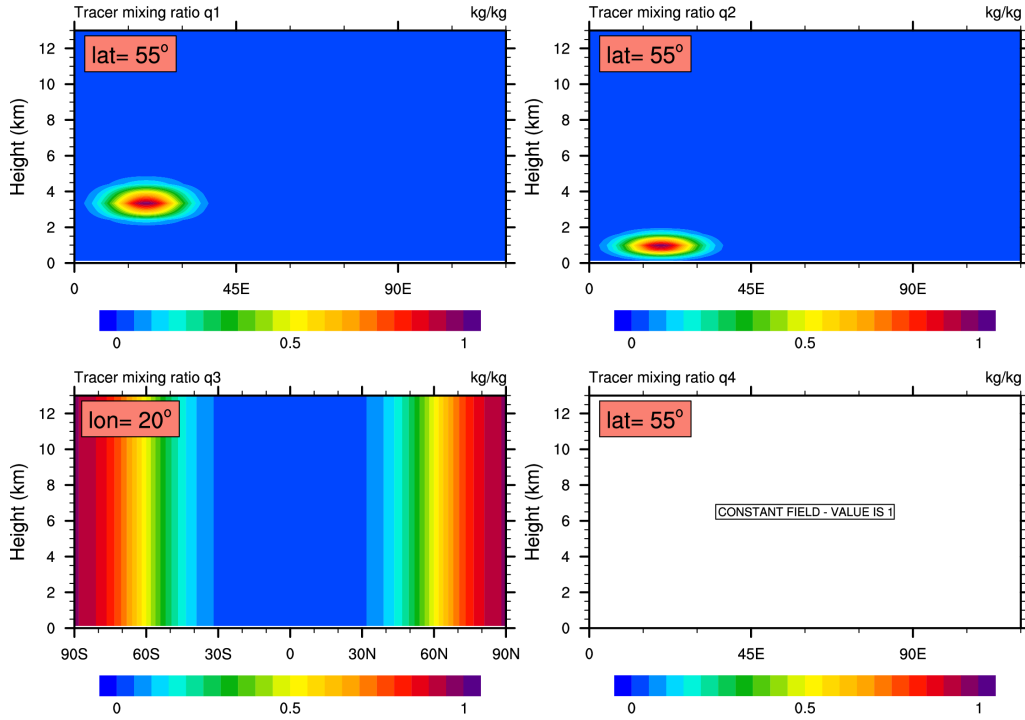


Figure 4.2.: Initial tracer distributions which are available for the Jablonowski-Williamson test case. Tracer $q3$ only depends on the latitudinal position, and tracer $q4$ is constant.

- Add the selected tracers to the list of output fields in the namelist `output_nml` (namelist parameters `ml_varlist` and/or `pl_varlist`).

By default, tracers in idealized tests are named qx , where x is a number indicating the position of the tracer within the ICON-internal 4D tracer container. For this testcase, tracer 1 is named $q1$, and so on.

Alternatively, the default names can be overwritten via the namelist variable `tracer_names` (namelist `transport_nml`, comma-separated list of string parameters). The n^{th} entry in `tracer_names` corresponds to the n^{th} entry in `tracer_inidist_list`.

4.2.3. Activation of Nested Domains

The Jablonowski-Williamson test is well suited to acquaint oneself with ICON's nesting capability, which is described in Section 3.9. The test is used e.g. by Zängl et al. (2022) in order to demonstrate the functionality of the grid nesting and to investigate related numerical errors.

Activating nested domains requires only a small number of additional namelist settings. In the following, we assume that three horizontal grid files are given

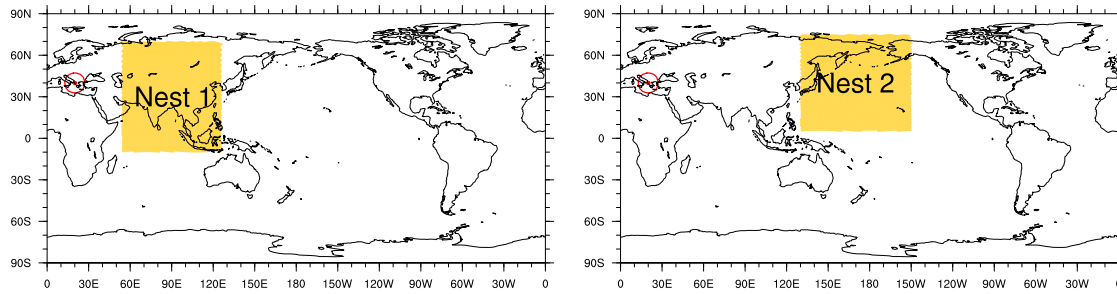


Figure 4.3.: Suggested location of nests for the baroclinic wave test case. The zonal wind perturbation triggering the baroclinic wave is centered at (20°E, 40°N) (red circle).

```
icon_grid_0014_R02B05_G.nc
icon_grid_0014_R02B05_N06_1.nc
icon_grid_0014_R02B05_N06_2.nc,
```

which contain a global grid and two nested grids at the same nesting level, respectively (see Figure 4.3 for the suggested location and extent of the nested domains). Additional information on the grid file naming convention can be found in Section 2.1. One or both nested domains may be activated by namelist settings:

Example 1: Settings for a global run with nest number 2 and 40 vertical levels each:

```
dynamics_grid_filename =
'icon_grid_0014_R02B05_G.nc', 'icon_grid_0014_R02B05_N06_2.nc'
num_lev = 40,40
```

Example 2: Settings for a global run with both nests and 40 vertical levels each:

```
dynamics_grid_filename =
'icon_grid_0014_R02B05_G.nc', 'icon_grid_0014_R02B05_N06_1.nc',
'icon_grid_0014_R02B05_N06_2.nc'
num_lev = 40,40,40
```

The parent-child relationships of the individual domains/nests are inferred automatically from the NetCDF attributes `uuidOfHGrid` and `uuidOfParHGrid` in the grid files, see Section 3.9.

4.3. Straka Density Current Test

Another well-known test case for the evaluation and intercomparison of dynamical cores is the nonlinear 2D density current test case described by [Straka et al. \(1993\)](#). See e.g. [Gallus and Rančić \(1996\)](#), [Satoh \(2002\)](#), [Skamarock and Klemp \(2008\)](#), [Guerra and Ullrich \(2016\)](#) for example applications.

In this test case a circular shaped bubble of cold air is initialized a few kilometers above ground in a neutrally stratified ($\theta = 300$ K) and hydrostatically balanced atmosphere at

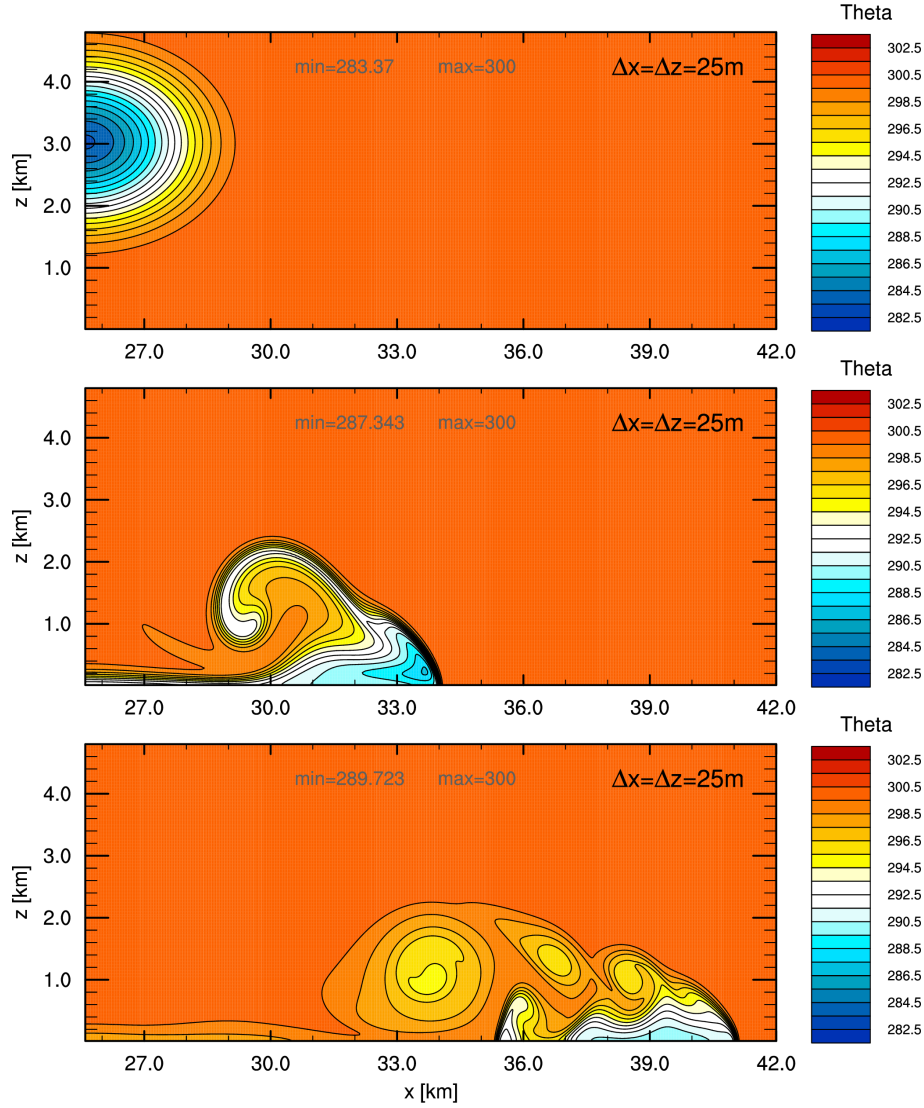


Figure 4.4.: Straka density current test case reference results for ICON with $\Delta x = \Delta z \approx 25$ m. Contours show the potential temperature for $t = 0$ min, 8 min, 15 min, respectively. The contour interval is 1 K. Due to the symmetry of the setup only the right moving density current is shown.

rest. The Coriolis force is set to zero. When integrated forward in time, the cold air bubble accelerates towards the ground and forms two symmetric density currents which spread laterally along the bottom boundary and form shear-driven Kelvin-Helmholtz (K-H) instabilities along their top (see Figure 4.4). The simulation is scale limited due to the application of a second order diffusion operator for potential temperature and momentum with constant diffusion coefficients $K_h = K_m = 75 \text{ m}^2 \text{ s}^{-1}$. The resolvable scales are, hence, limited by the viscosity of the simulated medium rather than the spatio-temporal resolution. This enables the computation of a grid-converged reference solution. [Straka et al. \(1993\)](#) showed that for $\Delta x = \Delta z \approx 25$ m the result can be regarded as grid-converged, since any additional resolution increase does not have a noticeable effect. For visual inter-

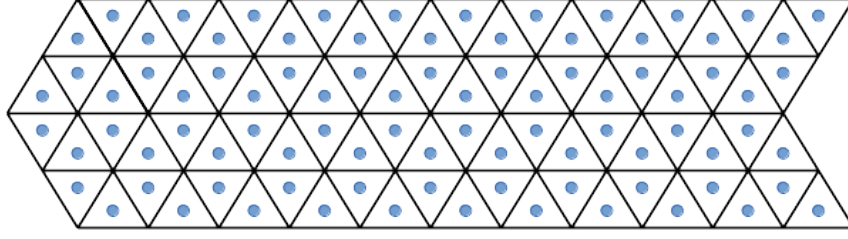


Figure 4.5.: Schematic of the quasi-2D Straka test torus grid, which consists of 4 cell rows in meridional direction. Due to technical reasons it is currently not possible to further reduce the number of rows.

comparison [Straka et al. \(1993\)](#) provided reference solutions of various dynamical cores. The $\Delta x = \Delta z \approx 25$ m reference solution of ICON is shown in Figure 4.4.

The computational domain consists of a quasi 2D torus grid (see Section 2.1.8) with doubly-periodic boundary conditions and a width of (at least) 40 km in zonal direction¹. In meridional direction, the domain consist of 4 cell rows (see Figure 4.5). Due to technical reasons it is not possible to further reduce the number of rows, however, the dynamical core gives identical results for each of these rows. The domain height is set to $H = 6400$ m, and the upper and lower boundary are treated as rigid lid (no flux).

The initial temperature disturbance is applied to the θ field and is given by

$$\Delta T = \begin{cases} 15.0 \cos^2\left(\frac{\pi}{2}L\right) & , \text{ if } L \leq 1 \\ 0.0 & , \text{ if } L > 1, \end{cases}$$

with

$$L = \left[\left(\frac{x - x_c}{x_r} \right)^2 + \left(\frac{z - z_c}{z_r} \right)^2 \right]^{\frac{1}{2}}.$$

The cold bubble is initially located at $(x_c, z_c) = (0 \text{ km}, 3 \text{ km})$ and has a radius of $(x_r, z_r) = (4 \text{ km}, 2 \text{ km})$.

In general, this test can be used to assess (among other things):

- the order of convergence of the dynamical core
- the quality at resolutions much coarser than $\Delta x = \Delta z \approx 25$ m. I.e. which resolution is necessary in order to resolve all three K-H rotors?
- the magnitude of phase speed errors, by adding a nonzero background wind and comparing the right- and left moving currents (see [Skamarock and Klemp \(2008\)](#)).

A complete list of the recommended namelist settings is given in Table 4.2.

¹Results are usually compared after $t = 15$ min simulation time, when the wave front has traveled approximately 15 km in both directions. Hence, a domain width of 40 km should be sufficient to avoid significant disturbances along the lateral boundaries.

4.3.1. Relevant Namelist Switches in `nh_testcase_nml`:

We conclude this section by a list of parameters that can be used to modify the Straka setup. Again, default values are given in **red**. They do not necessarily coincide with the recommended Straka settings (see Table 4.2).

bub_hor_width = 1000.0 (namelist `nh_testcase_nml`, real value)

Horizontal radius of the thermal perturbation in m

bub_ver_width = 1400.0 (namelist `nh_testcase_nml`, real value)

Vertical radius of the thermal perturbation in m

bubctr_z = 1400.0 (namelist `nh_testcase_nml`, real value)

Height of the center of the thermal perturbation in m

bub_amp = 2.0 (namelist `nh_testcase_nml`, real value)

Maximum amplitude of the center of the thermal perturbation in K

nh_brunt_vais = 0.01 (namelist `nh_testcase_nml`, real value)

Initial Brunt-Väisälä frequency (constant with height) in s^{-1}

nh_u0 = 0.0 (namelist `nh_testcase_nml`, real value)

Initial constant zonal wind speed. Can be used to break the symmetry of the Straka test case (see Section 4.3).

Namelist	Parameter	Unit	Value
nh_testcase_nml	nh_test_name		'straka93'
	nh_brunt_vais	s^{-1}	0.0
	bubctr_z	m	3000
	bub_hor_width	m	4000
	bub_ver_width	m	2000
	bub_amp	K	-15
run_nml	ltestcase		.TRUE.
	ldynamics		.TRUE.
	ltransport		.FALSE.
	iforcing		3
	num_lev		256
	dttime	s	0.18
	nsteps		6800
dynamics_nml	lcoriolis		.FALSE.
extpar_nml	itopo		0
grid_nml	dynamics_grid_filename		'plane-grid_1600_dx25.0.nc'
	is_plane_torus		.TRUE.
sleve_nml	top_height	m	6400
	min_lay_thckn		0.0
diffusion_nml	hdiff_order		3
	hdiff_efdt_ratio		10
	hdiff_smag_fac		0.12
turbdiff_nml	lconst_z0		.TRUE.
	const_z0	m	0.0003
nwp_phy_nml	inwp_turb		5
les_nml	is_dry_cbl		.TRUE.
	isrfc_type		0
	ufric	ms^{-1}	0.0
	smag_coeff_type		2
	Km_ext	m^2s^{-1}	75.0
	Kh_ext	m^2s^{-1}	75.0

Table 4.2.: Recommended namelist settings for the Straka density current test case at 25 m horizontal resolution on a planar torus grid. Please note that all NWP physics parameterizations must be switched off in `nwp_phy_nml`, except for turbulence (`inwp_turb`).

5. Running Real Data Cases

In this chapter you will learn about how to initialize and run the ICON model in a realistic NWP setup. The namelist settings to start from a DWD Analysis and from an IFS Analysis are discussed.

5.1. Model Initialization

The necessary input data to perform a real data run have already been described in Chapter 2. These include

- grid files, containing the horizontal grid information,
- external parameter files, providing information about the Earth's soil and land properties, as well as climatologies of atmospheric aerosols, and
- initial data (analysis) for atmosphere, land and sea.

ICON is capable of reading analysis data from various sources (see Section 2.2), including data sets generated by DWD's Data Assimilation Coding Environment (DACE) and interpolated IFS data. In the following we provide some guidance on how to set up real data runs, depending on the specific data set at hand.

Note that ICON aborts during the setup phase, if any of the required input files has not been found. Therefore, as a first step, check the filenames (and soft links) for the model input files (see Section 2).

Also make sure that the input data and grid files match. For example, take a look at the global attributes `number_of_grid_used` and `uuidOfHGrid` of the grid file(s). These values have to match the corresponding attributes of the external parameters and initial data file(s), see Section 2.1.7.

5.1.1. Basic Settings for Running Real Data Runs

Most of the main switches that were used for setting up idealized test cases, are also important for setting up real data runs. As many of them have already been discussed in Chapter 4, we will concentrate on their settings for real data runs. Settings appropriate for the exercises on this subject are highlighted in red.

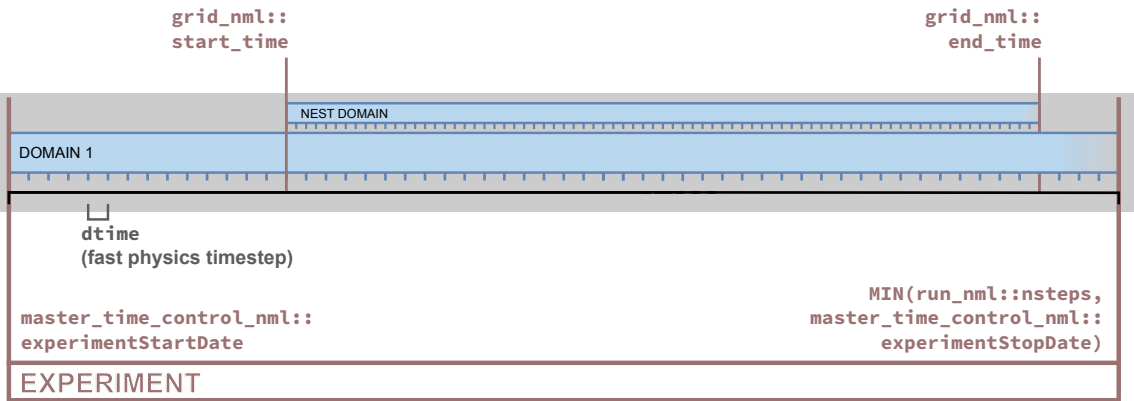


Figure 5.1.: Graphical illustration of the parameters for model start and end. The starting and termination of nested domains is explained in Section 5.2. Please also note Fig. 7.3, in which the program sequence is extended by restart.

Specifying Model Start and End Dates (Namelist `time_nml`)

For real case runs it is important that the user specifies the correct start date and time of the simulation, see Fig. 5.1.

In ICON there coexist two equally usable ways to control the experiment start and end date – without a compelling reason, though. These two alternatives are listed in the following.

Namelist `run_nml`:

time step	<code>dtime</code>	<code>modelTimeStep</code>
-----------	--------------------	----------------------------

Namelist `time_nml` (left) and `master_time_control_nml` (right):

experiment start	<code>ini_datetime_string</code>	<code>experimentStartDate</code>
experiment stop	<code>end_datetime_string</code>	<code>experimentStopDate</code>

Please note that the data types of the above-mentioned namelist parameters differ. The parameters that are listed on the right are consistently based upon the ISO 8601 representations of dates and time spans. However, `dtime` must be specified in seconds.

In the examples of this tutorial, start and end dates are given with `ini_datetime_string` using the ISO8601 format:

`ini_datetime_string = YYYY-MM-DDThh:mm:ssZ (namelist time_nml)`
— This must exactly match the validity time of the analysis data set!

Wrong settings lead to incorrect solar zenith angles and wrong external parameter fields. Setting the end date and time of the simulation via `end_datetime_string` is optional. If `end_datetime_string` is not set, the user has to set the number of time steps explicitly in `nsteps (run_nml)`, which is otherwise computed automatically.

General Settings (Namelist `run_nml`)

`ltestcase= .FALSE.` (namelist `run_nml`, logical value)
This parameter must be set to `.FALSE.` for real case runs.

iforcing= 3 (namelist run_nml, integer value)

A value of 3 means that dynamics are forced by NWP-specific parameterizations.

ldynamics= .TRUE. (namelist run_nml, logical value)

The dynamical core must, of course, be switched on.

ltransport= .TRUE. (namelist run_nml, logical value)

Tracer transport must be switched on. This is necessary for the transport of cloud and precipitation variables. Details of the transport schemes can be controlled via the namelist `transport_nml` (see Section 3.6.5).

Specifying the Horizontal Grid (Namelist `grid_nml`)

dynamics_grid_filename (namelist grid_nml, list of string parameters)

Here, the name(s) of the horizontal grid file(s) must be specified. For a global simulation without nests, of course, only a single filename is required. For a global simulation with multiple nests, a filename must be specified for each domain. Note that each name must be enclosed by single quotation marks and that multiple names must be separated by a comma (see Section 4.1.2 and the examples therein).

radiation_grid_filename (namelist grid_nml, string parameter)

If the radiative transfer computation should be conducted on a coarser grid than the dynamics (one level coarser, effective mesh size $2\Delta x$), the name of the base grid for radiation must be specified here. See Section 3.10 for further details.

Specifying External Parameters (Namelist `extpar_nml`)

itopo= 1 (namelist extpar_nml, integer value)

For real data runs this parameter must be set to 1. The model now expects one file per domain from which it tries to read topography data and external parameters.

extpar_filename (namelist extpar_nml, string parameter)

Filename(s) of input file(s) for external parameters. If the user does not provide namelist settings for `extpar_filename`, ICON expects one file per domain to be present in the experiment directory, following the naming convention

```
extpar_filename = "extpar_<gridfile>.nc"
```

The keyword `<gridfile>` is automatically replaced by ICON with the grid filename specified for the given domain (`dynamics_grid_filename`). As opposed to the grid-file specification namelist variables (see above), it is not allowed to provide a comma-separated list. Instead, the usage of keywords provides full flexibility for defining the filename structure.



By changing the above setting, the user has full flexibility with respect to the filename structure. The following keywords are allowed for the namelist parameter `extpar_filename`. The keywords are automatically replaced by ICON with the content described in the right column below.

<code><path></code>	model base directory (namelist parameter <code>model_base_dir</code> , namelist <code>master_nml</code>)
<code><gridfile></code>	grid filename for the given domain (<code>dynamics_grid_filename</code>)
<code><nroot></code>	grid root division <code>Rx</code> (single digit)
<code><nroot0></code>	grid root division <code>Rxx</code> (two digits)
<code><jlev></code>	grid bisection level <code>Byy</code> (two digits)
<code><idom></code>	domain number (two digits).

Specifying the Initialization Mode (Namelist `initicon_nml`)

ICON provides different real data initialization modes which differ in terms of the expected input fields and number of input files. Thereby ICON is able to handle analysis products from different models. The mode in use is controlled via the namelist switch `init_mode`.

init_mode (namelist `initicon_nml`, integer value)

It is possible to

- start from (interpolated) *uninitialized* DWD analysis without the IAU procedure: `init_mode = 1`
- start from interpolated IFS analysis: `init_mode = 2`
- start atmosphere from interpolated IFS analysis and soil/surface from interpolated ICON/GME fields: `init_mode = 3`
- start from non-interpolated, *uninitialized* DWD analysis, and make use of the IAU procedure to filter initial noise: `init_mode = 5`
- start from interpolated *initialized* ICON analysis with subsequent vertical remapping: `init_mode = 7`

The most relevant modes are mode 1, 2, 5 and 7. These will be explained in more detail below.

ICON supports NetCDF and GRIB2 as input format for input fields. In this context it is important to note that the field names that are used in the input files do not necessarily coincide with the field names that are internally used by the ICON model. To address this problem, an additional input text file is provided, a so-called *dictionary file*. This file translates between the ICON variable names and the corresponding GRIB2/NetCDF short names.

Generally the dictionary is provided via the following namelist parameter:

ana_varnames_map_file (namelist initicon_nml, string parameter)

Filename of the dictionary for mapping between internal names and GRIB2/NetCDF short names. An example can be found in `icon/run/ana_varnames_map_file.txt`.



The ICON model contains different map files, sometimes also called dictionaries. These plain text files translate between ICON internal variable names and GRIB2/NetCDF short names. There are several dictionaries that can be specified in the namelists of ICON.

ana_varnames_map_file (namelist initicon_nml, string parameter)

Used by the input module for first guess and analysis files.

Left column: ICON internal name; *Right column:* input name, e.g. GRIB2 short name

latbc_varnames_map_file (namelist limarea_nml, string parameter)

Used by the module for lateral boundary conditions.

Left column: ICON internal name; *Right column:* input name, e.g. GRIB2 short name

extpar_varnames_map_file (namelist extpar_nml, string parameter)

Used when reading the external parameter file `extpar_filename`. This dictionary is practically not used as external parameters are read directly using the NetCDF library.

output_nml_dict (namelist io_nml, string parameter)

Used by the namelist output module. This dictionary allows to use GRIB2 short names instead of ICON internal names in the output namelists of ICON.

Left column: namelist variable name; *Right column:* ICON internal name
A reasonable choice is to use the same table as for reading the initial data with inverted columns. This can be achieved with the namelist parameter `linvert_dict (io_nml)`.

netcdf_dict (namelist io_nml, string parameter)

The NetCDF dictionary can be used when writing to NetCDF files where variable short names can be chosen freely (in contrast to GRIB2). This allows to adopt ICON output variable names to already available post-processing scripts.

5.1.2. Starting from Uninitialized DWD Analysis

This analysis product is rarely the optimal choice for model initialization, as it generates a significant amount of spurious noise during the first few hours of a model run (see

Figure 2.8). Nevertheless, this mode is described for completeness. The process of obtaining the uninitialized DWD analysis for non-incremental update is described in Section 2.2.1.

Model initialization is basically controlled by the following four namelist parameters:

init_mode = 1 (namelist initicon_nml, integer value)

To start from uninitialized DWD analysis data (without incremental analysis update), the initialization mode must be set to 1.

lread_ana (namelist initicon_nml, logical value)

By default, this namelist parameter is set to .TRUE..

For `lread_ana=.TRUE.` the ICON model expects two input files per domain. One containing the ICON first guess (3 h forecast) fields, which served as background fields for the assimilation process. The other contains the analysis fields produced by the assimilation process. ICON reads the fields from the first guess file and replaces these subsequently with fields from the analysis file, where available. See Table 11.2 for a list of variables.

dwdfg_filename (namelist initicon_nml, string parameter)

Filename of the DWD first guess input file.

dwdana_filename (namelist initicon_nml, string parameter)

Filename of the DWD analysis input file.

Remember to make sure that the validity date for the first guess and analysis input file is the same and matches the model start date given by `ini_datetime_string`.

Input filenames need to be specified unambiguously, of course. By default, if the user does not provide namelist settings for `dwdfg_filename` and `dwdana_filename`, the filenames have the form

```
dwdfg_filename = "dwdFG_R<nroot>B<jlev>_DOM<idom>.nc"
dwdana_filename = "dwdANA_R<nroot>B<jlev>_DOM<idom>.nc"
```

This means, e. g., that the first guess filename begins with “dwdFG_”, supplemented by the grid spacing `RxxByy` and the domain number `DOMii`. Filenames are treated case sensitively.¹



By changing the above setting, the user has full flexibility with respect to the filename structure. The following keywords are allowed for the namelist parameters `dwdfg_filename`, `dwdana_filename` and `ifs2icon_filename` (for the latter see Section 5.1.5):

<code><path></code>	model base directory (namelist parameter <code>model_base_dir</code> , namelist <code>master_nml</code>)
<code><nroot></code>	grid root division <code>Rx</code> (single digit)
<code><nroot0></code>	grid root division <code>Rxx</code> (two digits)
<code><jlev></code>	grid bisection level <code>Byy</code> (two digits)
<code><idom></code>	domain number (two digits).

¹More precisely this behavior depends on the file system: UNIX-like file systems are case sensitive, but the HFS+ Mac file system (usually) is not.

For `lread_ana=.FALSE.`, only `dwdfg_filename` has to be specified. This can be used to run ICON from a first guess file. The combination of `init_mode=1` and `lread_ana=.FALSE.`, however, is rarely used. There is an overlap in possible applications with `init_mode=7`, which is usually the better choice due to the flexibility given by the vertical remapping.

5.1.3. Starting from Uninitialized DWD Analysis for IAU

As will be described in Section 11.3, IAU is a means to reduce the initial noise which typically results from small scale non-balanced modes in the analysis data set. Combining this analysis product with IAU is the preferred method in cases where the horizontal and vertical grid of the intended forecast run exactly match with that of the analysis. Since no horizontal interpolation is required, the forecast run can make use of the surface tile information which is specific to this analysis product. Moreover, this product exhibits the smallest noise level during model start (see Figure 2.8).

The process of obtaining the uninitialized analysis for IAU is described in Section 2.2.1.

Model initialization is basically controlled by the following namelist parameters:

init_mode = 5 (namelist initicon_nml, integer value)

To start from DWD analysis data with IAU, the initialization mode must be set to 5.

ICON again expects two input files. One containing the ICON first guess, which typically consists of a 1.5 h forecast taken from the assimilation cycle (as opposed to a 3 h forecast used for the non-IAU case). The other file contains the analysis fields (mostly increments) produced by the assimilation process. See Table 11.1 for a full list of variables.

dwdfg_filename (namelist initicon_nml, string parameter)

Filename(s) of the DWD first guess input file(s) for each domain. See Section 5.1.2 for an explanation of the filename structure.

dwdana_filename (namelist initicon_nml, string parameter)

Filename(s) of the DWD analysis input file(s) for each domain. See Section 5.1.2 for an explanation of the filename structure.

The behavior of the IAU procedure is controlled via the namelist switches `dt_iau` and `dt_shift`:

dt_iau = 10800 (namelist initicon_nml, real value)

Time interval (in s) during which the IAU procedure (i.e. dribbling of analysis increments) is performed.

dt_shift = -5400 (namelist initicon_nml, real value)

Time interval (in s) by which the model start is shifted ahead of the nominal model start date given by `ini_datetime_string`. Typically `dt_shift` is set to $-0.5 * dt_iau$ such that dribbling of the analysis increments is centered around `ini_datetime_string`.

As explained in Section 11.3.2 and depicted in Figure 11.2, you have to make sure that the first guess is shifted ahead in time by $-0.5 * dt_iau$ w.r.t. the analysis. The model start time `ini_datetime_string` must match the validity time of the analysis.

5.1.4. Starting from Initialized DWD Analysis

The initialized analysis is the product of choice in cases where the horizontal and/or vertical grid of the intended model run differs from that of the analysis. Using the uninitialized analysis for IAU is prohibited in such cases, as the horizontal interpolation of tiled surface fields makes no sense. Moreover, the initialized analysis product is less cumbersome to use, as it consists of a single file per domain, only. When compared to the standard uninitialized analysis product, spurious noise is significantly reduced (see Figure 2.8).

The process of obtaining the initialized analysis is described in Section 2.2.1.

Model initialization is basically controlled by the following namelist parameters:

init_mode = 7 (namelist initicon_nml, integer value)

To start from initialized DWD analysis data, the initialization mode must be set to 7.

If the number and or heights of the vertical levels differs between the model and the analysis, the input fields are automatically remapped in the vertical during read-in.

ICON expects a single input file. See Table 11.3 for a full list of variables.

dwdfg_filename (namelist initicon_nml, string parameter)

Filename(s) of the initialized DWD analysis input file(s) for each domain. Admittedly, the nomenclature “dwdfg” is a bit counter intuitive, as the file contains the full analysis rather than the first guess. See Section 5.1.2 for an explanation of the filename structure.

Remember to make sure that the model start time given by `ini_datetime_string` matches the validity date of the input file.

5.1.5. Starting from IFS Analysis

No filtering procedure is currently available when starting off from interpolated IFS analysis data. The model just reads in the initial data from a single file and starts the forecast.

The process of obtaining the IFS analysis and its content is described in Section 2.2.2.

init_mode= 2 (namelist initicon_nml, integer value)

To start from interpolated IFS analysis data, the initialization mode must be set to 2.

Note that for this initialization mode only input data in NetCDF format are supported and the specification of a dictionary file is not possible.

ifs2icon_filename (namelist initicon_nml, string parameter)

ICON expects a single file per domain from which interpolated IFS analysis can be read. With this parameter, the filename can be specified. Similar to the namelist parameters `dwdfg_filename` and `dwdana_filename`, which have been explained above in Section 5.1.2, the filenames have the form

```
ifs2icon_filename = "ifs2icon_R<nroot>B<jlev>_DOM<idom>.nc"
```

Remember to make sure that the model start time given by `ini_datetime_string` matches the validity date of the analysis input file.

5.2. Starting or Terminating Nested Domains at Runtime

Starting or terminating nested domains at runtime is possible by means of the namelist parameters `start_time` and `end_time` in the namelist `grid_nml`. Model calculations for the nested domain are performed if the simulation time of the parent domain is greater or equal to `start_time` and less than `end_time`. The settings are graphically illustrated in Fig. 5.1.

start_time (namelist grid_nml, list of real values)

Comma-separated list of real values. For each domain, the start time relative to the experiment start date can be specified in seconds. A value of 0 for the i th domain means that it is started at experiment start date which is either defined by `ini_datetime_string` or `experimentStartDate`. If Incremental Analysis Update (IAU) is used, `start_time` must be set equal to `dt_shift (initicon_nml)` (i.e. negative), in order for the nested domain to be active from the very beginning.

end_time (namelist grid_nml, list of real values)

Comma-separated list of real values. For each domain, the end time relative to the experiment start date can be specified in seconds. I.e. a value of 3600 specified for the i th domain means that it is terminated one hour after experiment start.

As discussed in Section 2.2, initial data files are usually required for each nested domain. With only little loss of forecast skill, this rather tedious procedure can be overcome by starting the nested domain(s) shortly after the global domain. In that case, nested domains are initialized by parent-to-child interpolation of the prognostic fields. Note, however, that surface tile information will be lost. Surface fields on the child domain are initialized with *aggregated* values interpolated from the parent domain.

6. Running ICON-LAM

The most important first: Running the limited area (regional) mode of ICON does not require a separate, fundamentally different executable. Instead, ICON-LAM is quite similar to the other model components discussed so far: It is easily enabled by a top-level namelist switch

```
Namelist grid_nml:      l_limited_area = .TRUE.
```

Other namelist settings must be added, of course, to make a proper ICON-LAM setup. This chapter explains some of the details.

Chapter Layout. Some of the pre-processing aspects regarding the regional mode have already been discussed in Section 2.3. Based on these prerequisites this chapter explains how to actually set up and run limited area simulations.

In the following, technical details on the limited area mode are provided, in particular on how to control the read-in of initial data and boundary data.

6.1. Limited Area Mode vs. Nested Setups

In Section 3.9.1 the nesting capability of ICON has been explained. Technically, the same computational grids may be used either for the limited area mode or the nested mode of ICON¹. Furthermore, both ICON modes aim at simulations with finer grid spacing and smaller scales. They therefore choose a comparable set of options out of the portfolio of available physical parameterizations.

However, there exist some differences between the regional and the one-way nested mode:

- ICON-LAM is driven by externally supplied boundary data which may come from a global model or a coarser resolution LAM that has been run in advance – that’s an obvious difference! During the simulation, boundary conditions are updated at regular time intervals by reading input files. Between two lateral boundary data samples the boundary data is linearly interpolated.
- Lateral boundary updates happen (significantly) less frequently compared to one-way nesting.
- The driving model may be different from the limited area model and may run on different computer sites. Both models may even differ in terms of the governing equations as well as numerical methods used.

¹Here, we do not take the reduced radiation grid into account, see Section 3.10. This serves to simplify the discussion at this point.

- ICON-LAM allows for a more flexible choice of vertical levels: Nested domains may differ from the global, “driving” grid only in terms of the top level height, but vertical layers must match between the nested and the parent domain (see Section 3.9.1). In contrast to that, the limited area mode performs a vertical interpolation of its boundary data. This is the default namelist parameter setting `itype_latbc=1` in the namelist `limarea_nml`. The level number and the level heights may therefore be chosen independently.
- ICON-LAM allows for a more flexible choice of the horizontal resolution. While for nested setups the increase in horizontal resolution per nesting level is constrained to a factor of 2, the resolution of the limited area domain can be freely selected. However, resolution jumps much larger than a factor of ~ 5 between the forcing data resolution and the target resolution should be avoided, since it will negatively impact the forecast quality.

6.2. Nudging in the Boundary Region

In order to prevent outward-propagating waves from reflecting back into the domain, a sponge layer is implemented along the lateral boundaries. Within this sponge layer the interior flow is relaxed towards externally specified boundary data. In addition to this lateral boundary nudging, upper boundary nudging along the model top can be switched on by choosing `nudge_type = 1` (namelist `nudging_nml`). The default value is 0, i.e. it is switched off. Figure 6.1 schematically depicts the partitioning of the limited area domain into the *lateral boundary zone*, labeled 0, the adjacent *lateral nudging zone* 1, the *upper nudging zone* 2, the *nudging overlap zone* 3 and the “free” *model atmosphere zone* 4. In the lateral boundary zone 0, which has a fixed width of 4 cell rows, externally supplied boundary data are prescribed.

The mathematical implementation of the sponge layer in the nudging zones follows the work by Davies (1976, 1983). An additional “forcing” term is added to the right hand side of the prognostic equations for v_n , θ_v , ρ , and q_v and is applied at each fast physics time step Δt :

$$\psi(t) = \psi^*(t) + \alpha_{\text{nudge}} \underbrace{[\psi_{\text{bc}}(t) - \psi^*(t)]}_{=\delta\psi}, \quad (6.1)$$

where ψ_{bc} is the externally specified value of the prognostic variable ψ at time t , and α_{nudge} is a dimensionless coefficient that controls the strength of the nudging. This coefficient gradually decreases with increasing distance from the boundary and is of the form

$$\alpha_{\text{nudge}} = \begin{cases} A_0 \exp\left(-\frac{|r-r_0|}{\mu}\right), & \text{if } r - r_0 \leq L \text{ (in region 1)} \\ B_0 \left(\frac{z-z_{\text{start}}}{z_{\text{top}}-z_{\text{start}}}\right)^2, & \text{in region 2} \\ \max\left\{A_0 \exp(\dots), B_0 (\dots)^2\right\}, & \text{in region 3} \\ 0, & \text{in region 4} \end{cases}, \quad (6.2)$$

with A_0 the maximum relaxation coefficient in the lateral nudging zone, L the width of the lateral nudging zone given in units of cell rows, μ the e-folding width given in

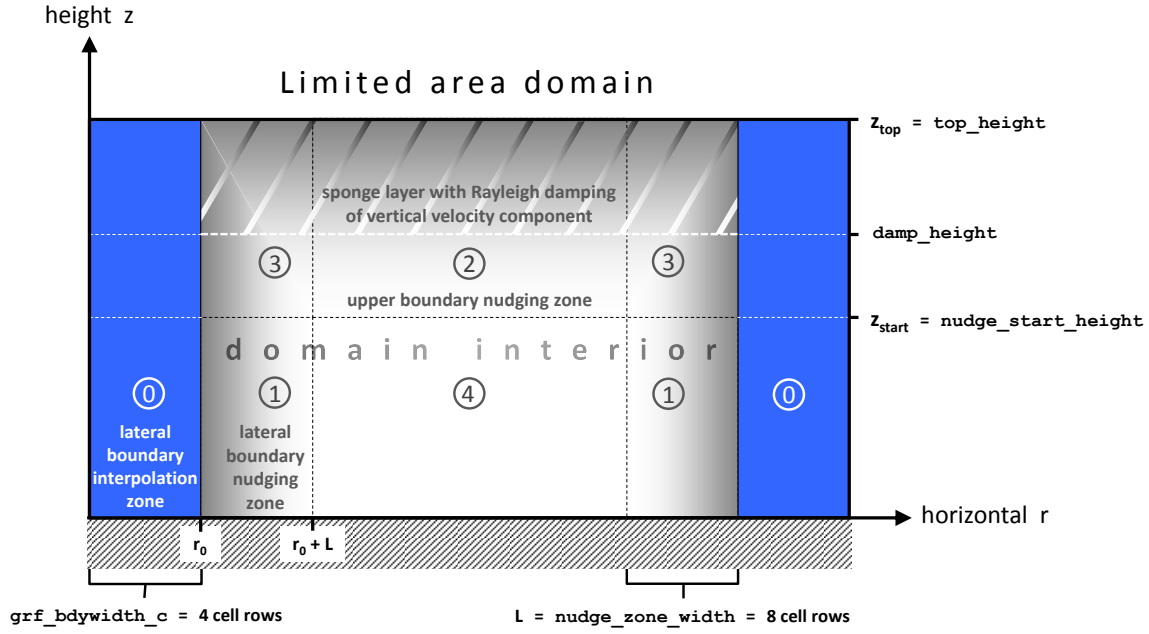


Figure 6.1.: Schematic illustration of the lateral boundary zone (blue) and the lateral and upper boundary nudging zones (gray) in the limited area mode.

units of cell rows, r the actual cell row index beginning with 1 in the outermost cell row of the boundary zone, and r_0 the cell row index at which the nudging zone starts (typically $\text{grf_bdywidth_c} + 1 = 5$, see Figure 6.1). The parameters L , μ , and A_0 can be specified via `nudge_zone_width`, `nudge_efold_width`, and `nudge_max_coeff` in the namelist `interp01_nml`. The nudge zone width should at least comprise 8 (better 10) cell rows in order to minimize boundary artifacts. For the variables v_n and q_v the parameter A_0 is multiplied by the factor 0.5.

B_0 is the maximum nudging coefficient in the upper boundary nudging zone between the model top at height z_{top} (`sleve_nml: top_height`) and the nudging start height z_{start} (`nudging_nml: nudge_start_height`). The value of B_0 is controlled by `max_nudge_coeff_vn` (`nudging_nml`) for the horizontal wind v_n , and `max_nudge_coeff_thermdyn` for θ_v and ρ .

Note that positive water vapor increments $\delta\psi = \delta q_v > 0$ are cut to zero in supersaturated regions ($q_c > 0$) in the lateral boundary nudging zone, in order to avoid an undesirable positive feedback on the growth of the amount of cloud water. In addition, water vapor is not subject to nudging in the upper boundary nudging zone. If the nudging data from the driving model contain hydrostatic variables (i.e. hydrostatic pressure), it might be more consistent to formulate the nudging in terms of the basic hydrostatic variables: pressure and temperature. This option is controlled by the namelist switch `nudge_hydro_pres` (`limarea_nml`), which applies to both lateral and upper boundary nudging. If set to `.TRUE.` (default), nudging increments of the hydrostatic pressure and the temperature, δp and δT ,

are computed and transformed into virtual potential temperature and density increments following the linear mapping

$$\begin{aligned}\delta\rho &= X_\rho\delta p + Y_\rho\delta T + Z_\rho\delta q_v \\ \delta\theta_v &= X_{\theta_v}\delta p + Y_{\theta_v}\delta T + Z_{\theta_v}\delta q_v,\end{aligned}$$

which is motivated by the total differential of the thermodynamic state equations. The factors X , Y and Z are determined by the state before the nudging (ψ^*). Note again that water vapor increments are nonzero only in the lateral boundary nudging zone.

The nudging in ICON should not be confused with a Newtonian relaxation approach in the proper sense

$$\psi(t) = \psi^*(t) + \frac{\delta\psi}{\tau_{\text{relax}}}\Delta t. \quad (6.3)$$

If we identify Eq. (6.1) with Eq. (6.3), we find for the relaxation time

$$\tau_{\text{relax}} = \frac{\Delta t}{\alpha_{\text{nudge}}}. \quad (6.4)$$

It is proportional to the time step Δt (\rightarrow `dtime`) and, hence, inversely proportional to the horizontal mesh size, i.e. τ_{relax} decreases with increasing horizontal resolution. The nudging coefficient α_{nudge} is a non-dimensional parameter and should not be considered as an inverse relaxation time. Given a proper relaxation time $\tau_{\text{relax}} = \text{const.}$, Eq. (6.3) converges to a differential equation in the limit $\Delta t \rightarrow 0$. Equation (6.1) would, however, diverge in this limit, so no corresponding differential equation exists. This fact should not be considered a problem, as the nudging describes no physical process anyway. If desired, you may of course emulate a relaxation approach. Choose the desired relaxation time and compute the corresponding nudging coefficients A_0 or B_0 according to Eqs. (6.4) and (6.2). You may want to implement this as a function in your runscript, in order to avoid (easily forgotten) recomputations by hand whenever you change the time step `dtime`.

For the sake of completeness, we mention the global nudging option `nudge_type=2`. In fact, it is intended for global simulations (`1_limited_area=FALSE.`), but it may be used in the limited area mode as well. For the most part, it makes use of the same infrastructure as the upper boundary nudging. Taking a look at Figure 6.1, global nudging under the limited area mode means, in practical terms, the absence of the lateral and overlap nudging zones 1 and 3, and of the free model atmosphere zone 4. Instead, region 2, now termed *global nudging zone*, covers the entire domain interior. In addition to the vertical nudging shape factor $[(z - z_{\text{start}})/(z_{\text{top}} - z_{\text{start}})]^2$ (see Eq. (6.2), region 2), global nudging offers a small selection of shape factors that enable a more uniform nudging throughout the vertical air column of the model². Please note that this nudging option is not in operational use, still mostly experimental and comes with no warranty. As its control parameters in `nudging_nml` are described in detail in the namelist documentation `icon/doc/Namelist_overview/Namelist_overview.pdf`, they will not be discussed further here. Just a few remarks: Upper boundary and global nudging share most namelist parameters. Where global nudging assumes different default values, this is marked by

²You can find schematic representations of the available vertical nudging shape factors at the end of the source code file `icon/src/configure_model/mo_nudging_config.f90`.

(`)_glbndg`. Where parameters apply to global nudging only, this is indicated in the *Description* and *Scope* columns. In contrast to upper boundary nudging, global nudging applies to the primary domain only. Within nested domains, if present, there is no direct forcing by global nudging.



Important notes on upper boundary and global nudging: The treatment of the vertical velocity component w along the model top is independent of the upper boundary nudging: w and corresponding vertical mass fluxes are set to zero at the uppermost half level of the computational domain. Starting from the height specified by `damp_height` (`nonhydrostatic_nml`), the vertical velocity is damped towards zero following the method proposed by [Klemp et al. \(2008\)](#).

If upper boundary nudging is switched on (`nudging_nml: nudge_type=1`), “lateral boundary” data (the driving data for the nudging) have to be provided for the entire limited area domain rather than the lateral boundary region, only. This mode requires the namelist parameter `latbc_boundary_grid` (`limarea_nml`) defining the grid file on which the lateral boundary data are defined to be empty, i.e. `latbc_boundary_grid=" "`. The same applies to global nudging (`nudging_nml: nudge_type=2`). Driving data have to be present for the entire primary domain (either a limited area or a global domain).

As with upper boundary nudging, details on the driving data for global nudging must be specified in namelist `limarea_nml`.

Upper boundary nudging is **not** restricted to the primary limited area domain. In multi-domain simulations, it is possible to switch on upper boundary nudging for nested domains, by setting the corresponding entries in `nudge_type` (comma separated list) to 1. All domains are nudged towards the same driving data, i.e. nested domains are equally nudged towards the “lateral boundary” data of the primary limited area domain.

Running the model in regional mode is quite often accompanied by choosing a lower model top height compared to global simulations. In these cases, the neglected air mass above model top can have a noticeable impact regarding the attenuation of the incoming solar irradiance and can be the source of a small but noticeable amount of downward long-wave irradiance. To account for that in a rather ad hoc manner, an additional model layer above model top can be added by setting `latm_above_top = .TRUE.` (namelist `nwp_phy_nml`). It is used by the radiation scheme, only. The additional layer has a (hard-coded) thickness of 1.5 times the thickness of the uppermost model layer. Currently, temperature is linearly extrapolated, assuming a vertical gradient of -5 K km^{-1} . For ozone, aerosols and cloud fields, a simple no-gradient condition is assumed. Despite this rather ad hoc solution, it is suggested to activate the additional layer. Please note that this option works only in combination with a reduced radiation grid.

6.3. Model Initialization

The necessary input data to perform a limited area run are basically identical to those required for a global run (i.e. horizontal grid(s), initial conditions, external parameter; see Section 5.1), with the exception that lateral boundary data are required in addition in order to drive the model.

Technically it is possible to combine initial- and boundary data from different sources (e.g. one might take boundary data from IFS and initial data from ICON). In general, however, it is better to use boundary and initial data from the same source.

Dependent on the available initial data, the following initialization modes can be used in limited area mode:

init_mode (namelist **initicon_nml**, integer value)

init_mode = 2 initialize from **IFS data**.

This mode has already been described in Section 5.1.5 in the context of reading IFS analysis data.

init_mode = 3 initialize from **IFS atmospheric** and **ICON surface data**.

This mode is of special interest for operational weather services who want to perform cold starts with IFS atmospheric data.

init_mode = 7 initialize from **ICON data**.

This mode has already been described in Section 5.1.4 in the context of reading in DWD's initialized analysis product.

These modes have in common that the read-in process is followed by a vertical interpolation of the input fields to the target vertical grid. Thus the target vertical grid can be chosen independent of the vertical grid on which the input is defined. Note that in case of **init_mode = 7** the vertical interpolation requires that the field **HHL** (vertical half level heights) is contained in the initial data.

Specifics of **init_mode = 2**

- Only input data in NetCDF format are supported.
- A single input file per domain is expected, containing the analysis (or, more generally, the initial state). The filename must be provided for **ifs2icon_filename** (see also Section 5.1.5).
- The required input fields are listed in Table 2.2. As an alternative to the horizontal velocity components (**U**, **V**), the normal velocity component **VN** may be provided for read-in as well.

Specifics of `init_mode = 7`

- A single input file per domain is expected, containing the analysis (or, more generally, the initial state). The filename must be specified with the parameter `dwdfg_filename(initicon_nml)`.
- As we do not make use of a second input file containing explicit analysis information, it is good practice to indicate this via the following namelist parameter.

`lread_ana = .FALSE.` (namelist `initicon_nml`, logical value)

By default, this namelist parameter is set to `.TRUE.`. If `.FALSE.`, a separate analysis file is not required.

Note that in the recent ICON version `lread_ana=.FALSE.` is set automatically for `init_mode = 7`, if it has been forgotten by the user.

- The required input fields are listed in Table 11.3. A valid option is to use DWD's initialized analysis product for initialization. See Section 2.2.1 for ways to obtain it.

Specifics of `init_mode = 3`

- Two files are needed, `dwdfg_filename` containing the surface fields and `ifs2icon_filename` including the atmospheric fields (both in `initicon_nml`).
- For the atmospheric data the same procedures are used internally as for `init_mode = 2`. As a consequence, the same set of atmospheric variables is required (see Table 2.2) and the data needs to be in NetCDF format.
- Internally, the same procedures are used for the surface data as for `init_mode = 7`. Hence, the same surface fields are required (see Table 11.3).
- Initially, this `init_mode` was designed to combine IFS atmospheric data with GME surface data. As there is no fundamental difference between the ICON and GME surface parameterizations, this `init_mode` can be used for both.

6.4. Reading Lateral Boundary Data

The read-in of lateral boundary data is fortunately less cumbersome than the read-in of initial data, as it is based on a decision tree. The user is no longer required to select a specific mode which (hopefully) fits the data at hand. Instead, ICON scans the boundary data file and, dependent on its content, ICON diagnoses additional fields so as to obtain the internally required set of variables. The decision tree is depicted in Figure 6.2. If the provided data set does not match any of the trees, an error is thrown. As a result, ICON can handle variable sets from hydrostatic models (e.g. IFS) as well as non-hydrostatic models (e.g. COSMO, ICON) without the assistance of the user.

As apparent from the decision tree, three different variable sets can be handled. See Figure 2.11 for its specific content.

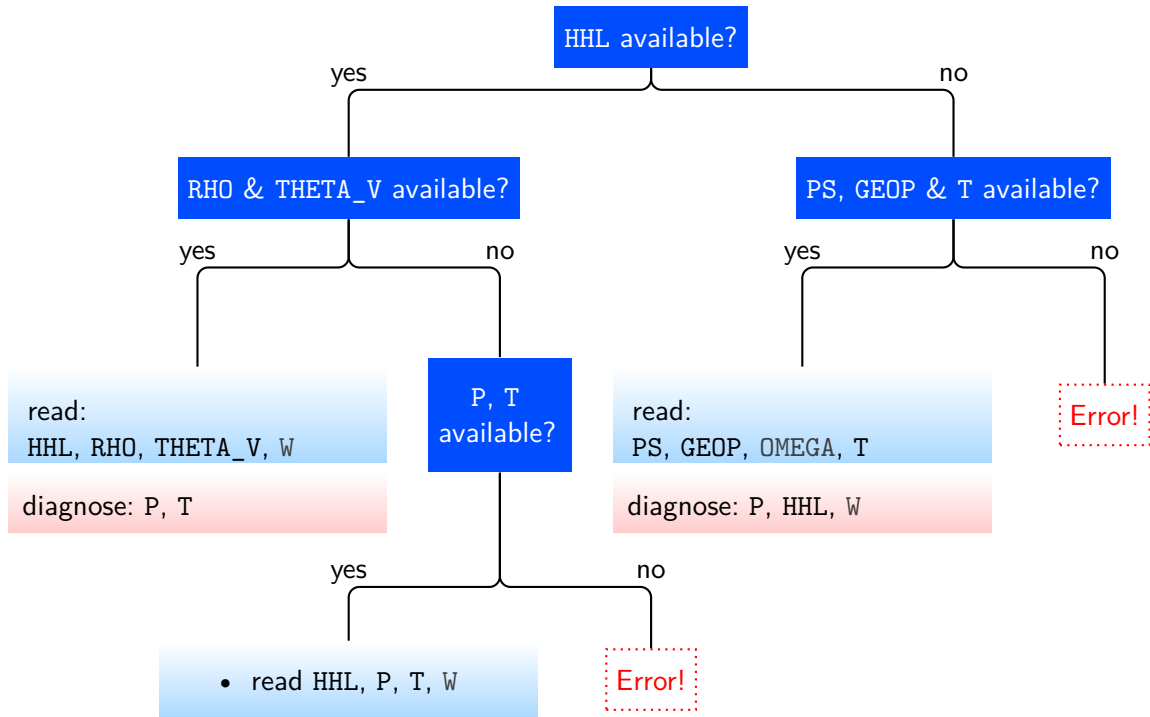


Figure 6.2.: Read-in of lateral boundary data. Based on this decision tree, ICON investigates the data file contents and diagnoses additional fields. The following fields are read additionally from file: velocity fields VN (or U, V) and mixing ratios QV, QC, QI (optional: QR, QS). The fields W and OMEGA are optional; if they are unavailable, the vertical wind is initialized with zero. For the input from a pressure based coordinate system (right branch), note that ICON expects the field OMEGA under the name W.



Important note: Boundary data sets originating from a non-hydrostatic model with height based vertical coordinates (e.g. COSMO or ICON) must contain the field HHL (vertical half level heights). It is required by the vertical interpolation procedure. Note, however, that the field only needs to be present in the boundary data set whose validity date equals the model start date.



Troubleshooting: The usage of a decision tree as depicted in Figure 6.2 has consequences when searching for the cause of an error. For example, a wrong specification of HHL in `latbc_varnames_map_file` leads to the following error behavior: ICON does not find HHL, so it erroneously takes the right branch of the decision tree. ICON then looks for the geopotential GEOSP (or alternatively GEOP_ML). This is also not found which results in the error that the geopotential was not found.

Read-in of boundary data is controlled by the following namelist parameters:

The type of lateral boundary conditions is specified by

itype_latbc (namelist **limarea_nml**, Integer value)

If set to 1 time-dependent boundary conditions are used. ICON then tries to read external data files at regular time intervals from a particular location specified by **latbc_filename** and **latbc_path** (see below).

If set to 0, time-constant lateral boundary conditions are used which are derived from the initial conditions.

Boundary data is read at regular time intervals. This is specified by the following namelist parameter:

dttime_latbc (namelist **limarea_nml**, floating-point value)

Time difference in seconds between two consecutive boundary data sets. At intermediate times, boundary conditions are computed by linear interpolation in time.

6.4.1. Naming Scheme for Lateral Boundary Data

Naturally, the sequence of lateral boundary data files must satisfy a consistent naming scheme. It is a good idea to consider this convention already during the pre-processing steps (see Section 2.3).

Filenames: **latbc_filename**, **latbc_path** (string parameters, **limarea_nml**)

By default, the filenames are expected to have the following form:

```
"prepiconR<nroot>B<jlev>_<y><m><d><h>.nc"
```

Here, several keywords are used which are further explained below. This naming scheme can be flexibly altered via the namelist parameter **latbc_filename** (namelist **limarea_nml**) using the available keywords. The absolute path to the boundary data can be specified with **latbc_path** (string parameter, **limarea_nml**).



By changing the above setting, the user has full flexibility with respect to the filename structure. The following keywords are allowed for the namelist parameter **latbc_filename**:

<nroot>	grid root division Rx (single digit)
<nroot0>	grid root division Rxx (two digits)
<jlev>	grid bisection level Byy (two digits)
<dom>	domain number (two digits)
<y>	year (four digits)
<m>	month (two digits)
<d>	day in month (two digits)
<h>	hour (UTC) (two digits)
<min>	minutes (UTC) (two digits)
<sec>	seconds (UTC) (two digits)
<ddhhmmss>	elapsed days, hours, minutes and seconds since ini_datetime_string or experimentStartDate (each two digits)
<dddhh>	elapsed days and hours since ini_datetime_string or experimentStartDate (three digits day, two digits hours).

Field names: `latbc_varnames_map_file` (namelist `limarea_nml`, string)

ICON supports NetCDF and GRIB2 as input format for boundary fields. Field names in input files do not necessarily coincide with internal ICON field names. Hence, an additional input text file (*dictionary file*) can be provided. This two-column file translates between the ICON variable names and the corresponding DWD GRIB2 short names or NetCDF variable names.

Specifying a valid dictionary file is currently mandatory, if pre-fetching of boundary data is selected `num_prefetch_proc=1` (see below).

Boundary grid: `latbc_boundary_grid` (namelist `limarea_nml`, string)

As it has been explained in Section 2.3, the lateral boundary data can be defined on an auxiliary grid, which contains only the cells of the boundary zone for optimization purposes.

If this is the case for the applied boundary data, the filename of this grid file must be specified with this namelist parameter.

6.4.2. Pre-Fetching of Boundary Data (Mandatory)

Pre-fetching strives to avoid blocking of the computation due to reading of boundary data. The term denotes the reading of files ahead of time, i.e. the next input file will be processed simultaneously with the preceding compute steps. This avoids waiting for the I/O processes during the time consuming procedure of opening, reading and closing of the input files.

`num_prefetch_proc = 1` (namelist `parallel_nml`, integer value)

If this namelist option is set to 1, one MPI process will run exclusively for asynchronously reading boundary data during the limited area run. This setting, i.e. the number of pre-fetching processors, can be zero or one.

Enabling the pre-fetching mode is **mandatory** for the described LAM setup.

6.5. Tropical Setup

The tropical setup namelist configuration of the COSMO model was and still is applied by many users of the COSMO model. As ICON is a global model, it is routinely applied to a wider range of conditions than a limited area only model like the COSMO model. Nevertheless, when choosing an area close to the equator some namelist parameters have to be adapted compared to mid-latitude high-resolution limited area setups. For some of these parameters, the necessary adaptations are quite straight forward. Other parameters require rigorous testing and tuning of the configuration in order to find a good combination for these parameters.

Table 6.1 provides an overview on namelist parameters that are sensitive to the choice of the domain location. For the terminal fall velocity of ice (`tune_zvz0i`) literature suggests higher values than used in current ICON setups (e.g., Heymsfield and Donner (1990) suggest a value which is higher by a factor of 3). Such literature values are derived under

Parameter	Mid-Latitudes	Tropical Setup	Description
<code>tune_zvz0i</code> (<code>nwp_tuning_nml</code>)	0.85	test & tune	Terminal fall velocity of ice, meaningful range for tuning between 1. and 3.5
<code>rat_sea</code> (<code>turbdiff_nml</code>)	0.8	test & tune	Ratio of laminar scaling factors over sea and land, meaningful range for tuning between 0.8 and 20.
<code>inwp_convection</code> <code>lshallowconv_only</code> (<code>nwp_phy_nml</code>)	1 .false.	test & tune	Convection parameterization, <code>inwp_convection=1</code> and <code>lshallowconv_only=.true.</code> for shallow convection only, <code>inwp_convection=0</code> for no convection parameterization
<code>top_height</code> (<code>sleve_nml</code>)	22000.	30000.	Model top height (only for <code>ivctype=2</code> , see Section 3.4)
<code>damp_height</code> (<code>nonhydrostatic_nml</code>)	12250.	18000.	Height at which Rayleigh damping of vertical wind starts

Table 6.1.: List of namelist parameters that are sensitive to the choice of the limited area domain location. Values that are listed in the table are suitable for setups around 2.5 km effective grid spacing.

certain conditions (e.g., particle shape, temperature range). Hence, the value which is required by a model in order to get good results can differ. This value has a strong impact on the radiative properties of ice clouds and can be used to compensate biases. In summary, `tune_zvz0i` is a tuning parameter worth investigating if there are biases in radiation under cloudy conditions.

The namelist parameter `rat_sea` increases the thickness of the laminar sublayer over sea. Larger values mean larger laminar sublayers, i.e. less heat and moisture fluxes over sea. For example, tuning this parameter to lower values might be beneficial if there is too little ocean-atmosphere exchange in case of tropical cyclones. This parameter can only have an impact if a significant part of the model is covered by ocean.

At effective grid spacings of 5 km or less, deep convection is resolved explicitly. The namelist switch `lshallowconv_only` allows to turn off the deep convection parameterization but keeping the shallow convection parameterization active. However, especially for arid regions at effective grid spacings of <3 km it can be beneficial to turn off convection completely by setting `inwp_convection=0`. Otherwise the already sparsely available water vapor is lifted by the shallow convection parameterization from the boundary layer into the free atmosphere. This can lead to a too strong cloud formation inhibition in arid regions.

The probably most important change for tropical setups is depicted schematically in Figure 6.3. The model top height (`top_height`) and the height above which the Rayleigh

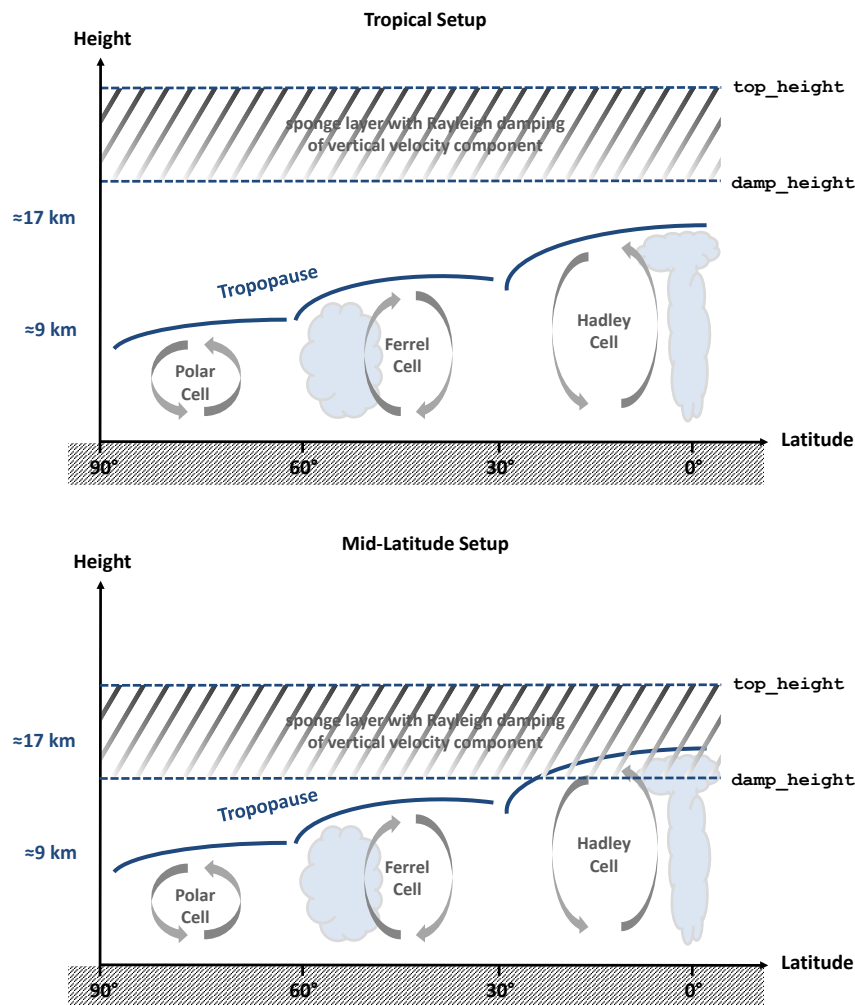


Figure 6.3.: Schematic illustration of the choice of the namelist parameters `damp_height` and `top_height` for tropical and mid-latitude setups.

damping of the vertical velocity becomes active (`damp_height`) have to be chosen such that convection is not inhibited by the Rayleigh damping. The tropical tropopause reaches to higher altitudes than the mid-latitude and polar tropopause. While it is sufficient for mid-latitude setups to choose the model top at 22km and start the damping layer at about 12km, the tropical tropopause is typically located at an altitude of 17km. Hence, the relaxation of the vertical velocity would in this case inhibit deep convection. It can be easily avoided by extending the vertical extent of the model simulation to, for example, `top_height=30000.0` and `damp_height=18000.0`. This requires a sufficient number of vertical levels (e.g., `num_lev=65`).



Tuning documentation: Recent ICON model versions contain a document named *ICON model parameters suitable for model tuning* which is located in `doc/tuning/icon_tuning_vars.pdf`.

7. Model Output

In this chapter we describe advanced settings for the namelist controlled model output. In particular, the ICON model offers several options for internal post-processing, such as the horizontal remapping of the prognostic output onto regularly spaced (“longitude-latitude”) grids and vertical interpolation, for example on pressure levels. Another type of output products is ICON’s checkpointing feature which allows to restart the execution from a pre-defined point using the data stored in a file.

7.1. Settings for the Model Output

Model output is enabled via the namelist `run_nml` with the main switch `output`. By setting this string parameter to the value `"nml"`, the output files and the fields requested for output can be specified by special namelists¹. In the following, this procedure will be described in more detail.

In general the user has to specify five individual quantities to generate output of the model. These are:

- a) The time interval between two model outputs.
- b) The name of the output file.
- c) The name(s) of the variable(s) to output.
- d) The type of the vertical output grid, e. g., pressure levels or model levels.
- e) The type of the horizontal output grid, i. e. ICON grid or geographical coordinates.

All of these parameters are set in the namelist `output_nml`. Multiple instances of this namelist may be specified for a single model run, where each `output_nml` creates a separate output file. The options d) and e) require an interpolation step. They will be discussed in more detail in Section 7.1.1.

In the following, we give a short explanation for the most important namelist parameters:

output_filename (namelist output_nml, string parameter)

This namelist parameter defines a prefix for the output filename (which may include the directory path). The domain number, level type, file number and file format extension will be appended to this prefix.

¹Another possibility is to set `output="none"`. This can be used to make scalability tests without being influenced by writing time.

output_bounds (namelist output_nml, three floating-point values)

This namelist parameter defines the start time and the end time for the model output and the interval between two consecutive write events. The three values for this parameter are separated by commas and, by default, they are specified in seconds.

ml_varlist (namelist output_nml, character string list)

This parameter is a comma-separated list of variables or variable groups (the latter are denoted by the prefix “group:”). The `ml_varlist` corresponds to model levels, but all 2D variables, for example surface variables, are specified in the `ml_varlist` as well. It is important to note that the variable names follow an ICON-internal nomenclature. The temperature field, for example, is denoted by the character string “temp”. A list of available output fields is provided in Appendix A.

Users can also specify the variable names in a custom naming scheme, for example “T” instead of “temp”. To this end, a translation table (a two-column ASCII file) can be provided via the parameter `output_nml_dict` in the namelist `io_nml`. An example for such a dictionary file can be found in the source code directory: `run/dict.output.dwd`.

m_levels (namelist output_nml, character string)

This character string specifies a list of model levels for which the variables and groups should be written to output. Level ordering does not matter.

Allowed is a comma- (or semicolon-) separated list of integers, and of integer ranges like “10...20”. One may also use the keyword “nlev” to denote the maximum integer (or, equivalently, “n” or “N”). Furthermore, arithmetic expressions like “(nlev-2)” are possible.

Basic example: `m_levels = "1,3,5...10,20...(nlev-2)"`

dom (namelist output_nml, integer values, comma-sep.)

Related to setups with nests, i.e. multiple domains: This namelist parameter sets the domains for which this namelist is used. If not specified (or specified as -1), this namelist will be used for all domains.

remap (namelist output_nml, integer value: 0/1)

This namelist parameter is related to the horizontal interpolation of the output to regular grids, see Sections 7.1.1 and 7.1.2.

filetype (namelist output_nml, integer value: 2/4)

ICON offers the possibility to produce output either in NetCDF or GRIB2 format. This can be chosen by the namelist parameter `filetype` of the namelist `output_nml`. Here, the value `filetype=2` denotes the GRIB2 output, while the value `filetype=4` denotes the NetCDF file format.



The namelist parameter `output_filename` provides only partial control over the resulting filename, namely its prefix. Complete control over the resulting filename can be achieved with the namelist parameter `filename_format` (namelist `output_nml`, character string). By default, `filename_format` is set to:

```
"<output_filename>_DOM<physdom>_<levtype>_<jfile>"
```

The following keywords are allowed for `filename_format` (the list is incomplete and shows only the most important options):

<code><path></code>	model base directory (namelist parameter <code>model_base_dir</code> , namelist <code>master_nml</code>)
<code><physdom></code>	domain number (two digits)
<code><levtype></code>	Level type (ML, PL, HL, IL)
<code><datetime></code>	ISO-8601 date-time stamp in format YYYY-MM-DDThh:mm:ss.sssZ
<code><ddhhmmss></code>	elapsed days, hours, minutes and seconds since <code>ini_datetime_string</code> or <code>experimentStartDate</code> (each two digits)
<code><datetime2></code>	ISO-8601 date-time stamp in format YYYYMMDDThhmmssZ
<code><jfile></code>	Consecutive file id.

As it has been stated before, each `output_nml` creates a separate output file. To be more precise, there are a couple of exceptions to this rule. First, multiple time steps can be stored in a single output file, but they may also be split up over a sequence of files (with a corresponding index in the filename), see the namelist parameter `steps_per_file`. Second, an instance of `output_nml` may also create more than one output file if grid nests have been enabled in the model run together with the global model grid, see the namelist parameter `dom`. In this case, each of the specified model domains is written to a separate output file. Finally, model output is often written on different vertical axes, e. g., on model levels and on pressure levels. The specification of this output then differs only in the settings for the vertical interpolation. Therefore it is often convenient to specify the vertical interpolation in the same `output_nml` as the model level output, which again leads to multiple output files.

7.1.1. Output on Regular Grids and Vertical Interpolation

Many diagnostic tools, e. g., to create contour maps and surface plots, require a regularly spaced distribution of the data points. Therefore, the ICON model has a built-in output module for the interpolation of model data from the triangular mesh onto a regular longitude-latitude grid. Further information on the interpolation methods can be found in the database documentation [Reinert et al. \(2024\)](#), see Section 0.2.

The relevant namelist parameters for the *horizontal* interpolation of the output fields are set in the namelist `output_nml`. As it was already described in Section 7, multiple instances of this namelist may be specified for a single model run, where each `output_nml` creates a separate output file.

remap (namelist `output_nml`, integer value 0=triangular / 1=lon-lat)

Set this namelist parameter to the value 1 to enable horizontal interpolation onto a regular grid. This option needs to be defined combination with `reg_lat_def` / `reg_lon_def`.

reg_lat_def / **reg_lon_def** (namelist `output_nml`)

Latitudes and longitudes for the regular grid points are each specified by three values:

start, increment, end value; given in degrees. Alternatively, the user may set the number of grid points instead of an increment.

Furthermore, the model output can be written on a different *vertical* axis, e. g., on pressure levels, height levels or isentropes. In the following we will describe how to specify these options in the namelist `output_nml`. The relevant namelist parameters for the vertical interpolation of the output fields are:

hl_varlist / pl_varlist / il_varlist (character string lists)

Similar to the namelist parameter `ml_varlist` mentioned above, these parameters are comma-separated lists of variables or variable groups. While the `hl_varlist` sets the output for height levels, `pl_varlist` defines variables on pressure levels and `il_varlist` specifies output on isentropic levels.

h_levels / p_levels / i_levels (floating point values, comma-sep.)

Comma separated list of height, pressure, and isentropic levels for which the variables and groups specified in the above mentioned variable lists should be output. Height levels must be given in meters (m), pressure levels in Pascal (Pa) and isentropes in Kelvin (K). Level ordering does not matter.



ICON's interpolation on pressure levels is extrapolating into the topography. This has the simple reason that contour plots, e.g. for 850hPa, usually do not show missing values over regions like Antarctica. There is no option to change this behavior in ICON. If needed, this has to be accounted for in the post-processing.

7.1.2. Remarks on the Horizontal Interpolation

ICON supports several numerical methods for interpolating data horizontally from the native triangular grid onto a regular latitude-longitude grid:

- Radial basis functions (RBF)
- Barycentric interpolation
- Nearest-neighbor interpolation

This section provides some explanation on these algorithms. They also apply to the `iconremap` tool, which is a way to interpolate ICON data even between different triangle grids.

The concrete interpolation procedure depends on the variable and its physical characteristics. It is prescribed for the output module as indicated in the output product tables of ICON's database description, see [Reinert et al. \(2024\)](#). For the `iconremap` tool, the interpolation method is explicitly set by the user for each field.

First, a small number of output fields is treated with a *nearest-neighbor interpolation*. The nearest neighbor algorithm selects the value of the nearest point and does not consider the values of neighboring points at all, yielding a piecewise-constant interpolant.

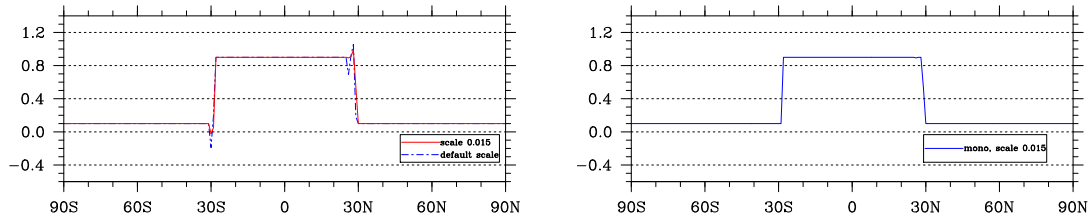


Figure 7.1.: *Left:* Examples for over- and undershoots for an RBF-based interpolation. *Right:* RBF interpolant with cut-off applied.

Barycentric interpolation is a two-dimensional generalization of linear interpolation. This method uses just three near-neighbors to interpolate and avoids over- and undershoots, since extremal values are taken only in the data points. For a detailed treatment we refer to the literature, for example to [Press et al. \(2007\)](#), Section 21.7.1: "Two-Dimensional Interpolation on an Irregular Grid". This interpolation makes sense for fields where the values change in a roughly piecewise linear way.

Barycentric interpolation needs to be enabled with the following namelist setting (otherwise it is replaced by a fallback interpolation):

```
support_baryctr_intp = .FALSE. (namelist interpol_nml, logical value)
```



*The **icondelaunay** tool:* The DWD ICON Tools contain the **icondelaunay** binary, which processes existing ICON grid files. It appends a Delaunay triangulation of the cell circumcenters to the grid file. This auxiliary triangulation can be used then to speed up the interpolation process.

Note that this way of pre-processing the ICON grid files is mandatory for DWD's NEC SX-Aurora platform where the spherical Delaunay algorithm has not been vectorized.

Most of the output data on regular grids is processed using an *RBF-based interpolation method*. The algorithm approximates the input field with a linear combination of radial basis functions (RBF) located at the data sites, see, for example, [Ruppert \(2007\)](#). RBF interpolation typically produces over- and undershoots at position where the input field exhibits steep gradients. This behavior is illustrated in Fig. 7.1. Therefore, the internal interpolation algorithm performs a cut-off by default. Note that RBF-based interpolation is not conservative.

The *shape parameter* (or *scale parameter*) is an important parameter which affects the quality of the RBF interpolation. The core of the interpolation method are the *radial basis functions* which are for ICON chosen to be of Gaussian type, i. e.

$$f(x) := e^{-\left(\frac{x}{a}\right)^2}, \quad a > 0,$$

with shape parameter a .

When we choose a smaller value for a then the RBF basis functions approach the Dirac delta function, which yields an almost-nearest-neighbor interpolation. Larger values for a generally reduce the interpolation error, but there exists a (grid specific) bound where the Cholesky decomposition of certain dense matrices fails, that are necessary for the RBF weight computation (see the info box below). The ICON model provides a heuristic which estimates a proper RBF shape parameter for which the Cholesky decomposition succeeds in floating-point arithmetic. This estimation method is applied when the user does not provide a specific value via the namelist. For the latter case, see ICON's namelist documentation for the namelist parameter `rbf_scale` (`output_nml`).

The estimated value is reported in the log output of the ICON model:

```
mo_intp_lonlat::rbf_setup_interpol_lonlat_grid:
  auto-estimated shape_param = 1.1730307423896675E-002
```



Model abort due to failed Cholesky decomposition: The Cholesky decomposition of the RBF interpolation weight computation may fail with an error message of the following kind:

```
# mo_math_utilities:choldec: error in matrix inversion, nearly singular matrix
mo_remap_rbf_errana::rbf_error: Cholesky decomposition failed!
```

This may happen for example when a bad value for the shape parameter has been chosen manually. However, the *automatic* shape parameter estimation may fail as well: This algorithm estimates largest-as-possible shape parameter by extrapolation from a number of sample (test) decompositions. When it fails to compute these samples, even the automatic estimator may abort with the above error message.

In these cases, please adjust the shape parameter manually (which may require several trial-and-error steps).

7.1.3. Interpolation onto Rotated Lat-Lon Grids

Users of the COSMO model are familiar with *rotated lat-lon grids*: Here, the computational spherical coordinate system is rotated in such a way that a pole problem is avoided and minimal convergence of the meridians is achieved. To some extent, this output can be reproduced by the ICON model:

north_pole (namelist `output_nml`)

Definition of the north pole for rotated lat-lon grids (`[longitude, latitude]`). The default is `north_pole = 0,90`.

Note, however, that the “COSMO output” is in detail not quite what the COSMO user expects, especially with regard to wind speeds U, V : First, the basis vectors, i.e. the meridional and longitudinal directions, are not rotated. Second, in COSMO these speeds are defined on horizontally and vertically shifted grids (“staggered grids”). This is not the case with ICON and especially difficult to detect in data sets.

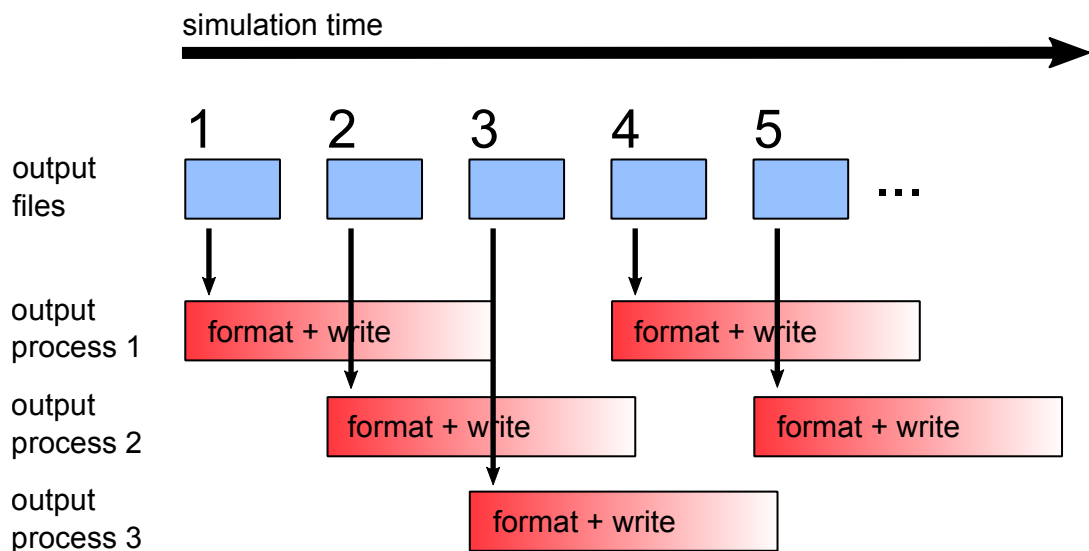


Figure 7.2.: Schematic illustrating the distribution of output load onto three output processes, using `num_io_procs=3` and `stream_partitions_ml=3`.

Regular grids over the poles. If one wishes to obtain “circular” lat-lon grids near the poles, that is, e.g., grids that cover everything north of a chosen latitude circle, then the unrotated definition of the area in the `output_nml` can be used, provided it includes the pole.

Namelist example for ICON’s `output_nml`:

```
reg_lat_def = 40,0.5,90
reg_lon_def = 0,0.5,360
```

For rotated lat-lon grids – which then turn out rectangular in the satellite projection – one additionally uses the namelist parameter `north_pole` (`output_nml`), which defines the coordinate north pole of a rotated (λ, φ) system (longitude, latitude). The rotation operation is quite well described in the documentation of the COSMO model from which it was adopted for ICON, see Section 2.3 in [Doms and Baldauf \(2018\)](#); Section 4.1 and Appendix A in [Doms et al. \(2003\)](#).

Namelist example:

```
reg_lon_def      = -10.,0.5,10.0
reg_lat_def      = -15.,0.5,15.0
north_pole       = 0.,0.
```

7.1.4. Output Rank Assignment

When a large number of different output files is written during the simulation, the task of formatting and writing may put an excessive load on the output processes. The number of output processes which share this output load can be increased by setting the

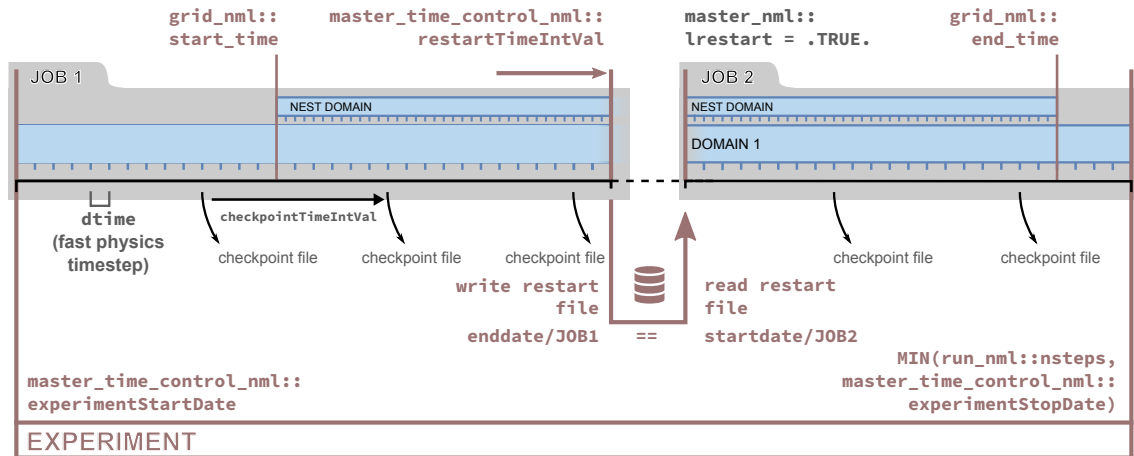


Figure 7.3.: Specifying experiment restart; compare this illustration to Fig. 5.1. The namelist parameters are explained in Section 7.2. Here we prefer the ISO8601 date-time specification (e.g. `checkpointTimeIntVal`) over the older settings (e.g. `dt_checkpoint`).

`num_io_procs` (`parallel_nml`) namelist parameter, see Section 8.2. If there exist multiple groups (e.g. output files, different variable sets, output intervals, interpolation grids) then these so-called *streams* will be distributed automatically over the available output processes.

`stream_partitions_m1` (namelist `output_nml`, integer)

It may be even useful to spread the files of a *single* stream over multiple output processes. For example, when each output file is relatively large, then the subsequent file of this stream can be written by a different output process in order to diminish the risk of congestion. Please use the namelist parameter `stream_partitions_m1` to set the number of output processes among which the output files should be divided.

The distribution of output load using `stream_partitions_m1` is illustrated in Fig. 7.2.

`pe_placement_m1` (namelist `output_nml`, integer array)

This array is related to the namelist parameters `num_io_procs` and `stream_partitions_m1` and allows for an even more fine-tuned distribution of the output workload. At most `stream_partitions_m1` different ranks can be specified, ranging between 0 ... (`num_io_procs` - 1). This explicitly assigns the output streams to specific PEs and facilitates a load balancing with respect to small and large output files.

7.2. Checkpointing and Restart

There are many reasons why a simulation execution may be interrupted prematurely or unexpectedly. The checkpoint/restart option can save you from having to start the ICON

model over from the beginning if it does not finish as expected. It allows you to restart the execution from a pre-defined point using the data stored in a checkpoint file.

Activating the restart. The checkpoint/restart functionality is controlled by the following namelist parameters, which are also illustrated in Fig. 7.3.

dt_checkpoint (namelist io_nml, floating-point value)

This parameter specifies the time interval for *writing* restart files. The restart files are written in NetCDF format, and their names are specified by the namelist parameter `restart_filename`, see below.

Note that if the value of `dt_checkpoint` resulting from the model default or user's specification is larger than `dt_restart` (see below), then it will be automatically reset to `dt_restart`, s.t. at least one restart file is generated during the restart cycle.

Similar to the namelist parameters described in Section 5.1.1, which specify the model start and end dates, there exist character string replacements for `dt_checkpoint` and `dt_restart`:

- `restartTimeIntVal` (namelist `master_time_control_nml`, ISO8601, character string)
- `checkpointTimeIntVal` (namelist `master_time_control_nml`, ISO8601, character string)

lrestart (namelist master_nml, logical value)

If this namelist parameter is set to `.TRUE.` then the current experiment is resumed from a restart file.

Instead of searching for a specific data filename, the model reads its restart data always from a file with name `restart_atm_DOM01.nc` (analogously for nested domains). It is implicitly assumed that this file contains the newest restart data, because during the writing of the checkpoints this file is automatically created as a symbolic link to the latest checkpoint file.

restart_filename (namelist run_nml, string parameter)

This namelist parameter defines the name(s) of the checkpointing file(s). By default, the checkpoint files (not the symbolic link) have the form

`gridfile_restart_atm_restarttime.nc`

dt_restart (namelist time_nml, floating-point value)

This parameter is in some ways related to the `dt_checkpoint` parameter: It specifies the length of a restart cycle in seconds, i.e. it specifies how long the model runs until it saves its state to a file *and stops*. Later, the model run can be resumed, s.t. a simulation over a long period of time can be split into a chain of restarted model runs.

Similar to the asynchronous output module, the ICON model also offers the option to reserve a dedicated MPI task for writing checkpoint files. This feature can be enabled by setting the parameter `num_restart_procs` in the namelist `parallel_nml` to an integer value larger than 0.

Restart modes. Different restart write modes are available, which allow for a distributed writing and read-in of restart files, depending on the parallel setup. These different restart modes are controlled via the namelist parameter `restart_write_mode` (`io_nml`).

Allowed – character strings (!) – settings for `restart_write_mode` are:

"joint procs multifile"

All worker processes write restart files to a dedicated directory. Therefore, the directory itself represents the restart data set. The information is stored in a way that it can be read back into the model independent from the processor count and the domain decomposition.

Read-in: All worker processes read the data in parallel.

"dedicated procs multifile"

In this case, all the restart data is first transferred to memory buffers in dedicated restart writer processes. After that, the work processes carry on with their work immediately, while the restart writers perform the actual restart writing asynchronously. Restart processes can parallelize over patches and horizontal indices.

Read-in: All worker processes are available to read the data in parallel (though this is usually limited by the number of restart files).

"sync"

'Old' synchronous mode. Process # 0 reads and writes restart files. All other processes have to wait.

"async"

'Old' mode for asynchronous restart writing: Dedicated processes write restart files while the simulation continues (`num_restart_proc > 0`). Restart processes can only parallelize over different patches.

Read-in: Processes # 0 reads while other processes have to wait.

" "

Fallback mode.

If `num_restart_proc` (`parallel_nml`) is set to 0, then this behaves like "sync", otherwise like "async".

7.3. Meteogram Output

The ICON model also features a special output product called *meteograms*, containing the model variables with respect to time for a particular location, i.e. at single grid points.

ICON's built-in meteograms are intended for *non-operational use*. They should be seen as a by-product of the usual data output, rather for the purpose of error detection during development. Meteogram data files are written in the NetCDF data format, where an example of the (non-standard) file structure is given below. The output is enabled via the namelist setting `output = 'nml'` (namelist `io_nml`) in combination with a special namelist, `meteogram_output_nml`:

```

&meteogram_output_nml
  lmeteogram_enabled = .TRUE.
  n0_mtgrm           = initial time step for meteogram output
  ninc_mtgrm         = output interval (in time steps)
  stationlist_tot     = 50.050, 8.600, 'Frankfurt-Flughafen',
                        40.74153, -73.98537, 'New York City',
  ...

```



Meteogram functionality through ComIn.: It should be noted that the ICON community interface *ComIn* provides an alternative way to export meteorological variables with respect to time for a given location. For details on *ComIn*, which is part of the ICON model code, see Section 9.5.

Especially for plugins written in the Python programming language and attached to the running ICON simulation, graphical meteogram presentations can be generated by sampling field entries through the `comin` module.

Only a few lines of code are required to accomplish this task. Furthermore, third-party modules can be integrated to identify the ICON triangles for specific geographic locations (`science.spatial.kdtree`) and to plot the sample results (e.g. `matplotlib`).

Details on the namelist settings. In ICON's built-in meteogram mechanism, in addition to the namelist parameters in the example above, the following settings are worth mentioning:

`max_time_stamps` (namelist `meteogram_output_nml`, integer value)
 number of output time steps to record in memory before flushing to disk

`zprefix` (namelist `meteogram_output_nml`, character string)
 string with file name prefix for output file

`var_list` (namelist `meteogram_output_nml`, list of character strings)
 Optional positive-list of variables. Only variables contained in this list are included in the meteogram. If the default list is not changed by this user setting, then all available variables are added to the meteogram.

During the model simulation, one of the asynchronously running output processes (see Section 8) collects the meteogram buffers from the compute processes and writes the data to a file. Meteograms do not use ICON's variable list infrastructure (see Section 9.4). However, the output can be easily extended to sample of additional model variables. To this end, see the extensive comments in the source code, `src/io/atmo/mo_mtgrm_output.f90`.

For basic textual output, there exists an auxiliary NCL script

```
scripts/postprocessing/tools/mtgrm_cosmo.ncl
```

For an introduction to the NCAR Command Language NCL see Section 10.3.2. This script can be applied to a data file with the following command:

```
ncl -n mtgrm_cosmo.ncl DataFileName="'METEOGRAM.nc"' itime=0;
```

The same directory also contains scripts for plotting meteogram data with NCL.

File format description. As mentioned before, the NetCDF meteogram output has a non-standard file structure. We list the most important file entries in the following:

meteogram station info:

station_name, station_lon, station_lat, station_hsurf,
and station_idx, station_blk: global triangle adjacent to meteogram station

sample date info:

date (sample dates) and time_step (plain time step indices).

info and value buffer for surface (2D) variables, 1, ..., nsfcvars:

sfcvar_name, sfcvar_long_name, sfcvar_unit,
sfcvalues(time, nsfcvars, nstations): value buffer

info and value buffer for 3D variables 1, ..., nvars:

var_name, var_long_name, var_unit,
heights(max_nlevs, nvars, nstations): level heights,
var_levels: plain level indices,
values: value buffer

8. Parallelization and Performance Aspects

This chapter gives an overview of the different mechanisms for parallel execution of the ICON model. These settings become important for performance scalability when increasing the model resolution and core counts.

8.1. Modes of Parallel Execution

The ICON model supports different modes of parallel execution:

- In the first place, ICON has been implemented for distributed memory parallel computers using the *Message Passing Interface* (MPI). MPI is a library specification, proposed as a standard by a broadly based committee of vendors, implementors, and users, see <http://www.mcs.anl.gov/research/projects/mpi>. Multiple ICON processes (*processing elements*, PEs) are started simultaneously and communicate by passing messages over the network. Each process is assigned a part of the grid to process.
- Moreover, on multi-core platforms, the ICON model can run in parallel using *shared-memory parallelism with OpenMP*. The OpenMP API is a portable, scalable technique that gives shared-memory parallel programmers a simple and flexible interface for developing parallel applications on platforms ranging from embedded systems and accelerator devices to multi-core systems and shared-memory systems, see <http://openmp.org>. An implementation of OpenMP ships with your Fortran compiler. OpenMP-parallel execution therefore does not require the installation of additional libraries.

To be precise, “OpenMP” refers to shared-memory parallelization within the host processor. The OpenMP target offloading for accelerators is not supported in ICON. Instead, ICON uses OpenACC for accelerator support, see the following option.

- As a third option, ICON can make use of graphics processing units (GPUs) to accelerate calculations. Compared to standard x86 CPUs, GPUs typically have many times the number of cores but require a sophisticated host-device memory management when computational problems are offloaded. ICON-Atmosphere (ICON-A) has been gradually ported to GPUs using OpenACC directives, see Section 8.5.

These mechanisms are not mutually exclusive. A *hybrid* approach is also possible: Multiple ICON processes are started, each of which starts multiple threads. The processes communicate using MPI. The threads communicate using either OpenMP or OpenACC.

Finally, note that although ICON has been implemented for distributed memory parallel computers using the *Message Passing Interface* (MPI), the model can also be installed

on sequential computers, where MPI and/or OpenMP are not available. Of course, this execution mode limits the model to rather small problem sizes.

8.2. Settings for Parallel Execution

First, we focus on some namelist settings for the distributed-memory MPI run. Processors are divided into

<i>Worker PEs</i>	this is the majority of MPI tasks, doing the actual work
<i>I/O PEs</i>	dedicated output server tasks ¹
<i>Restart PEs</i>	for asynchronous restart writing (see Section 7.2)
<i>Prefetch PE</i>	for asynchronous read-in of boundary data in limited area mode (see Section 6.4.2)
<i>Test PE</i>	MPI task for verification of MPI parallelization (debug option)

Several settings must be adjusted to control the parallel execution:

Namelist `parallel_nml`

The configuration settings are defined in the namelist `parallel_nml`. To specify the number of output processes, set the namelist parameter `num_io_procs` to a value larger than 0, which reserves a number of processors for output. While writing, the remaining processors continuously carry out calculations. Conversely, setting this option to 0 forces the worker PEs to wait until output is finished. For the writing of the restart checkpoints (see Section 7.2), there exists a corresponding namelist parameter `num_restart_procs`.

During start-up, the model prints out a summary of the processor partitioning. This is often helpful to identify performance bottlenecks. First of all, the model log output contains a one-line status message:

Number of procs for

test: `xxx`, work: `xxx`, I/O: `xxx`, Restart: `xxx`, Prefetching: `xxx`

Afterwards, the sizes of grid partitions for each MPI process are summarized as follows:

Number of compute PEs used for this grid: `xxx`
 # prognostic cells: max/min/avg `xxx` `xxx` `xxx`

Given the case that the partitioning process would fail, these (and the subsequently printed) values would be grossly out of balance.

Apart from the namelist settings, the user has to specify the computational resources that are requested from the compute cluster. In addition to the number of MPI tasks and OpenMP threads, here the user has to set the number of cluster-connected *nodes*.

¹The notation “I/O” is justified by historical arguments. In the current version of ICON, these MPI processes exclusively operate as *output* servers.

Increasing the number of nodes allows to use more computational resources, since a single compute node comprises only a limited number of PEs and OpenMP threads. On the other hand, off-node communication is usually more expensive in terms of runtime performance.

The computer platform at DWD, the NEC SX-Aurora, uses the batch system PBS to control the requested resources. When using the `qsub` command to submit a script file, the batch system PBS allows for specification of options at the beginning of the file prefaced by the `#PBS` delimiter followed by PBS commands. Similar settings can be made for the *Slurm* scheduler (`#SBATCH` prefix).

Finally, the user must set the correct options for the application launcher, which in most situations is the `mpirun` command.

8.3. Best Practice for Parallel Setups

8.3.1. MPI Tasks and OpenMP Threads

ICON employs both distributed memory parallelization and shared memory parallelization, i.e. a “hybrid parallelization”. Only the former type actually performs a decomposition of the domain data, using the de-facto standard MPI. The shared memory parallelization, on the other hand, uses OpenMP directives in the source code. In fact, nearly all `DO` loops that iterate over grid cells are preceded by OpenMP directives. For reasons of cache efficiency the `DO` loops over grid cells, edges, and vertices are organized in two nested loops: “`jb` loops” and “`jc` loops”¹ Here the outer loop (“`jb`”) is parallelized with OpenMP.

There is no straight-forward way to determine the optimal hybrid setup, except for the extreme cases: If only a single node is used, then the global memory facilitates a pure OpenMP parallelization. Usually, this setup is only feasible for very small simulations. If, on the other hand, each node constitutes a single-core system, a multi-threaded (OpenMP) run would not make much sense, since multiple threads would interfere on this single core. A pure MPI setup would be the best choice then.

In all of the other cases, the parallelization setup depends on the hardware platform and on the simulation size. In practice, 4 threads/MPI task have proven to be a good choice on x86-based systems. This should be combined with the *hyper-threading* feature, i.e. a feature of the x86 architecture where one physical core behaves like two virtual cores.

Starting from this number of threads per task the total number of MPI tasks is then chosen such that each node is used to an equal extent and the desired time-to-solution is attained – in operational runs at DWD this is ~ 1 h. In general one should take care of the fact that the number of OpenMP threads evenly divides the number of cores per CPU socket, otherwise inter-socket communication might impede the performance.

Finally, there is one special case: If an ICON run turns out to consume an extraordinarily large amount of memory (which should not be the case for a model with a decent memory scaling), then the user can resort to “investing” more OpenMP threads than it is necessary for the runtime performance. Doing so, each MPI process would have more memory at its disposal.

¹This implementation method is known as *loop tiling*, see also Section 9.4.

8.3.2. Blocking (**nproma**)

ICON runs on x86 computer architectures, as well as on classical vector processors, and graphics accelerators. Depending on the given architecture, the namelist parameter **nproma** must be adapted accordingly. The abbreviation **nproma** (probably) stands for *nombre de profondeurs maximal* – maximum of maximal depths.

For x86 architectures we suggest **nproma=16**, such that the line index fits into the cache. In contrast to this, on GPUs the **nproma** parameter should equal the number of grid points handled by each MPI process, see Section 8.5.2. Finally, for NEC's SX vector architecture, a much larger value **nproma=752**, say, is suitable to match the very wide vector length.

On GPU architectures there exists the useful option **nblocks_c** to specify the number of blocks instead of the block size **nproma** in the namelist. See Section 8.5.2 for details.

There is another important namelist option which is enabled for the NEC vector machine. On the NEC system the parameter **proc0_shift** (namelist **parallel_nml**, INTEGER value) is set to "1" which means that the first MPI rank (worker) does not take part in the actual computation – it gets 0 cells in the domain decomposition process. The reason behind this setting is the so-called "hybrid execution" on the NEC: The first MPI rank runs on an x86 CPU (vector host) and executes the read-in procedures only while the remaining processes run on a vector machine, doing the computational work. On x86 architectures or GPU accelerators the **proc0_shift** setting is unnecessary.

8.3.3. Mixed Single/ Double Precision in ICON

To speed up code parts strongly limited by memory bandwidth, the option exists to use single precision for variables that are presumed to be insensitive to computational accuracy – primarily the dynamical core and the tracer advection.

This affects most local arrays in the dynamical core routines, some local arrays in the tracer transport routines, the metrics coefficients, arrays used for storing tendencies or differenced fields (gradients, divergence etc.), reference atmosphere fields, and interpolation coefficients. Prognostic variables and intermediate variables affecting the accuracy of mass conservation are still treated in double precision. To activate the mixed-precision option, run the configure script with the **--enable-mixed-precision** flag.

8.3.4. Bit-Reproducibility

Bit-reproducibility refers to the feature that running the same binary multiple times should ideally result in bit-wise identical results. Depending on the compiler and the compiler flags used this is not always true if the number of MPI tasks and/or OpenMP threads is changed in between. Usually compilers provide options for creating a binary that offers bit-reproducibility, however this is often paid dearly by strong performance losses. With the NEC SX-Aurora compiler, it is however possible to generate an ICON binary offering bit-reproducibility with only little performance loss.

Bit-reproducibility is generally an indispensable feature for debugging. It is helpful

- for checking the MPI/OpenMP parallelization of the code. If the ICON code does not give bit-identical results when running the same configuration multiple times, this is a strong hint for an OpenMP race condition. If the results change only when changing the processor configuration, this is a hint for an MPI parallelization bug.
- for checking the correctness of new code that is supposed not to change the results.

8.4. Basic Performance Measurement

The ICON code contains internal routines for performance logging for different parts (setup, physics, dynamics, I/O) of the code. These may help to identify performance bottlenecks. ICON performance logging provides timers via the two namelist parameters `ltimer` and `timers_level` (namelist `run_nml`).



Note for advanced users: The built-in timer output is rather non-intrusive. It is therefore advisable to have it enabled also in operational runs.

With the following settings in the namelist `run_nml`,

```
ltimer          = .TRUE.
timers_level    =    10
```

the user gets a sufficiently detailed output of wall clock measurements for different parts of the code:

name	# calls		total min (s)	total max (s)
total	237	...	903.085	903.089
L integrate_nh	170640	...	884.128	892.143
L nh_solve	5972400	...	401.055	428.694
L nh_solve.veltend	7166880	...	36.469	51.376
...				
physics	49678	...	103.107	104.759
L nwp_radiation	10030	...	40.402	42.985
L radiation	220674	...	31.845	34.963
...				
model_init	711	...	59.875	59.876
...				

Note that some of the internal performance timers are nested, e.g. the timer log for `radiation` is contained in `physics`, indicated by the “L” symbol. For correct interpretation of the timing output and computation of partial sums one has to take this hierarchy into account.

- The column “total max (s)” contains the maximum timing in seconds (maximum over all MPI tasks, OpenMP master thread).

- The row `"model_init"` contains the measurements for the model setup (allocation, read-in, etc.).
- The row `"total"` contains the model run-time, *excluding* the initialization and finalization phase.

8.5. ICON on Accelerator Devices (GPUs)

Section author

M. Jacob,
DWD Numerical Models Division

ICON supports massively parallel accelerator devices such as GPUs (Graphics Processing Units). These devices are programmed using a parallelization model that is different from OpenMP and MPI, as described in Section 8.1. The following section describes the technique used and how to configure ICON for compilation and runtime.

ICON-Atmosphere (ICON-A) has been gradually ported to GPUs using OpenACC, with most components used for limited area and global weather prediction now supported (as of March 2024). In particular, the dynamical core and the NWP physics package have been ported in a joint effort by MeteoSwiss, CSCS, C2SM and DWD, with technical support from NVIDIA. In addition to the NWP application, ICON-A is also ready for climate-oriented studies on GPUs, such as those carried out by [Giorgetta et al. \(2022\)](#). Details of the porting strategy can also be found in that publication as well.

The GPU support for some ICON-A components is limited to certain variants and options. Unsupported model configurations will result in error warnings when executed with OpenACC. Supported and tested configurations can be found in the example namelists in the `run` directory in the `exp.mch_*` and `exp.dwd_run_ICON_09_R2B4N5_EPS*` experiments. It should be noted that the OpenACC version of ICON is not yet operational, and that most of these experiments only run for short periods, so software bugs may still be discovered after extended use. Therefore, it is advisable to consider the OpenACC version of ICON as experimental, but worth a try if a GPU machine is available.



OpenACC is an API for offloading programs written in C, C++, and Fortran from a host CPU to an attached accelerator device (see <https://www.openacc.org>). It uses compiler directives (pragmas) and is similar to classical OpenMP, which handles shared-memory parallelization within the host processor. OpenMP target offloading for accelerators is not supported in ICON.

OpenACC is used to parallelize the innermost loops with unit stride memory rather than the blocks, which are parallelized with OpenMP. Current GPUs used for HPC have dedicated device memory. For example, an NVIDIA A100 GPU provides up to 80 GB of memory (40 and 96 GB are also available). This means that OpenACC must also manage this dedicated accelerator memory.

The OpenACC code in ICON controls the parallel execution on the accelerators and as well as the memory management. The host CPU is always responsible for the accelerator management. Each ICON CPU process can only use one accelerator, but OpenACC acceleration and MPI process parallelism can be combined to use multiple accelerators. Direct MPI communication between GPU memory is possible by compiling ICON with the `--enable-mpi-gpu` configure option. OpenMP and OpenACC cannot be combined in ICON, although technically a program could use OpenMP for CPU thread parallelism and OpenACC for acceleration. OpenACC should not to be confused with CUDA, which is a proprietary API developed by NVIDIA for direct programming of NVIDIA GPUs.

8.5.1. Configuring and Compiling ICON-OpenACC

Compiling ICON for use with an accelerator requires a compiler with OpenACC support.

Currently, the best support for ICON-OpenACC is provided by *nvfort*, formerly known as the PGI compiler, which is part of the NVIDIA HPC SDK (<https://developer.nvidia.com/hpc-sdk>). The *nvfort* requires NVIDIA GPUs. The minimum tested version of the NVIDIA HPC SDK is 21.3, but it is recommended to use a more recent version such as 23.3. Avoid version 23.11 due to known bugs related to ICON. However 24.x versions should work, but they have been tested less intensively than 23.3.

The latest HPC Cray Fortran compiler *cce* version 16.0.1.1 also supports ICON-OpenACC on AMD GPUs. However, it is less extensively tested than *nvfort*.

The ICON repository contains configuration wrappers for various machines, including:

- `config/cscs/balfrin.gpu.nvidia` for *Balfrin* at CSCS
- `config/cscs/daint.gpu.nvidia` for *Piz Daint* at CSCS
- `config/dwd/linuxWS.gpu.nvidia-22.7` for Linux workstations
- `config/dwd/gpnl.gpu.nvidia-22.7` for *GPNL* at DWD
- `config/jsc/juwels.gpu.omp_nvnhpc-22.7` and `config/jsc/juwels.gpu.psmapi_nvnhpc-22.7` for *JUWELS* at JCE.
- `config/dkrz/levante.gpu.nvnhpc-22.5` for *Levante* at DKRZ
- `config/kit/hk.gpu.nvnhpc` for *HoreKa* at KIT

The Balfrin configuration will be operational after May 2024.

To compile ICON on a Linux workstation, create a build directory and configure and compile ICON as described in Section 1.3 using the following commands. The paths in the wrapper are prepared for DWD systems and the network mount `/uwork1` is required to access the NVIDIA SDK.


```
mkdir linuxWS.gpu.nvidia-22.7
cd linuxWS.gpu.nvidia-22.7
../config/dwd/linuxWS.gpu.nvidia-22.7
make -j 6
```

The configurations on *Piz Daint* and *Balfrin* use the package management tool *Spack* (see <https://spack.readthedocs.io>). The respective builds can be initialized by calling `config/cscs/daint.gpu.nvidia` and `config/cscs/balfrin.gpu.nvidia` respectively.

8.5.2. Special Namelist Options for ICON-OpenACC

Some namelist settings in ICON-OpenACC need to be adjusted for optimal performance.

The most critical setting is `nproma` (namelist `parallel_nml`), which controls the length of the “jc loops”. These loops are parallelized with OpenACC, so they should be longer than the number of parallel computing units on the accelerator, e.g. 3456 double precision cores for an NVIDIA A100 GPU. The larger the `nproma`, the faster the computation, but if it is too large, memory will be wasted and performance may suffer. Having fewer grid cells than arithmetic units is inefficient and will increase MPI communication.

For a nested single-domain experiment, `nproma` should be equal to the number of grid points handled by each MPI process. The aim is to run a very long jc loop, i.e. to have only one jb block loop over all cells (see also Section 9.4).

However, instead of setting `nproma` manually, the number of cell blocks can be set directly using `nblocks_c = 1` (namelist `parallel_nml`, INTEGER value). Note that setting `nblocks_c > 1` overrides `nproma`.

For a nested multi-domain experiment it is not possible to predetermine an optimal `nproma` because the numbers of grid points per domain handled by each process will be different. In this case, `nproma` must be set manually, and `nblocks_c` cannot be used. `nproma` should be set to a value equal to the maximum number of cells of all domains and processes. If this is not possible due to memory limitations, it should at least be greater than the number of grid points per process in the smallest domain. As a starting point for optimization, set `nproma = 20000` (with default `nblocks_c = 0`), then conduct a series of short benchmark experiments to find the optimal `nproma` value.

Another setting, `nproma_sub` (namelist `parallel_nml`, INTEGER value), controls the chunk size for sub-blocks used in the ecRad and RRTMGP radiation codes. The sub-blocks help to reduce the radiation memory footprint. For single-domain experiments, `nblocks_sub` (namelist `parallel_nml`, INTEGER value) can be used instead of `nproma_sub`. A recommended starting value for `nblocks_sub` is 6, but it is advisable to experiment with smaller or larger values to determine the optimal setting based on memory availability. In contrast, for nested multi-domain experiments, the optimal setting for `nproma_sub` cannot be predetermined and must be set manually. It is recommended to start with a value of 7000 and adjust it based on performance results obtained from a series of short benchmark experiments.

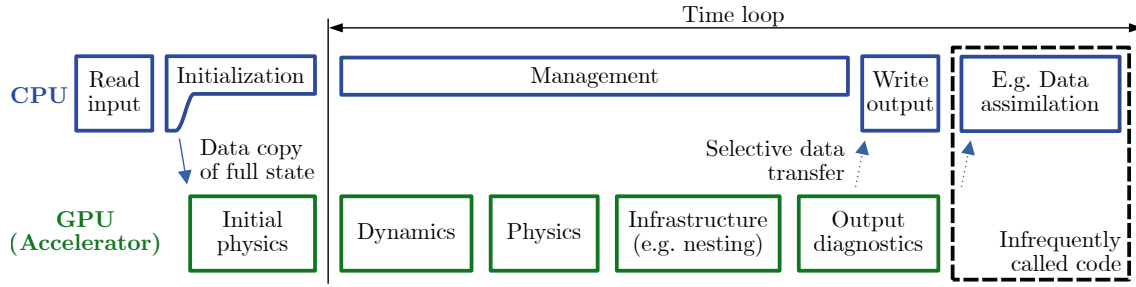


Figure 8.1.: Program flow of ICON-OpenACC.

The ecRad radiation scheme (`inwp_radiation =4`) provides a McICA radiation solver optimized for OpenACC, which can be enabled by setting `ecrad_isolver =2` (namelist `radiation_nml`, INTEGER value). The default value (`ecrad_isolver =0`) which is optimal for CPUs is not available with OpenACC.

The namelist setting `proc0_shift` is not needed for ICON-OpenACC and must not be set or set to its default value 0. `proc0_shift` is only needed for the experiments on the NEC SX Aurora system.

8.5.3. Implementation Details

Current GPU supercomputers have separate GPU and CPU memory. Data transfer between the two is relatively slow compared to the direct access of local memory. Therefore, recurring data transfers between CPU and GPU in the time loop must be avoided, as the compute intensity (ratio of computations to memory load) in ICON is relatively low. On the other hand, code that runs only once per simulation is less performance-critical, so that its porting effort can be saved. These principles lead to an ICON-OpenACC program flow (Figure 8.1), which runs the model initialization only on the CPU and activates the GPU when the first physics components are called. At the moment, it does not seem worthwhile to port a few infrequently called code packages, such as the output for data assimilation. However, these rare packages require the correct data transfer from GPU to CPU and possibly vice versa in the code.

The typical structure of an OpenACC-accelerated loop is presented in the following listing. The OpenACC code starts with the `!$ACC` pragma. Note the different levels that are parallelized using OpenMP or using OpenACC. In principle, ACC kernels (i.e. the code between `ACC PARALLEL` and `ACC END PARALLEL`) should run asynchronously so that the CPU can prepare the next kernel while the accelerator runs the current kernel. Usually, however, usually all kernels use the same async queue 1 and are therefore executed in order.

```
!$OMP PARALLEL
!$OMP DO PRIVATE(jb, jc, i_startidx, i_endidx)
  DO jb = i_startblk, i_endblk
```

```

        CALL get_indices_c(ptr_patch, jb, i_startblk, i_endblk,      &
                           i_startidx, i_endidx, i_rlstart_c, i_rlend_c)

        !$ACC PARALLEL DEFAULT(PRESENT) ASYNC(1) IF(lzacc)
        !$ACC LOOP GANG VECTOR
        DO jc = i_startidx, i_endidx

            var(jc, jb) = ... !

        ENDDO
        !$ACC END PARALLEL

    ENDDO
!$OMP END DO NOWAIT
!$OMP END PARALLEL

```

Certain code, such as utility functions or communication, is used during the initialization and the time loop. This means that such code must be able to be executed on CPU and the accelerator with the respective memory. The OpenACC standard provides the `IF(condition)` clause to execute a code on the accelerator if `condition` evaluates to `.TRUE.`, and on the CPU otherwise. All (modern) ICON subroutines that support OpenACC have an optional `lacc` argument that is used to make this decision. As `lacc` can only be used in the subroutine if it is passed as an argument (`present()` in the Fortran sense), the non-optional variable `lzacc` is used in the subroutine code. The value of `lzacc` can be derived from `lacc` by calling `set_acc_host_or_device(lzacc, lacc)`. The `set_acc_host_or_device` routine defaults to `.FALSE.` if `lacc` is not present. This means that code without OpenACC support does not need to be modified when porting a routine and adding `lacc` to the routine's argument list.

The dynamical core and legacy code use the global variable `i_am_accel_node` as a condition for the `IF()` clause. Originally, this variable was introduced for a validation mode where a non-accelerator-processes computes the entire domain on the CPU and compares it to the domain-decomposed results from the GPU nodes. Routines that use `lacc` and call other routines that depend on `i_am_accel_node` should have a `CALL assert_lacc_equals_i_am_accel_node(routine_name, lacc)` at the top.

The OpenACC directive code in ICON follows the “ICON OpenACC style and implementation guide” which can be found in the [ICON developer wiki on Gitlab](https://gitlab.dkrz.de/dwd-sw/icon-openacc-beautifier). The ICON ACC beautifier (<https://gitlab.dkrz.de/dwd-sw/icon-openacc-beautifier>) can be used to apply the spacing, comma, colon, capitalization and line continuation rules automatically as outlined in the style guide.

9. Programming ICON

Just because something doesn't do
what you planned it to do doesn't
mean it's useless.

Thomas Edison

The previous chapters' topics have been guided by questions of how to run ICON simulations in various settings and how to control and understand the model's characteristics. In this short chapter, instead, we will introduce ICON's inner workings, i.e. the code layout and the most important data structures.

The description is detailed enough to make it relatively easy for the reader to modify the Fortran code. We exemplify this in Section 9.4 by implementing an own simple diagnostic.

Alternatively, there is a flexible way to extend the code with your own routines and variables at run-time. The ICON Community Interface *ComIn* is a plug-in mechanism that even enables the integration of Python scripts. We will discuss this relatively new approach at the end of the chapter.

9.1. Representation of 2D and 3D Fields

We begin with a suitable representation of two- and three-dimensional fields. Here, we refer to a discrete variable as a *2D field* if it depends on the geographical position only. A *3D field*, in addition, contains a vertical dimension, associated with the grid column.

Indexing. Recalling the unstructured nature of ICON's computational grids (see Section 2.1) there is no obvious order of the cells in a 2D array like indexing them according to longitudes and latitudes. Instead, we just order the cells in a deliberate way and index them in this order with ascending integer numbers. This means that our 2D field becomes a 1D array, referenced by the cell indices as subscript values.

Most arrays are associated with the centers of the triangular grid cells, but we do that in a similar way for the edges and vertices of the triangles. An extension to 3D fields, i.e. including a vertical dimension, results in 2D arrays, the first index being the cell (or edge or vertex) index, the second index being the height level.

Blocking. For reasons of cache efficiency nearly all DO loops over grid cells, edges, and vertices are organized in two nested loops: “jb loops” and “jc loops”. Often, the outer loop (“jb”) is parallelized with OpenMP.

With respect to the data layout, this means that the long vector of the 2D array is split into several chunks of a much smaller length `nproma` (this is a run-time parameter, defined in the namelist `parallel_nml`, see also Section 8.3.2). We store the long vector in an array with two indices, the first index counting the elements in a block (*line index*), the second index counting the blocks. The last block may be shorter since `nproma` is not necessarily a divisor of the number of cells. The blocking procedure is illustrated in the lower half of Figure 9.1. There exist auxiliary functions `idx_no`, `blk_no` and `idx_1d`, which help to calculate the blocked indices from the 1D array index and vice versa (declared in the module `mo_parallel_config`).

Finally, let us consider the 3D fields that were indexed with the cell index as the first dimension and the second being the vertical coordinate. With index blocking, these fields will be stored in 3D arrays with the first index counting the elements in a block, the second index counting the levels and the third index counting the blocks. The reason is that the blocks are often passed one by one to some subprograms which are called in a loop over the blocks. Since Fortran stores arrays in column-major order, the data for a single `jb` is stored contiguously in memory. Thus we can pass this chunk of data to the subprograms without any reshaping of the arrays.

Domain decomposition. Domain decomposition is, naturally, a prerequisite for scalability of grid point models on modern parallel computers. For large scale realistic ICON setups and with operational core counts in the range of tens of thousands, the use of persistent global-sized arrays is unacceptable. Each model domain is therefore distributed onto several processors¹. This means that we have only certain regions of a domain on each processor.

Generally, several subdivision steps are performed recursively. The division criterion (module `mo_setup_subdivision.f90`) subdivides a partition wrt. the cell *latitudes* if the range of covered latitudes is larger than the range of longitudes, otherwise the subdivision operates on the *longitudes*. This procedure vaguely reminds of the creation of a kd-tree², but the method also accounts (empirically) for the convergence of meridians and it does some boundary smoothing to reduce inter-process communication.

Each processor’s region consists of an inner portion and a lateral boundary portion. The latter may be either a lateral boundary for the entire domain or a *halo region*, i. e. a lateral boundary of the partial domain which is overlaid with neighboring partial domains. The halo region (which is also known as a *ghost-cell region*) is illustrated in the upper half of Figure 9.1.

The programmer is responsible for the distribution of the data among the processors and the correct communication through MPI calls. This means that all halo regions have to be updated by the neighboring partial domains. We will sketch this synchronization process in Section 9.2.4 below.

¹In the following, we will use the generic term “Processing Element” (PE).

²See https://en.wikipedia.org/wiki/K-d_tree.

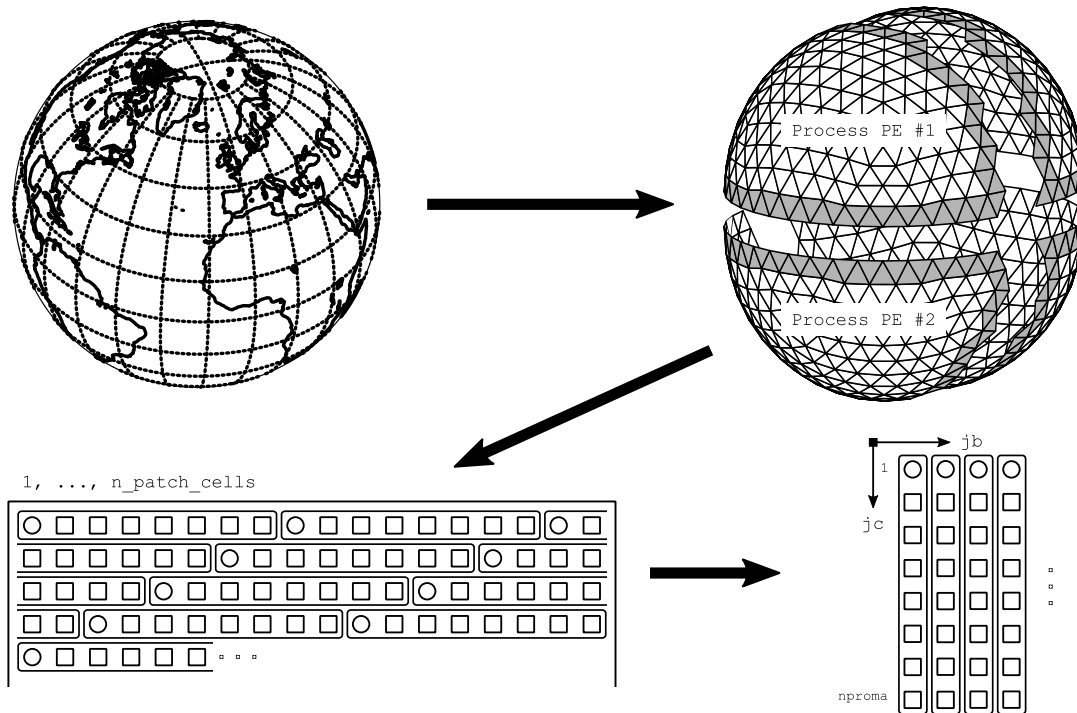


Figure 9.1.: Illustration of the 2D field representation. The original spherical domain is decomposed (light-gray: halo region). Afterwards, the long vector of grid cells is split into several chunks of a much smaller length `nproma`.

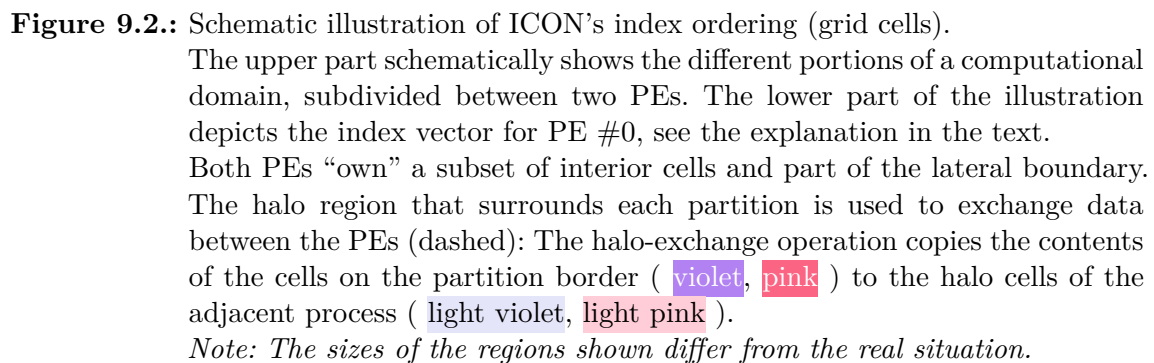


Keep in mind that it is the width of the halo region which defines a size limit for your stencil calculations: It is not possible to include cells in a stencil operator which extend beyond the halo region!

Index ordering. After the domain decomposition, which takes place in the model initialization phase, each PE performs a sorting of its locally allocated cells (and edges and vertices). This *local* index ordering is determined by the `refin_XXX_ctrl` index which counts the distance from the lateral boundary in units of cell/edge/vertex rows. In particular, note the `refin_c_ctrl` array which already played a role for the preparation of lateral boundary input data, see Section 2.3. Portions of the triangular cells correspond to different values of `refin_c_ctrl`, which allows a sorting into the following categories: the cell rows at the lateral boundary, the nudging zone, the inner cells, and the halo region. Of course, for a global domain only the two latter categories exist.

The upper part of Figure 9.2 schematically shows the different parts of a computational domain, subdivided between two PEs: Each PE “owns” a subset of interior cells and part of the lateral boundary. The halo region is shared between the PEs.

The lower part of Figure 9.2 visualizes the ordering of the grid portions in the index vector. It can be seen that the leftmost indices (i.e. the smallest subscripts) correspond to the lateral boundary region, followed by the prognostic cells. The sorting of these prognostic

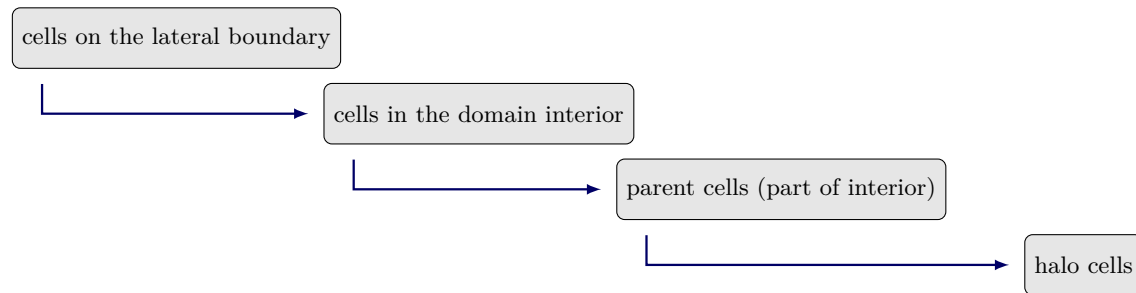


Thus the indices are ordered in such a way that typical iterations over grid portions like prognostic cells, lateral boundary points etc. can be realized without conditional statements. Each portion is annotated by an index. For convenience, there exists the auxiliary function `get_indices_c` (declared in the module `mo_loopindices`) which helps to adjust the loop iteration accordingly: For a given value of `refin_c_ctrl` and a specific block index we get the start and end indices to loop over.

194

The following detailed explanation in this paragraph is formulated for the set of *cell* indices of a specific MPI task. However, it applies to edges and vertices in a similar fashion.

Let us assume that the given MPI task comprises cells on the lateral domain boundary, as well as cells which are the parent cells for one or more child domains. Then, as noted previously, the general 1D ordering of cells is



More precisely, the ordering is prescribed by *cell rows*, which can be identified by the values in the array `refin_c_ctrl`:

- SEC-1 Cell rows in the lateral domain boundary have positive `refin_c_ctrl` numbers: The outermost (full) cell row has the number 1, the adjacent cell row has the no. 2, and so on. In total, `max_rlcell` = 5 cell rows are part of this section.
- SEC-2 This section contains unordered, prognostic, non-halo cells. The corresponding `refin_c_ctrl` index is 0.
- SEC-3 Cell rows which are parent cells of finer domains are denoted by negative `refin_c_ctrl` numbers (cell with the value -1 are at the outer boundary of a nested domain). Note that this SEC-3 does no longer exist in ICON's current domains (patches), with the exception of so-called *local parent patches*, see below for explanation.
- SEC-4 Halo cells (when they do not overlap with a child patch) have a negative numbering ranging between `min_rlcell` and `min_rlcell_int-1`. In total there are $(\text{min_rlcell_int} - \text{min_rlcell}) = 4$ slots available to denote halo cell rows, although in practice only two full cell rows are used for halo exchanges. Also note that halo cells are numbered as *half* cell rows to allow for a more fine-grained definition of MPI exchange patterns.

As it has been described in the previous sections, ICON's arrays are logically stored as blocked 2D-arrays (`nproma, nblks_c`). For every cell row of each of the above sections, we can query the first and the last block for this row with the auxiliary arrays `start_block(:)` and `end_block(:)`. These arrays are allocated in the ICON code with array dimensions (`min_rlcell:max_rlcell`).

The halo cells deserve one additional remark – note the comment “*when not overlapping with lateral boundary*” in Fig. 9.2: In the special case, when no lateral boundary is present (for a global grid, say, or when a PE operates only on an inner portion of the domain),

the halo cells are stored in a contiguous fashion at the end of the index vector. When a lateral boundary is present, however, there exist some halo cells which also belong to the lateral boundary. These cells are then sorted into the leading part of the index array, [SEC-1](#). This exception avoids zero-initialization of some fields during runtime, and it has no practical drawbacks in practice because the ability to address boundary cells in a contiguous fashion is much more important. Note that this exceptional sorting of halo cells does not affect *all* halo cells of the lateral boundary region, though, but only their outermost rows. A possibility to distinguish between prognostic cells and halo points is provided by the `decomp_domain` data structure and the “owner info” field, see the following section.

9.2. Data Structures

This section describes ICON’s most important data structures. The majority of the data structures mentioned here exists several times, since a separate structure is created for each computation domain.



The number of computational domains can be taken from the variable `n_dom` which corresponds to the number of entries in the namelist parameter `dynamics_grid_filename`, see [Page 121](#). Here, the reduced radiation grid ([Section 3.10](#)) is not taken into account.

9.2.1. Description of the Model Domain: `t_patch`

The `t_patch` data structure contains all information about the grid coordinates and topology, as well as parallel communication patterns and decomposition information. It is declared in `src/shr_horizontal/mo_model_domain.f90` as an array of length `n_dom`, where the coarsest base grid is denoted by the index 1, while the refined domains are denoted by numbers 2, 3 and so on.

All contained data arrays and indices relate to the index/block ordering described in [Section 9.1](#). Non-existent indices, e. g. neighbors of cells located at the domain boundary, are denoted by `-1`. The most important contents of the `t_patch` data structure are

t_patch

<code>grid_filename</code>	character string, containing grid file name
<code>ldom_active</code>	indicator if current model domain is active, see Section 5.2
<code>parent_id</code>	domain ID of parent domain
<code>child_id(1:n_chiiddom)</code>	list of child domain ID's
<code>n_patch_cells/edges/verts</code>	number of locally allocated cells, edges ...
<code>n_patch_XXXX_g</code>	global number of cells, edges and vertices
<code>nblks_c/e/v</code>	number of blocks
<code>npromz_c/e/v</code>	chunk length in last block
<code>cells / edges / verts</code>	lower-level data structures, see below
<code>comm_pat_c/e/v</code>	halo communication patterns, see Section 9.2.4
	:

When it comes to dimensioning fields, the application programmer normally uses the size `n_patch_cells` (or `n_patch_cells_g` for global arrays which should generally be avoided) for cells, and `n_patch_edges`, `n_patch_verts` for edges and vertices, respectively. The product `nproma*nblks_c` only provides an upper bound: The last block of `nproma` indices does not necessarily have to be filled completely, which is indicated by the variable `npromz_c`, see the explanation in Section 9.1.

The data members `cells`, `edges`, and `verts`, which are of the types `t_grid_cells`, `t_grid_edges`, and `t_grid_vertices`, respectively, give us information about the grid cells themselves, in particular about their geographical coordinates. For example,

t_grid_cells

<code>center(:, :)</code>	longitude and latitude of cell circumcenters, dimensions: [1:nproma , 1:nblks_c]
<code>neighbor_idx(:, :, :)</code>	line indices of triangles next to each cell, dimensions: [1:nproma , 1:nblks_c, 1:3]
<code>decomp_info</code>	information on domain decomposition
	:

Essentially, all data arrays which are contained in the grid files and which are described in Section 2.1.1 have a counterpart in this derived data type.

Besides, the data member `decomp_info` which separately exists for cells, edges and vertices, deserves additional comments. Its data type `t_grid_domain_decomp_info` is declared in `/src/parallel_infrastructure/mo_decomposition_tools.f90`:

t_grid_domain_decomp_info

<code>glb_index(:)</code>	global index of local cell, dimension: 1:n_patch_cells
<code>decomp_domain(:, :)</code>	domain decomposition flag, dimensions: [1:nproma , 1:nblks_c]
	:

The global index (`glb_index`) is particularly useful to perform operations (or write out data) which must not depend on the parallel domain decomposition of the model run. The “owner info” (`decomp_domain`) can be used to distinguish between prognostic cells and halo points whose values are just copied from adjacent PEs. For cells, those cells that are owned by the current PE are denoted by a value of 0.

9.2.2. Date and Time Variables

When installing own functions within ICON’s time loop, the question for the current (simulation) time naturally arises. All global date and time variables are contained in the data structure `time_config`, which is of the derived data type `t_time_config` and declared in `src/configure_model/mo_time_config.f90`. The dates are initialized with the corresponding namelist parameters given in Section 5.1.1.

t_time_config

<code>tc_exp_startdate</code>	experiment start (tc means “time control”)
<code>tc_exp_stopdate</code>	experiment stop
<code>tc_startdate</code>	start of current simulation. In case of restart this is the date at which the simulation has been continued.
<code>tc_stopdate</code>	end of single run
<code>tc_current_date</code>	current model date
	:

The dates and time spans make use of the `mtime` calendar library³ which is precise up to milliseconds without round-off errors. The `mtime` library resides in the directory `externals/mtime`. It is written in C and has a Fortran interface (module `mtime`).

We motivate the use of the `mtime` module by two examples. First, we perform a date calculation, adding a time span of 1 day to a given date. We make use of two variables: `mtime_date` (TYPE(`datetime`), POINTER) and `mtime_td` (TYPE(`timedelta`), POINTER).

```
mtime_td  => newTimedelta("P01D")
mtime_date => newDatetime("2014-06-01T00:00:00")

mtime_date = mtime_date + mtime_td
```

³The NWP mode uses the *proleptic* Gregorian calendar that is a backward extension of the Gregorian calendar to dates before its introduction October 15, 1582.

```
CALL datetimestostring(mtime_date, dstring)
WRITE (*,*) "2014-06-01T00:00:00 + 1 day = ", TRIM(dstring)

CALL deallocateDatetime(mtime_date)
CALL deallocateTimedelta(mtime_td)
```

As a second example, we demonstrate the `mtime` event mechanism which may be used to start certain processes in the program. An event (`TYPE(event)`, `POINTER`) is defined by a start date, a regular trigger interval and an end date. Besides, let `RefDate` denote the event *reference date* (anchor date) in our example. Then the event triggers every `RefDate + k * interval`, but only within the bounds given by `startDate` and `endDate`.

```
advectionEvent => newEvent('advection', RefDate, startDate, &
    & endDate, interval)
IF (isCurrentEventActive(advectionEvent, current_date)) THEN
    WRITE (*,*) 'Calculate advection!'
ENDIF
CALL deallocateEvent(advectionEvent)
```

9.2.3. Data Structures for Physics and Dynamics Variables

On each model domain we need the same collection of 2D and 3D fields in order to describe the state of the atmosphere. These fields are collected in the data structure `t_nh_state`. This derived type and the following types are declared in `src/atm_dyn_iconam/mo_nonhydro_types.f90`.

First, the prognostic fields, which are integrated over time, are collected in the data structure `t_nh_prog`. Elements of `t_nh_prog` are allocated for each time slice that is needed for the time integration. For the nonhydrostatic time integration, the number of time slices is two, time t for the current time and $t + \Delta t$ for the prediction.

`t_nh_prog`

<code>w</code>	orthogonal vertical wind [m s^{-1}]
<code>vn</code>	orthogonal normal wind [m s^{-1}]
<code>rho</code>	density [kg m^{-3}]
<code>exner</code>	Exner pressure
<code>tke</code>	turbulent kinetic energy [$\text{m}^2 \text{s}^{-2}$]
<code>tracer</code>	tracer concentration [kg kg^{-1}]
	:

A global variable `p_nh_state` of type `t_nh_state` is instantiated in the module `/src/atm_dyn_iconam/mo_nonhydro_state.f90`. This is an array whose index corresponds to the model domain. The density of the atmosphere as state variable of the nonhydrostatic dynamical core is therefore given as

```
p_nh_state(domain)%prog(time slice)%rho(index, level, block)
```

Regarding the `time slice` argument, we briefly comment on ICON's handling of the two-time-level scheme and the mechanism to avoid reallocation or unnecessary data copies:

For each prognostic variable in the two-time-level scheme, two arrays are pre-allocated:

```
p_nh_state%prog(1)%field(:, :)
p_nh_state%prog(2)%field(:, :)
```

Additionally, we introduce two global INTEGER index variables `nnow` and `nnew` which we initialize at model start with

```
nnow = 1
nnew = 2
```

The result values of the integration scheme (time slice $t + \Delta t$) are stored on the `nnew` time level. We therefore access the data on this time level by

```
p_nh_state%prog(nnew)%field(:, :)
```

While the calculations in the dynamical core fill this array with values, the prognostic state of the “old” time step can be accessed by

```
p_nh_state%prog(nnow)%field(:, :)
```

Then, at the end of each dynamic (sub-)step, the time step `n+1` becomes the “old” one, while the time step `n` is freed and can be used as the new working array for the time stepping. This operation does not require any copying but merely exchanges the roles of `nnow` and `nnew`:

```
CALL swap(nnow, nnew)
```

Unfortunately, the whole process is complicated by the following two facts: First, `nnow`, `nnew` are defined for each domain separately. The above examples therefore require the domain index `jg`, i.e. `nnow(jg)`, `nnew(jg)`. Second, as it has been explained in Section 3.7.1, different integration time steps are applied for efficiency reasons. A separate `nnow_rcf/nnew_rcf` accounting is required for the basic time step which is used for tracer transport, numerical diffusion and the fast-physics parameterizations, to distinguish it from the short time step used within the dynamical core⁴.

The data type `t_nh_diag` (defined in `src/atm_dyn_iconam/mo_nonhydro_types.f90`) contains a collection of diagnostic fields, determined by all prognostic variables, boundary conditions and the compositions of the atmosphere.

<code>t_nh_diag</code>	
<code>u</code>	zonal wind [m s^{-1}]
<code>v</code>	meridional wind [m s^{-1}]
<code>temp</code>	temperature [K]
<code>pres</code>	pressure [Pa]
	\vdots

Similar to the prognostic fields, the domain-wise data of type `t_nh_diag` can be accessed via `p_nh_state(domain)%diag`.

⁴Here, the suffix `rcf` stands for “reduced calling frequency”.

9.2.4. Parallel Communication

To simplify the data exchange between neighboring domain portions, ICON contains the synchronization routine `exchange_data`, defined in the module

```
src/parallel_infrastructure/mo_communication.f90.
```

This takes the specific halo region as an argument and several exchange patterns are pre-defined for each domain (see the data type `t_comm_pattern` and the derived data type `t_patch`). For example, `comm_pat_c`, defines the halo communication for *cells* which are owned by neighboring processes. The subroutine call

```
CALL exchange_data(patch%comm_pat_c, array)
```

would perform a typical synchronization of the halo regions.

Additionally, there exist variants of the `exchange_data` routine for gather operations. Calling `exchange_data` with an argument of type `t_comm_gather_pattern`, typically the pre-defined data structure `patch%comm_pat_gather_c`, takes elements from many processes and gathers them to one single process. There are corresponding data structures for communicating edges and vertices.

As it has been noted in Section 8.2, there exist different process groups in ICON: I/O, restarting and computation. These groups are related to MPI communicators which are defined in the module `src/parallel_infrastructure/mo_mpi.f90`. Probably the most important MPI communicator in the ICON code is `p_comm_work` (which is identical to the result of a call to `get_my_mpi_work_communicator()`). This is the MPI communicator for the *work group*.

The work group has a total size of `num_work_procs`, where each process may inquire about its rank by calling `get_my_mpi_work_id()`. On a non-I/O rank, its work group comprises all processes which take part in the prognostic calculations. It is therefore used by the `exchange_data` synchronization routine.

A final remark is related to the endless stream of the status log (screen) output: The `process_mpi_stdio_id` is always the 0th process of the MPI communicator `process_mpi_all_comm`, which is the MPI communicator containing all PEs that are running this model component. A typical code line for printing out a message to screen would be

```
IF (my_process_is_stdio()) write (0,*) "Hello world!"
```

With this, the message print-out would be suppressed on all PEs 1, 2, ...

For convenience, there exists an auxiliary subroutine `message()` in ICON, whose exact purpose is what we achieved manually above:

```
CALL message('caller', 'message text')
```

The `message()` subroutine is located in the module `mo_exception` and restricts the print-out to PE #0. It takes the caller's subroutine name as an additional argument.

9.3. NWP Call Tree

All of ICON's NWP and infrastructure modules, however numerous they may be, can roughly be classified into an initialization phase, the time integration loop and the clean-up phase. In Figure 9.3 we restrict ourselves to the most important subprograms.

These are listed together with a short description of their location and purpose, which, of course, change gradually between released versions. We recommend to compare this to the flow chart of processes in the physics-dynamics coupling, see Fig. 3.9.

9.4. Implementing Own Diagnostics

A thorough description of how to modify the ICON model and implement one's own diagnostics would certainly be a chapter in its own right. Here, we try to keep things as simple and short as possible and apply some of the features from the previous sections.



Adding new modules: The dependency generator of ICON automatically detects Fortran (.f90) and C (.c) files in the `src` directory. These files are automatically included in the compilation process. Thus, when adding new source code files to `src`, then these files are automatically recognized. On the other hand, when creating a backup copy of a file it must not end with `.f90` or `.c` to avoid ambiguous module and subroutine names.

Adding new fields. ICON keeps so-called *variable lists* of its prognostic and diagnostic fields. This global registry eases the task of memory (de-)allocation and organizes the field's meta-data, e.g., its dimensions, description and unit. The basic call for registering a new variable is the `add_var` command (module `mo_var_list`). Its list of arguments is rather lengthy and we will discuss it step by step.

First, we need an appropriate **variable list** to which we can append our new variable. For our example we choose an existing diagnostic variable list, defined in the module `mo_nonhydro_state`:

```
p_diag_list => p_nh_state_lists(domain)%diag_list
```

The corresponding type definition can be found in the module `mo_nonhydro_types`. There, in the derived data type `t_nh_diag`, we place a **2D variable pointer**

```
REAL(wp), POINTER :: newfield(:, :)
```

which we can afterwards access as `p_nh_state(domain)%diag%newfield`.

Note that we have not yet allocated the variable, but have only defined an anchor for accessing it.

Each ICON variable must be accompanied by appropriate **meta-data**. In this example we will initialize GRIB and NetCDF variable descriptors for a variable located in the cell circumcenters (mass points). To keep this presentation as short as possible we have omitted the necessary `USE` statements:

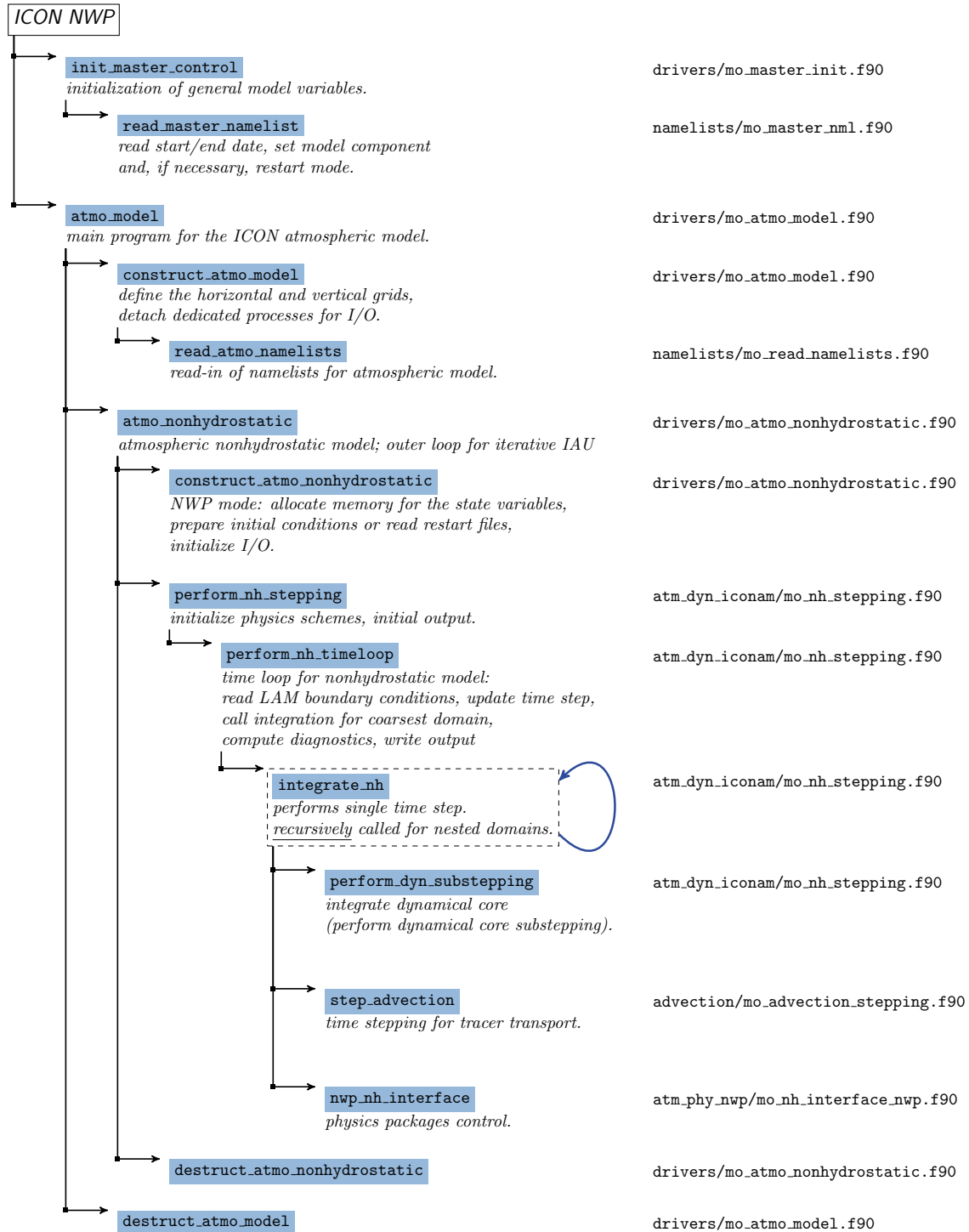


Figure 9.3.: Call tree of ICON's NWP component (note that this list has been restricted to the most important subroutines).

```

cf_desc    = t_cf_var("newfield", "unit", "long name", DATATYPE_FLT32)
grib2_desc = grib2_var(discipline, parameterCategory, parameterNumber, &
                      DATATYPE_PACK16, GRID_UNSTRUCTURED, GRID_CELL)

```

The derived types `t_cf_var` and `t_grib2_var` are defined in the modules `mo_cf_convention` and `mo_grib2`, respectively. The expression `grib2_var` is actually a call to a constructor function, also defined in `mo_grib2`, which takes a triple of integers (*discipline*, *parameterCategory*, *parameterNumber*) as the field specifier.

Let us create an INTEGER array of length 2 with the name `shape2d_c`, denoting the **dimensions** of the new variable. The dimensions of a 2D field will be explained below. Here we take them as given:

```
shape2d_c = (/ nproma, nblks_c /)
```

Now, with the essential ingredients at hand, we define our new field by the following call. We will place it at the very end of the subroutine `new_nh_state_diag_list` in the module `mo_nonhydro_state`.

```
CALL add_var( p_diag_list, 'newfield',           &
              p_nh_state(domain)%diag%newfield, &
              GRID_UNSTRUCTURED_CELL, ZA_SURFACE, &
              cf_desc, grib2_desc,               &
              ldims=shape2d_c, lrestart=.FALSE. )
```

The INTEGER parameters `GRID_UNSTRUCTURED_CELL` and `ZA_SURFACE` define the type of the horizontal grid and the (trivial) vertical axis. From now on the new field can be specified in the output namelists that were described in Section 7:

```
&output_nml
...
ml_varlist = 'newfield'
/
```



The `extra_2d` and `extra_3d` fields: When debugging the model code, it is often advantageous to be able to output intermediate results and ad hoc calculated diagnostic fields. However, it would be an unnecessary effort to define and allocate new variables especially for these test situations. ICON has a special mechanism for this purpose:

By setting the namelist parameter `inextra_2d` (namelist `io_nml`, INTEGER value) or `inextra_3d` (namelist `io_nml`, INTEGER value), respectively, a number of 2D or 3D cell-based floating-point arrays with the names `extra_2d1`, `extra_2d2`, ..., and `extra_3d1`, `extra_3d2`, ... is automatically created. Inside the model code, these can be accessed as

```
p_nh_state(domain)%diag%extra_2d_ptr(1)%p_2d(:, :)
```

and similar. Note that all extra variables are actually stored in common buffers

```
p_nh_state(domain)%diag%extra_2d(:, :, 1:inextra_2d)
p_nh_state(domain)%diag%extra_3d(:, :, :, 1:inextra_3d)
```

The fields can be used as output buffers for the temporary output data.

Looping over the grid points. Of course, the newly created field '*newfield*' still needs to be filled with values. As explained in Section 9.1 above, nearly all DO loops that iterate over grid cells are organized in two nested loops: “jb loops” and “jc loops”. Here the outer loop (“jb”) is parallelized with OpenMP and limited by the *cell block number* `nblks_c`. The innermost loop iterates between 1 and the block length `nproma`.

Since the ICON model is usually executed in parallel, we have to keep in mind that each process can perform calculations only on a portion of the decomposed domain. Moreover, some of the cells between interfacing processes are duplicates of cells from neighboring sub-domains (so-called *halo cells*). Often it is not necessary to loop over these halo points, since they will be updated by the next parallel synchronization call.

The auxiliary function `get_indices_c` (declared in the module `mo_loopindices`) helps to adjust the loop iteration accordingly:

```
i_startblk = p_patch(domain)%cells%start_block(grf_bdywidth_c+1)
i_endblk   = p_patch(domain)%cells%end_block(min_rlcell_int)

DO jb = i_startblk, i_endblk
    CALL get_indices_c(INp_patch(domain), INjb, INi_startblk, INi_endblk, OUTis, OUTie, &
                     INgrf_bdywidth_c+1, INmin_rlcell_int)
    DO jc = is, ie
        p_nh_state(domain)%diag%newfield(jc,jb) = ...
    END DO
END DO
```

The constants `grf_bdywidth_c` and `min_rlcell_int` can be found in the modules `mo_impl_constants_grf` and `mo_impl_constants`, respectively. Note that these constants have to be inserted in `start_block`, `end_block` and also in the argument list of `get_indices_c`. A graphical interpretation of these constants is provided by Figure 9.2. The loop example therefore iterates over all prognostic cells (denoted by the blue area).



Loop exchange: The special pre-processor flag `__LOOP_EXCHANGE` (configure option `--enable-loop-exchange`) can be found in numerous places of the ICON model code. It causes a loop interchange in many performance critical loops: If applied, the variable used in the inner loop switches to the outer loop.

Usually, this means that the loop over the vertical levels becomes the fastest running loop, compared to the iteration indices for the horizontal location. When arrays do not contain vertical levels, access to array elements may take advantage of the CPU cache. For a disabled loop exchange, the innermost loop for the horizontal location can benefit from vectorization.

Placing the subroutine call. Having encapsulated the computation together with the DO loops in a custom subroutine, we are ready to place this subroutine call in between ICON’s physics-dynamics cycle.

Let us take a look at Figure 3.9: The outer loop “Dynamics → Physics → Output” is contained in the core module `mo_nh_stepping` inside the `TIME_LOOP` iteration. For diagnostic calculations it is important to have all necessary quantities available for input. On the other hand the result must be ready before the call to the output module,

```
CALL write_name_list_output(jstep, ...)
```

The fail-safe solution here is to place the call immediately above this call.

Having inserted the call to the diagnostic field computation, we are done with the final step. Recompile the model code and you are finished with extending the model!



Style recommendations: When writing your own extensions to ICON it is always a good idea to keep an eye on the quality of your code.

Make sure that there is **no duplicate functionality** and try to improve the readability of your subroutines through **indentation**, **comments** etc. This will make it easier for other developers to understand and assimilate. Better introduce own **modules** with complete interfaces and avoid `USES` and `PUBLIC` fields.

A good starting point for your own project are the **template files**, given in the subdirectory `src/templates` of the ICON source code. These provide examples for modules, functions and subroutines.

Additional remarks: 3D fields, accumulated quantities. For the sake of brevity, only the simple case of two-dimensional fields has been discussed so far.

Three-dimensional fields would have an additional dimension for the **column levels**:

```
shape3d_c = (/ nproma, nlev, nblks_c /)
```

This information needs to be provided to the constructor `add_var(...)`, together with an `INTEGER` parameter which indicates the type of the vertical axis:

Usually, the ICON generates its vertical axis on-the-fly, i.e. during the model setup. The user may choose between the hybrid Gal-Chen coordinate and the (more common) SLEVE coordinate via namelist parameters, see Section 3.4. However, it is practically impossible to encode the exact vertical coordinate parameters themselves in the data sets which are produced by the ICON model. Apart from very basic information like the number of vertical levels, only a number identifying the special vertical grid used is provided. This indirect approach is indicated by the parameter `ZA_REFERENCE`.

Finally, it is often necessary to **reset accumulated quantities** in regular intervals. This can be achieved by

```
action_list = actions(new_action(ACTION_RESET, interval))
```

For example, by setting `interval = "PT06H"`, the respective field would be reset to zero every 6 hours.

```
CALL add_var( p_diag_list, 'newfield', &
             p_nh_state(domain)%diag%newfield, &
             GRID_UNSTRUCTURED_CELL, ZA_REFERENCE, &
             cf_desc, grib2_desc, &
             action_list=actions(new_action(ACTION_RESET,interval)), &
             ldims=shape3d_c, lrestart=.FALSE. )
```

9.5. The ICON Community Interface ComIn

Section authors

N.-A. Dreier, M. Haghighatnasab and F. Prill
DWD / DKRZ

9.5.1. Introduction

The ICON Community Interface (ComIn) organizes the data exchange and simulation events between the ICON model and third party modules (“plugins”) activated at run-time. While the adapter library is coded in Fortran 2003, it offers interfaces for external plugins developed in C/C++ and the Python programming language.

The ComIn library has been published as part of the first open source version of the ICON model, released January 2024 under the BSD-3C license. However, it is also possible to download and build the ComIn library independently without ICON, see the [ComIn releases homepage](#).

The ComIn project started in 2022 as a collaboration between DWD, DLR-IPA and DKRZ⁵. Publications – in addition to [Prill \(2023\)](#), [Hartung et al. \(2023\)](#) – are currently in preparation, but extensive documentation can already be found on the project homepage <https://gitlab.dkrz.de/icon-comin>, including a user guide and the technical interface description. The ComIn developer team can be contacted via the e-mail address comin@icon-model.org.

Figure 9.4 illustrates the basic idea of the community interface: Plugins are enabled via a special namelist `comin_nml` provided that the ICON model has been built with the configure option `--enable-comin`. Afterwards, plugins written in Fortran, C/C++ or Python can be dynamically loaded at run-time as shared libraries. For example, a Python script `simple_python_plugin.py` is attached to ICON by the following namelist settings:

```
&comin_nml
  plugin_list(1)%name          = "simple_python_plugin"
  plugin_list(1)%plugin_library = "libpython_adapter.so"
  plugin_list(1)%options       = "simple_python_plugin.py"
/
```

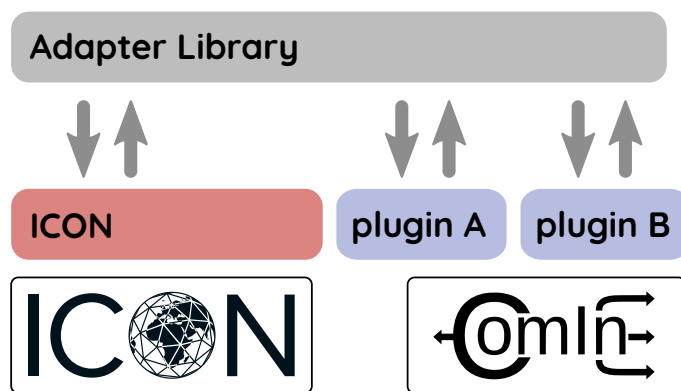


Figure 9.4.: General illustration of the ComIn plugin mechanism. Third party modules (“plugins”) are loaded dynamically at run-time and connected to the ICON model.

The individual functions that are defined within the plugin can be attached to about 40 entry points, depending on namelist settings and program flow. Entry points denote events during the ICON model simulation, which can trigger a subroutine call of the plugin. The distribution of entry points in the ICON code is illustrated in Figure 9.5. Within the ICON source code, look out for `CALL icon_call_callback` in order to find the exact locations. ComIn takes care of the callbacks of these plugin functions and handles the data exchange.

The use of ICON with ComIn comes with certain restrictions. Firstly, ICON initiates the call to ComIn from the plugins rather than the reverse. This means, for example, that plugins have no influence over ICON’s restart behavior. Individual processes in the ICON host model can only be deactivated using ICON namelist switches in order to replace them by a plugin routine.

Furthermore, the granularity of interaction between ICON and the plugins is limited to the block loop level. Only global variables are exposed for access, and while MPI exchanges are possible, only process-local MPI partitions can be accessed directly. Lastly, there is no support for asynchronicity between ICON and the third-party modules.



Similar to the approach pursued by ICON ComIn, the *Plume* project of the European Center for Medium-Range Weather Forecasts (ECMWF) aims to develop a plug-in mechanism for the IFS model of the ECMWF. The project has been in intensive development since around November 2022 and has recently reached version 0.2.0. Further information and the current status of the project can be found on GitHub at <https://github.com/ecmwf/plume> and in a corresponding conference paper (Bonanni et al., 2023).

9.5.2. Example Plugins

ComIn is bundled together with several example plugins written in Fortran, C and Python. After cloning the ICON source code, these can be found in `externals/comin/plugins`.

⁵See the `AUTHORS.md` file for a full list of contributing developers.

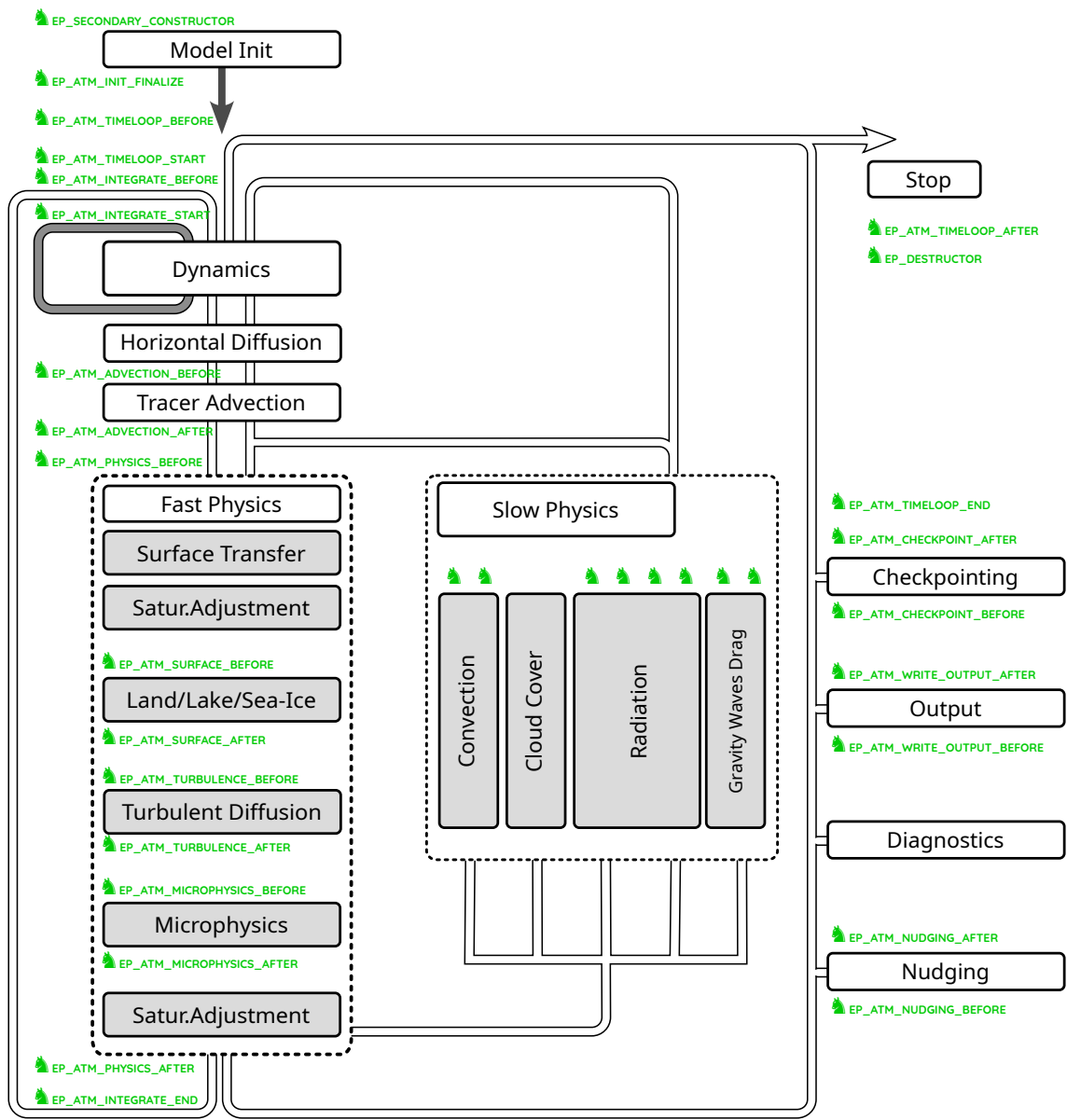


Figure 9.5.: Scheme of ICON control flow with ComIn entry points depicted in green. The location of the entry points refers to the ComIn library v0.1.

For comprehensive instructions on how to build and use these example plugins see the user guide on the project homepage.

Script name	Language	Description
simple_fortran	Fortran	Simple ComIn plugin written in the <i>Fortran</i> programming language.
calc_water_column	Fortran	Simple diagnostic application, calculating the liquid water path, the ice water path, and the total water column.
yaxt_fortran	Fortran	Example plugin to demonstrate the usage of the YAXT communication library.
simple_c	C/C++	Simple ComIn plugin written in the <i>C</i> programming language.
yac_input	C/C++	Plugin written in C which encapsulates a YAC coupling inside a ComIn plugin.
yaxt_c	C/C++	Plugin written in C to demonstrate using the YAXT communication library.
simple_python_plugin.py	Python	Simple ComIn plugin written in Python, using the ComIn Python adapter.
point_source.py	Python	Test plugin requesting a tracer that participates in ICON's turbulence and convection scheme.

9.5.3. Building Blocks of a Fortran ComIn Plugin

As explained above, ComIn plugins can be written in either Fortran 2003, C/C++ or Python. To illustrate, this section first briefly explains the building blocks of a plugin implemented in Fortran. Then the next Section 9.5.4 will focus on an example written in Python.

Each plugin regardless of the programming language consists of three building blocks:

- *Primary constructor*: This is a function which registers the plugin and appends subroutines of the 3rd party module to the callback register. The 3rd party module may also register additional variables.
- *Secondary constructor*: This function is called after the allocation of ICON variable lists and fields and before the time loop. It obtains readable and/or writable pointers to the ICON data fields. Technically the secondary constructor is also just a callback function.
- *Callback function(s)*: These are called by the ICON model during the simulation.

Basically, the primary constructor contains the following steps: Plugins are allowed to register additional model variables for ICON. A list of to-be-created variables made known to the ICON via the function `comin_var_request_add`.

```
CALL comin_var_request_add(var_descriptor, lmodexclusive)
```

The `var_descriptor` is used to describe (and uniquely identify) a model variable in ICON:

```
var_descriptor = t_comin_var_descriptor( &
    id = domain_id, name = "variable_name")
```

Whenever a plugin calls `comin_var_request_add`, there is a check to determine if the requested variable is already registered. In this case, the existing variable will be used instead of creating a new one. However, the plugin may also assert that the variable has been exclusively requested, using the `lmodexclusive` setting.

Variables may also be appended to ICON's container of tracer variables, newly created fields can be added to ICON's set of restart variables, or other meta-data may be set through the function `comin_metadata_set`:

```
CALL comin_metadata_set(var_descriptor, metadata key, value)
```

The primary constructor then appends subroutines of the 3rd party module to the callback register via the adapter library subroutine `comin_callback_register`. Entry points are denoted by named integer constants and a table of available entry points can be found in the [technical documentation](#).

```
CALL comin_callback_register(entry_point_id, fct_ptr)
```

Here, the callback function (denoted by an argument `fct_ptr`) does not have additional arguments or return values.

Another important part of the ComIn adapter library are the *descriptive data structures*. The descriptive data structures contain information on the ICON setup (e.g. Fortran KIND values), the computational grid(s), and the simulation status. All descriptive data structures are treated as read-only from the perspective of the 3rd party plugins.

- *Global data* is available for the plugins primary constructor and all subsequent subroutine callbacks. Global data is never changed or updated and invariant w.r.t. the computational grid (logical domain ID).
- *Grid information* is available for the 3rd party module's primary constructor and all subsequent subroutine callbacks. Grid information is never changed or updated. The data structures in this section are replicated for each computational domain (logical domain ID).
- *Timing information* on the simulation.

Access to ICON data fields happens via an accessor function `comin_var_get` in a secondary constructor. It registers the access to a variable and returns a variable handle.

```
CALL comin_var_get(context, var_descriptor, flag, var_pointer)
```

Basically, this returns a 5-dimensional `REAL(wp)` pointer `var_pointer`. As input arguments, the plugin constructor has to provide a `context`, i.e. the name of the entry point where the variable is used, a `var_descriptor` as described above, and, optionally, an access `flag` which provides information on the data flow, like `COMIN_FLAG_READ`, `COMIN_FLAG_WRITE`.

Once the Fortran plugin is written, the next step involves creating the shared library of your plugin. It's recommended to use `cmake` for the build process, and the detailed instructions can be found in the user guide.

9.5.4. Step-by-step Tutorial: A ComIn plugin written in Python

Python plugins can be attached to ComIn via the Python adapter which is located in the `plugins` directory in the ComIn source code. Python plugins do not need any compilation process at all: They use a pre-built adapter library `libpython_adapter.so` that is compiled when the `--enable-bundled-python=comin` flag is set in the ICON configuration.

The Python adapter embeds a Python interpreter, which also has the `comin` Python module available. This module contains all the functions, variables, constants and data structures of the Python language API. When including the Python adapter in the namelist, the Python plugin script must be specified as the `options`, which can be modified while the actual ComIn plugin Python adapter (`libpython_adapter.so`) remains unchanged. This script is executed in the primary constructor of the Python adapter. Further callbacks can then be registered by the `@comin.context` function decorator.

To write a Python plugin, start by importing the necessary environments. You may import as many packages as needed, such as `numpy`, `sys`, etc. However, it's crucial to also load the `comin` module:

```
import comin
import numpy as np
```

The next step involves registering new variables for the desired domain. Additionally, setting the essential metadata for variables is possible; for instance, it can be specified whether the new variable is a tracer. A table containing lists of all metadata that can be set is available in the technical documentation.

```
comin.var_request_add((variable_name, domain_id), lmodexclusive=False)
comin.metadata_set((variable_name, domain_id), tracer=True)
```

One can also access descriptive data structures, for instance the global data structure and one of its member (`nproma`) can be obtained as


```
global_data = comin.descrdata_get_global()
nproma = global_data.nproma
```

Given the fact that the secondary constructor is a callback itself, a function can be defined and then registered to the secondary constructor entry point. Within the secondary constructor, accessing the ICON data fields and the fields which were created in the previous step is possible.

```
@comin.EP_SECONDARY_CONSTRUCTOR
def plugin_constructor():
    global var
    var_descriptor = ("variable_name", domain_id)
    var = comin.var_get([comin.context], var_descriptor)
    comin.metadata_get(var_descriptor, metadata_key)
```

Note that the function `comin.register_callback` is used as a function decorator. Finally, the third party modules can be implemented through callback functions like the following:

```
@comin.context
def plugin_func():
    np.asarray(var)[:]= 2.0
```

where `var` is a variable handle obtained as above.

The ComIn repository contains a Python API description detailing all the available functions and descriptive data structures that can be implemented.

10. Post-Processing and Visualization

Oh my God! Look at that picture over there! There's the Earth coming up. Wow, is that pretty.

W. Anders, Apollo 8

ICON offers the possibility to produce output either in NetCDF or GRIB2 format. Many visualization environments such as GrADS, Matlab or R now include packages with which NetCDF data files can be handled. The GRIB format, which is also commonly used in meteorology, can be processed with these tools as well. However, since the standardization of unstructured GRIB records is rather new, many post-processing packages offer only limited support for GRIB data that has been stored on the triangular ICON grid.

Since the visualization of *regular* (lat-lon) grid data is relatively straightforward, we limit ourselves in our description to a very simple program, `ncview`, which does not have a large functionality but is an easy-to-use program for a quick view of NetCDF output files. It is therefore very useful for a first impression.

Model data that has been stored on the *triangular* ICON grid can be visualized with the Python programming language, the NCL scripting language or the R statistical computing language. Section 10.3 contains some examples how to visualize NetCDF data sets without the need of an additional regridding.

10.1. Retrieving Data Set Information

We begin with command-line utilities which provide a textual description of the data sets. For a quick overview of dimensions and variables, the command-line utility `ncdump` can be used. This program will shortly be described first. More sophisticated tools exist, for cutting out subsets of data, e. g., and producing averages or time series. One of these tools are the `cdo` utilities.

10.1.1. The `ncdump` Tool

The tool `ncdump` comes with the NetCDF library as provided by Unidata and generates a text representation of a NetCDF file on standard output. The text representation is in a form called CDL (network Common Data form Language). `ncdump` may be used as a simple browser for NetCDF data files, to display the dimension names and sizes, variable names, types and shapes, attribute names and values, and optionally, the data values

themselves for all or selected variables in ASCII format. For example, to investigate the structure of a NetCDF file, use

```
ncdump -h data-file.nc
```

With this command, dimension names and sizes, variable names, dependencies and values of dimensions will be displayed. To show the type version of a NetCDF file, type

```
ncdump -k data-file.nc
```

This gives information about the NetCDF base format of a specific file, for example NetCDF Classic Format, NetCDF 64-bit Offset Format or NetCDF-4 Format.

NetCDF data can also be redirected to a text file with

```
ncdump -b c data-file.nc > data-file.cdl
```

This produces an annotated CDL version of the structure and the data in the NetCDF file *data-file.nc*. You can also save data for specified variables to text files just using:

```
ncdump -v variable data-file.nc > data-file.txt
```

For further information on working with `ncdump` see

https://docs.unidata.ucar.edu/nug/current/netcdf_utilities_guide.html

10.1.2. CDO – Climate Data Operators

The Climate Data Operators (CDO) are a collection of command-line operators to manipulate and analyze NetCDF and GRIB data. The CDO package is developed and maintained at the MPI for Meteorology in Hamburg. Source code and documentation are available from [Schulzweida \(2024\)](#).

There is a possibility to get support via the forums¹ or issue tracking system of CDO. Further details are provided in the FAQ².

The tool includes more than 400 operators to print information about data sets, copy, split and merge data sets, select parts of a data set, compare data sets, modify data sets, arithmetically process data sets, to produce different kind of statistics, to detrend time series, for interpolation and spectral transformations. The CDOs can also be used to convert from GRIB to NetCDF or vice versa, although some care has to be taken there.

In particular, the "operator" `cdo infov` writes information about the structure and contents of all input files to standard output. By typing

```
cdo infov data-file.nc
```

in the command-line for each field the following elements are printed: date and time, parameter identifier and level, size of the grid and number of missing values, minimum, mean and maximum. A variant of this CDO operator is

```
cdo sinfov data-file.nc
```

which prints out short information of each field.

¹<https://code.mpimet.mpg.de/projects/cdo/boards>

²<https://code.mpimet.mpg.de/projects/cdo/wiki/FAQ>

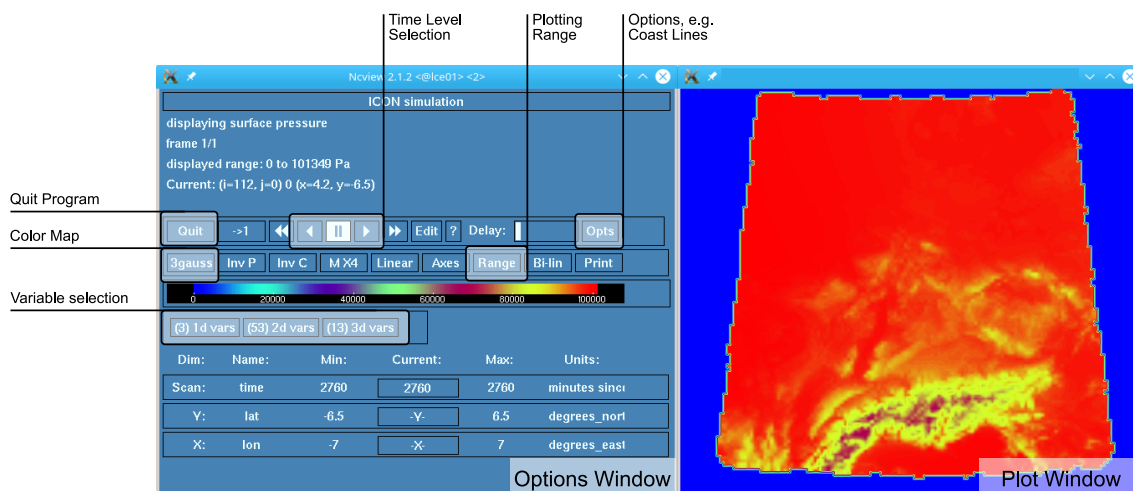


Figure 10.1.: Screenshot of the *ncview* NetCDF plotting tool.

10.2. Plotting Data Sets on Regular Grids: *ncview*

ncview is a visual browser for NetCDF format files developed by David W. Pierce. Using *ncview* you can get a quick and easy look at *regular* grid data in your NetCDF files.

To install *ncview* on your local platform, see the *ncview* website:

http://meteora.ucsd.edu/~pierce/ncview_home_page.html

You can run the program by typing:

```
ncview data-file.nc
```

which will open a new window with the display options. It is possible to view simple movies of data, view along different dimensions, to have a look at actual data values at specific coordinates, change color maps, invert data, etc., see the screenshot in Fig. 10.1.

If *data-file.nc* contains wildcards such as '*' or '?' then all files matching the description are scanned, if all of the files contain the same variables on the same grid. Choose the variable you want to view. Variables which are functions of longitude and latitude will be displayed in two-dimensional images. If there is more than one time step available you can easily view a simple movie by just pushing the forward button. The appearance of the image can be changed by varying the colors of the displayed range of the data set values or by adding/removing coastlines. Each one- or two-dimensional subset of the data can be selected for visualization. *ncview* allows the selection of the dimensions of the fields available, e.g. longitude and height instead of longitude and latitude of 3D fields.

The pictures can be sent to Postscript (*.ps) output by using the function `print`. Be careful that whenever you want to close only a single plot window to use the `close` button, because clicking on the -icon on the top right of the window will close all *ncview* windows and terminate the entire program!

10.3. Plotting Data Sets on the Triangular Grid

Let us now focus on ICON's original computational mesh, the triangular grid. In this section we will sketch several approaches for the visualization of data sets based on the triangular cells.

In recent years, numerous open source packages have appeared: We will present quick start examples for the Python programming language, NCL, and R. The different tools vary significantly in terms of their functionality, but also regarding their state of development – some software packages (NCL) have been discontinued, so no further development is taking place. All approaches have in common that the NetCDF grid file must be read in together with the data file.

10.3.1. Visualization with Python

By now there is quite a number of visualization packages available that rely on the Python ecosystem.

Here, as a introductory example, we will be using [Matplotlib](#), a plotting library for the Python programming language and its numerical mathematics extension NumPy. The [Cartopy package](#) extends the Matplotlib functionality and offers map projection definitions, and arbitrary point, line, and polygon transformations (see the [Cartopy list of projections](#) for more information).

The following minimal example displays ICON surface data. First, we load the necessary Python modules for data read-in and plotting:

```
import pathlib, numpy, netCDF4, cartopy
from matplotlib import pyplot as plt
```

Second, set the file locations. For convenience, we use a path specification relative to the user's \$HOME directory:

```
home = str(pathlib.Path.home())
grid_filename = home + "icon_DOM01.nc"
data_filename = home + "external_parameter_icon_DOM01_tiles.nc"
```

Open the NetCDF grid file and load the data sites. The data sites for the surface height field `topography_c` are the triangle circumcenters, located at `clon`, `clat`.

```
ds = netCDF4.Dataset(grid_filename)
cx = numpy.degrees(numpy.asarray(ds["clon"]))
cy = numpy.degrees(numpy.asarray(ds["clat"]))
```

Load external parameters data set `topography_c` from a second file:

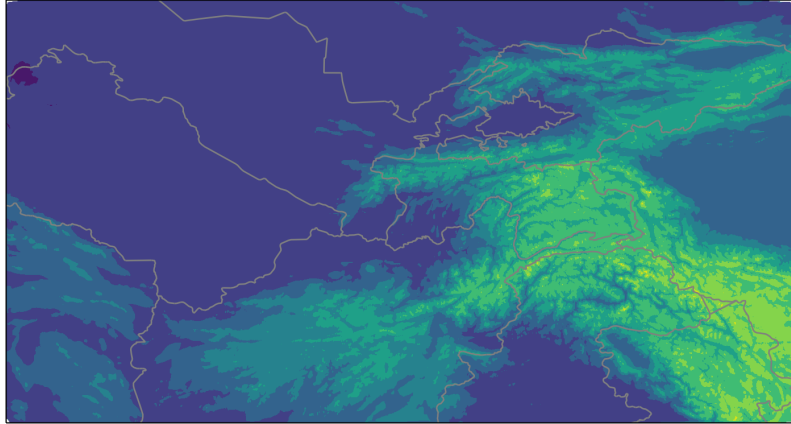


Figure 10.2.: The plot generated by the Python example script using Matplotlib and Cartopy, see Section 10.3.1.

```
ds = netCDF4.Dataset(data_filename)
src_data = numpy.asarray(ds["topography_c"])
```

Finally, we plot with the function `tricontourf`, resulting in the plot shown in Fig. 10.2:

```
fig = plt.figure(figsize=(9, 9))
ax = plt.axes(projection=cartopy.crs.PlateCarree())
ax.add_feature(cartopy.feature.BORDERS, edgecolor='gray')

ax.tricontourf(cx, cy, src_data, transform=cartopy.crs.PlateCarree())

plt.show()
fig.savefig("HSURF.png", dpi=200)
```



Psypplot: For more advanced visualizations we recommend at this point the package *psypplot*, again based on the Matplotlib package. See the website <https://psypplot.github.io> for further information including a rich collection of documented examples. Psypplot even features a viewer application for netCDF files, see [psy-view](#), that is highly motivated by the *ncview* software but works also for unstructured grids.

10.3.2. NCL – NCAR Command Language

The NCAR Command Language (NCL) is an interpreted language designed specifically for scientific data analysis and visualization.

NCL allows convenient access to data in a variety of formats such as NetCDF and GRIB1/2, among others. It has many features common to modern programming languages, such as

types, variables, operators, expressions, conditional statements, loops, and functions and procedures, see <https://www.ncl.ucar.edu> for details.

Besides an interactive mode, NCL allows for script processing (recommended). NCL scripts are processed on the command-line by typing

```
ncl filename.ncl
```

For visualizing ICON data on the native triangular grid, we recommend using NCL 6.2.0 or higher.

Pivot to Python. Note that NCAR has made the decision to adopt Python as the scripting language platform of choice for future development of analysis and visualization tools. NCAR will not add new features to the NCL language or function library although the NCL software package continues to build on currently supported platforms.

NCL Quick-Start Example

The following example script creates a temperature contour plot with NCL (see Figure 10.3):

```
begin

; Open model level output file
File = addfile( "JABW_DOM01_ML_0001.nc", "r" )

; read grid information (i.e. coordinates of cell centers and vertices)
rad2deg = 45./atan(1.)          ; radians to degrees
clon = File->clon * rad2deg      ; cell center, lon (ncells)
clat = File->clat * rad2deg      ; cell center, lat (ncells)

vlon = File->clon_bnds * rad2deg ; cell vertices, lon (ncells,3)
vlat = File->clat_bnds * rad2deg ; cell vertices, lat (ncells,3)

; read data
;
temp_ml = File->temp(:,:,:)    ; dims: (time,lev,cell)
print("max T " + max(temp_ml) )
print("min T " + min(temp_ml) )

; create plot
;
wks = gsn_open_wks("ps","outfile")
gsn_define_colormap(wks,"testcmap")          ; choose colormap

ResC                                     = True
ResC@sfXArray                           = clon      ; cell center (lon)
ResC@sfYArray                           = clat      ; cell center (lat)
ResC@sfXCellBounds                      = vlon      ; define triangulation
ResC@sfYCellBounds                      = vlat      ; define triangulation
```

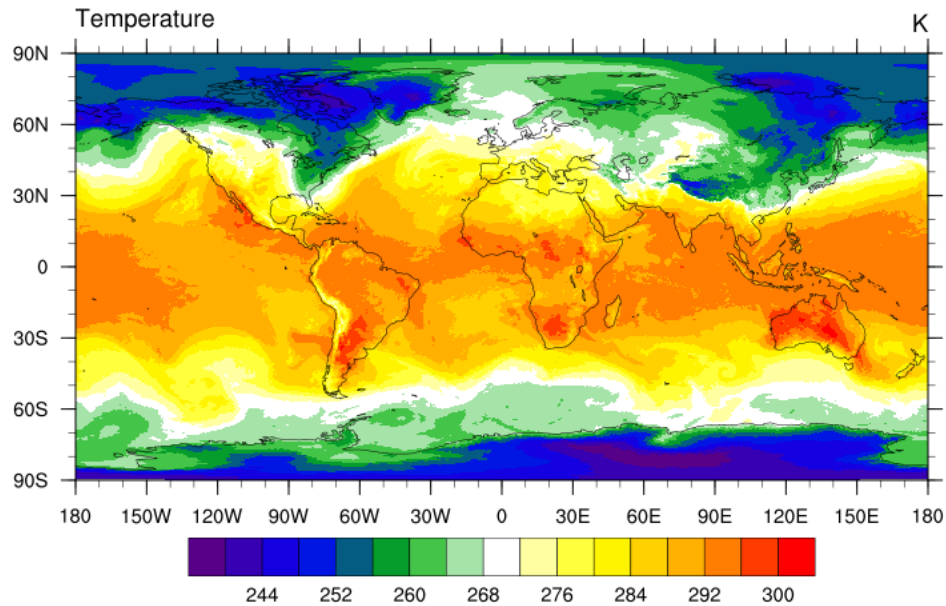



Figure 10.3.: ICON temperature field on a specific model level produced with the above NCL script.

```

ResC@cnFillOn      = True      ; do color fill
ResC@cnFillMode    = "cellfill"
ResC@cnLinesOn     = False     ; no contour lines

; plot temperature level
plot = gsn_csm_contour_map(wks,temp_ml(0,80,:),ResC)

end

```

To open a data file for reading, the function `addfile` returns a file variable reference to the specified file. Second, for drawing graphics, the function `gsn_open_wks` creates an output resource, where the “ps”, “pdf” or “png” format are available. Third, the command `gsn_csm_contour_map` creates and draws a contour plot over a map.

Loading the coordinates of the triangle cell centers into NCL (resources `sfXArray` and `sfYArray`) is essential for visualizing ICON data on the native grid. Loading the vertex coordinates of each triangle (resources `sfXCellBounds` and `sfYCellBounds`), however, is optional. If not given, a Delaunay triangulation in the 2D plane will be performed by NCL, based on the cell center information. If given, the triangles defining the mesh will be deduced by sorting and matching vertices from adjacent cell boundaries. If you are interested in the correct representation of individual cells, the resource `sf[X/Y]CellBounds` should be set.

Creating a plot can get very complex depending on how you want to look at your data. Therefore we refer to the [NCL documentation](#).

10.3.3. Visualization with R

Section authors

J. Förstner and M. Köhler,
DWD Physical Processes Division

R is a free software environment for statistical computing and graphics. R is available as free software under the terms of the Free Software Foundation's GNU General Public License. More Information can be found here:

<https://www.r-project.org>

To start R in an interactive mode simply type

R

on the command line. Afterwards R commands can be entered, which are then interpreted line by line. R can be extended (easily) via packages. Additional packages have to be installed via the R command

```
install.packages("package_name")
```

For the compilation of some packages it might be necessary to provide specific (development) libraries on the system and to give information about the include and library paths as additional arguments to that command.

Most packages are available through the CRAN family of internet sites. An exception to this is the **gribr** package, which is used in the example script below to read in GRIB2 data:

<https://github.com/nawendt/gribr>

This package uses and therefore needs a recent installation of ecCodes (it is not working with the GRIB API). Additional information about (other) prerequisites and the installation of the package can be found on the given website.



Installation of R on the DWD computer system: The R software has been preinstalled on DWD's `rc1.dwd.de` as a software module, where R/4.2.0 is the current default (for which the below example has been tested). Further modules have to be loaded as well, see the comment in the following example.

Besides an interactive mode, R allows for script processing (recommended). R scripts are processed on the command-line by typing

```
Rscript filename.R
```

As default a PDF named `Rplots.pdf` will be created.

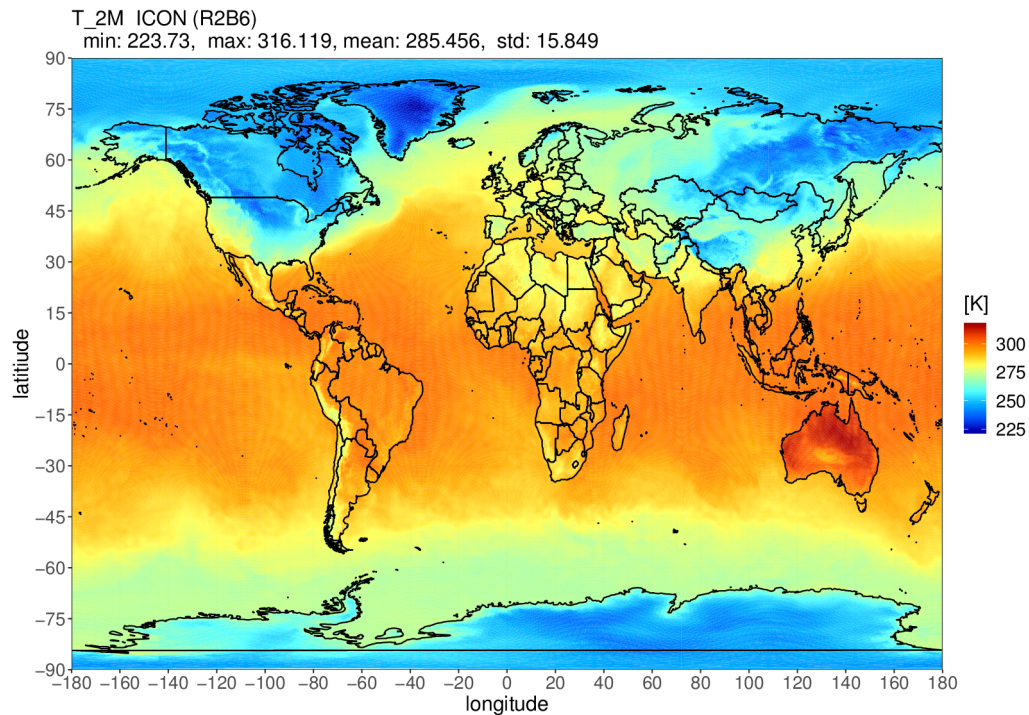


Figure 10.4.: ICON 2m temperature field produced with the given example script for R.

R Quick-Start Example

The following example script `icon_native_4_tutorial1.R` can be found in the tarball directory `scripts`. It creates a global temperature contour plot with R (see Figure 10.4):

```
# ... on DWD's RCL, as a prerequisite issue the following commands:
# module load netcdf4 oracle
# module load R
# module load gribr

# -----

# load necessary libraries
library(gribr)
library(RNetCDF)
library(data.table)
library(ggplot2)
library(dplyr)
library(colorRamps)

# --- setup -----

shortName <- "T_2M"
title     <- paste(shortName, " ICON (R2B6)\n")
```

```

# data file names
grid <- "./icon_grid_0024_R02B06_G.nc"
data <- "./T_2M.R2B06_ICON_global.grb"

# --- icon grid -----

ncHandle <- open.nc(grid)

# function to convert coordinates in degrees
rad2deg <- function(rad) {(rad * 180) / (pi)}

# get longitudes and latitudes of triangle vertices of a cell
vlon <- rad2deg(var.get.nc(ncHandle,"clon_vertices"))
vlat <- rad2deg(var.get.nc(ncHandle,"clat_vertices"))

close.nc(ncHandle)

# --- icon data -----

gribHandle <- grib_open(data)

# select the grib record based on a given list of keys
gribRecord <- grib_select(gribHandle, list(shortName = shortName))

grib_close(gribHandle)

# create a data table with data to plot - ids and values are tripled
DT <- data.table(lon = as.vector(vlon),
                 lat = as.vector(vlat),
                 id = rep(1:(dim(vlon)[2]) , each=3),
                 var = rep(gribRecord$values, each=3))

# --- domain and edges -----

# Global
xrange <- c(-180., 180.)
yrange <- c(-90., 90.)

# select the subset of ids in the domain
usedIDs <- unique(DT[lon%between%xrange & lat%between%yrange]$id)

# use only the respective subset of the data
DT <- DT[id %in% usedIDs]

# special treatment for the cells near the date line
# ... when the 3 corners on opposite sides of the date line move one corner
IDsR = DT[,list( (max(lon)-min(lon))>200 & mean(lon)>0.0 ), by=id][V1==T]$id
IDsL = DT[,list( (max(lon)-min(lon))>200 & mean(lon)<0.0 ), by=id][V1==T]$id
DT[id %in% IDsR & lon < 0.0, lon := lon + 360.0]
DT[id %in% IDsL & lon > 0.0, lon := lon - 360.0]
# ... copy triangles by 360deg to fill holes near date line
DTT <- DT[id%in%IDsR]
DTT[lon > 0.0, lon := lon - 360.0]

```

```

DTT$id <- DTT$id + max(DT$id)
DT      <- rbind(DT, DTT)
DTT     <- DT[id%in%IDsL]
DTT[lon < 0.0, lon := lon + 360.0]
DTT$id <- DTT$id + max(DT$id)
DT      <- rbind(DT, DTT)

# --- plot data using ggplot2 with geom_polygon -----

# landscape mode
pdf(paper="a4r", width=11.692, height=8.267)

# create the plot object
pp <- ggplot() +
  geom_polygon(data = DT, aes(x = lon, y = lat, group = id, fill = var)) +
  borders(colour = "black", xlim = xrange, ylim = yrange) +
  scale_fill_gradientn(colours = matlab.like(100)) +
  coord_cartesian(xlim = xrange, ylim = yrange, expand = FALSE) +
  scale_x_continuous(breaks = seq(xrange[1], xrange[2], 20)) +
  scale_y_continuous(breaks = seq(yrange[1], yrange[2], 15)) +
  theme_bw() +
  labs(x="longitude", y="latitude", fill="[K]") +
  theme(axis.text    = element_text(size=14),
        axis.title   = element_text(size=16),
        plot.title   = element_text(size=16, hjust=0),
        legend.title  = element_text(size=16),
        legend.text   = element_text(size=14)) +
  ggtitle(paste0(title,
    " min: ", round(min (DT$var, na.rm=TRUE),3), ", ",
    " max: ", round(max (DT$var, na.rm=TRUE),3), ", ",
    "mean: ", round(mean(DT$var, na.rm=TRUE),3), ", ",
    " std: ", round(sd (DT$var, na.rm=TRUE),3)))

# issue the plot object
pp

```

10.4. Post-Processing of Data Sets

10.4.1. Post-Processing using the CDO

The Climate Data Operators (CDO) have already been introduced in the previous Section 10.1.2 for retrieving information on data files. Additionally, the CDO are also capable of performing a horizontal remapping to regular grids or a vertical remapping from model levels to pressure levels.

Caveat: Alas, the CDO do not handle the entire set of GRIB2 meta-data correctly. Some meta-data items, for example those which regard ensemble runs, still remain unsupported. However, this limitation does not matter if the desired output format is NetCDF and not GRIB2, or if the processed data fields are rather standard. Alternatives for remapping ICON data sets to lon-lat grids are the *fieldextra* software (Section 10.4.2) or the DWD ICON Tools, which are internally used at DWD but lack official support.

Horizontal Remapping

Basically, two steps are necessary to process the ICON output files with CDO:

- In the first call of the CDO ("gencon"), 1st order conservative remap weights are generated and stored to a separate file:

```
cdo gencon, lat-lon_grid_description icon_grid coefficient_file
```

The generation of the interpolation weights may take some time.

- Afterwards, the interpolation is done with

```
cdo remap, grid_description, coefficient_file in_file out_file
```

Here, the file *icon_grid* is a NetCDF file which contains the topological and geometric information about the triangular ICON grid. This grid information must correspond to the data set *in_file* and it is not part of the data set itself. Instead, it must be downloaded separately from the web. See http://icon-downloads.mpimet.mpg.de/dwd_grids.xml for the list of "official" DWD ICON grids³.

With respect to data sets with “missing values” the above CDO workflow with pre-calculated weights fails. The following remark applies, for example, to the ICON-D2 data sets:

Since the “missing values” are not part of the source grid definition, the CDOs detect the masking only when reading the data file. Then, to avoid the risk of using undefined values for the interpolation stencils, the whole process is aborted in this case. A workaround when interpolating a single file is to do the weight calculation simultaneously with the interpolation:

```
cdo remapcon, grid_description -setgrid, icon_grid:2 in_file out_file
```

³Currently in operational use is the global grid #26 with 13km horizontal grid spacing (http://icon-downloads.mpimet.mpg.de/grids/public/edzw/icon_grid_0026_R03B07_G.nc).

In this example we have also set the source grid index explicitly to 2 to distinguish between the cell grid, edge grid, and the vertex grid in the `icon_grid` file. The concrete index can be obtained from the output of `cdo sinfov data-file.nc`.



DWD OpenData Documentation: A valuable web resource in this context is DWD's OpenData website. Here, ICON data sets are made freely available to the public and the whole procedure of remapping ICON data with the CDO tool is also described.

OpenData information on ICON forecast data:

<https://www.dwd.de/DE/leistungen/opendata/hilfe.html>

Furthermore, there even exists a Docker software container in which all necessary packages for the CDO and the ecCodes are combined:

<https://github.com/deutscherwetterdienst/regrid>

The structure of the lon-lat grid description file is explained in [Appendix D](#) of the CDO documentation ([Schulzweida, 2024](#)). A global regular grid, for example, would be defined as

```
gridtype = lonlat
xsize    = 1440
ysize    = 721
xfirst   = 0.00
xinc     = 0.25
yfirst   = -90.00
yinc     = 0.25
```

Vertical Interpolation

Although ICON provides the functionality to write variables on pressure or isentropic levels (Section 7.1.1), post-processing of model level output might be necessary for example when retrieving data from an archive. For this purpose, the `ap2plx` function of CDO can be used.

CDO expects `air_pressure` as the NetCDF standard name of the pressure field. This is usually not the case for ICON data. Hence, as a first step, a renaming of the pressure field standard name is necessary. Assuming the the pressure field is named `pres` and the file containing the pressure field is named `pres.nc`:

```
ncatted -O -a standard_name,pres,o,c,"air_pressure" pres.nc pres.nc
```

Please note that `ncatted` is provided by the NCO library⁴. With `air_pressure` as the pressure standard name, CDO can be used the remap model level variables from a file `vars.nc` to pressure levels 500 hPa and 825 hPa:

```
cdo ap2plx,50000,82500 -merge pres.nc vars.nc vars_pres.nc
```

⁴<https://nco.sourceforge.net/>

10.4.2. Post-Processing using Fieldextra

Section authors

P. Baumann and J.-M. Bettems,
MeteoSwiss

Fieldextra is an official post-processing software of the COSMO Consortium. It is maintained and developed at MeteoSwiss.

Fieldextra is a generic tool to manipulate NWP model data and gridded observations. It offers best compatibility with ICON(-ART), COSMO(-ART), and IFS models. Besides support of regular grids, the program supports both GRIB2 and NetCDF data that has been stored on the triangular ICON grid, and provides transparent access to the DWD ICON tools interpolation methods (see Section 1.4).

The software is implemented in Fortran 2008 as a single but modular code. A control file composed of Fortran namelists defines the set of operations to apply on the input data. Simple data processing and more complex data operations are supported. Fieldextra is designed as a toolbox; a set of primitive operations is provided, which can be freely combined and iterated.

As examples of post-processing tasks, we mention

- complex transformations from one grid to another, including the vertical dimension, for example the interpolation of pollen boundaries from ICON-ART to a grid at higher horizontal and vertical resolution,
- computation of the height of the tropopause, based on complex conditions on meteorological fields, including gradients,
- computation of ensemble-based products, like probabilities, quantiles, standard deviation,
- format transformations between GRIB1, GRIB2, and NetCDF (including EXTPAR output).

But fieldextra is also useful to solve pre-processing tasks like

- remapping gridded Radar observations from a regular grid to the ICON triangular grid to feed the latent-heat-nudging process,
- merging the sea surface temperature from the IFS into the ICON analysis at sea points,
- generating initial conditions on the ICON unstructured grid from initial conditions calculated on a regular grid, like COSMO type data,
- interpolating IFS boundary conditions from the regular grid to the ICON unstructured grid.

Since the primary focus of the program is the production environment, a lot of effort has been put into

- robustness of the code,
- extensive reporting of exceptions,
- careful processing of field meta-information, with systematic checks to avoid meaningless products,
- IO, memory, and CPU optimization (OpenMP parallelism), and
- comprehensive diagnostic and profiling.

The program is used, in particular, for production at MeteoSwiss, at DWD and other centers, and for COSMO-LEPS at ECMWF.

An overview document, a primer, release notes, and other information are publicly available at the dedicated fieldextra GitHub wiki at <https://github.com/COSMO-ORG/fieldextra-wiki/wiki>.

Full installations of the latest releases of fieldextra are accessible for the UNIX group cfextra on the Atos HPCF at ECMWF in `/ec/res4/hpcperm/chcosmo/projects/fieldextra`.

If you want to install fieldextra on your own platform, you can either retrieve the code from the private fieldextra repository on GitHub⁵, or download a self-contained package of the latest major production release from the COSMO web page at <http://www.cosmo-model.org/content/support/software/default.htm#fieldextra>.

In order to access the GitHub repository, you need to have access to the GitHub platform. For that, you first have to create a GitHub account, which is free of charges, and then ask for access to the fieldextra repository on GitHub via the mailing list fieldextra@cosmo-model.org.

The code is portable. It uses only standard Fortran features and should work on any UNIX / Linux platform. Once the code is installed, all resources files used at run time are also provided. Furthermore, comprehensive documentation, including an introductory tutorial, and many commented examples are available. A set of command line tools, based on fieldextra, is also part of the installation.

Fieldextra is subject to licensing. Its usage is free to all COSMO members. Community support is available via the mailing list fieldextra@cosmo-model.org. Licences are free for the R&D community, but without support.

For a first introduction to fieldextra, take a look at the overview and the first contact documents, which are available at the fieldextra GitHub wiki⁶. The first contact document in particular provides some insight into the internal processing of the software, which helps to better understand the namelists. As a next step, the commented examples in the cookbook folder of the code repository provide an overview of the large spectrum of possible applications (see `cookbook/README.cookbook`), and are a good start to develop your own namelists. A comprehensive description of the program functionalities is given in the file `documentation/README.user`. Also available is a FAQ document in `documentation/FAQ`; consult it before looking for support!

⁵<https://github.com/COSMO-ORG/fieldextra>, restricted access

⁶<https://github.com/COSMO-ORG/fieldextra-wiki/wiki>

11. ICON's Data Assimilation System and Analysis Products

In this chapter you will get to know basic components of the ICON data assimilation system. It consists of a whole collection of programs and modules both for the atmospheric variables of the model as well as for soil, snow, ice and sea surface, all collected into the *Data Assimilation Coding Environment* (DACE). The analysis products of this software package are discussed in Section 11.4.

11.1. Data Assimilation

Numerical weather prediction (NWP) is an initial value problem. The ability to make a skillful forecast heavily depends on an accurate estimate of the present atmospheric state, known as *analysis*. In general, an analysis is generated by combining, in an optimal way, all available observations with a short term forecast of a general circulation model (e.g. ICON).

Stated in a more abstract way, the basic idea of data assimilation is to fit model states x to observations y . Usually, we do not observe model quantities directly or not at the model grid points. Here, we work with *observation operators* H which take a model state and calculate a simulated observation $y = H(x)$. In terms of software, these model operators can be seen as particular modules, which operate on the ICON model states. Their output is usually written into so-called feedback files, which contain both the real observation y_{meas} with all its meta data (descriptions, positioning, further information) as well as the simulated observation $y = H(x)$.

However, data assimilation cannot be treated at one point in time only. The information passed on from the past is a crucial ingredient for any data assimilation scheme. Thus, *cycling* is an important part of data assimilation. It means that we

1. Carry out the core data assimilation component 3D-VAR to calculate the so-called *analysis* $x^{(a)}$, i.e. a state which best fits previous information and the observations y ,
2. Propagate the analysis $x_k^{(a)}$ to the next analysis time t_{k+1} . Here, it is called *first guess* or *background* $x_{k+1}^{(b)}$.
3. Carry out the next analysis by running the core data assimilation component, generating $x_{k+1}^{(a)}$, then cycling the steps.

See Figure 11.1 for a schematic of the basic assimilation process.

ICON Basic Cycling Environment

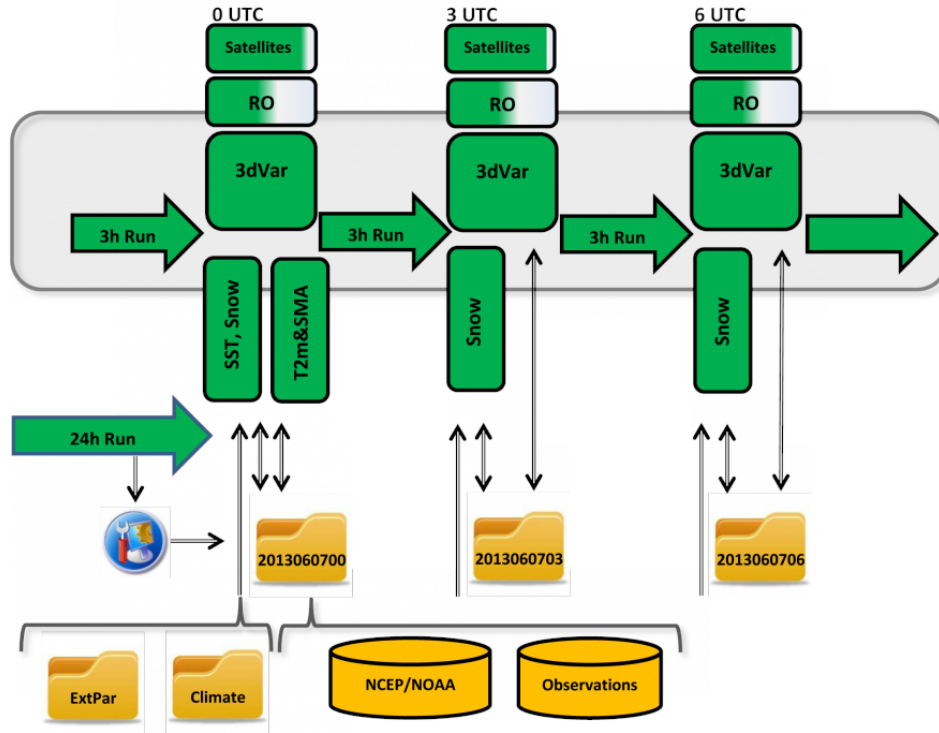


Figure 11.1.: Basic ICON cycling environment using 3D-VAR. Observations are merged with a background field taken from a 3 h forecast (first guess) of the ICON model. *Courtesy of R. Potthast, DWD.*

11.1.1. Variational Data Assimilation

The basic 3D-VAR step minimizes the functional

$$\mu(x) := \|x - x^{(b)}\|_{B^{-1}}^2 + \|y - H(x)\|_{R^{-1}}^2, \quad (11.1)$$

where B is the background state distribution *covariance matrix* which is making sure that the information which is available at some place is distributed into its neighborhood properly, and R is the error covariance matrix describing the error distribution for the observations. The minimizer of (11.1) is given by

$$x^{(a)} = x^{(b)} + BH^T(R + HBH^T)^{-1}(y - H(x^{(b)})). \quad (11.2)$$

The *background* or *first guess* $x^{(b)}$ is calculated from earlier analysis by propagating the model from a state x_{k-1} at a previous analysis time t_{k-1} to the current analysis time t_k . In the data assimilation code, the minimization of (11.1) is not carried out explicitly by (11.2), but by a conjugate gradient minimization scheme, i.e. in an iterative manner, first solving the equation

$$(R + HBH^T)z_k = y - H(x_k^{(b)})$$

in observation space calculating z_k at time t_k , then projecting the solution back into model space by

$$\delta x_k = x_k^{(a)} - x_k^{(b)} = BH^T z_k.$$

We call δx_k the *analysis increment*.

The background covariance matrix B is calculated from previous model runs by statistical methods. We employ the so-called NMC method initially developed by the US weather bureau. The matrix B thus contains statistical information about the relationship between different variables of the model, which is used in each of the assimilation steps.

11.1.2. Ensemble Kalman Filter

To obtain a better distribution of the information given by observations, modern data assimilation algorithms employ a dynamical estimator for the covariance matrix (B -matrix). Given an ensemble of states $x^{(1)}, \dots, x^{(L)}$, the standard stochastic covariance estimator calculates an estimate for the B -matrix by

$$B = \frac{1}{L-1} \sum_{\ell=1}^L (x_k^{(\ell)} - \bar{x}_k)(x_k^{(\ell)} - \bar{x}_k)^T, \quad (11.3)$$

where \bar{x} denotes the mean defined by

$$\bar{x}_k = \frac{1}{L} \sum_{\ell=1}^L x_k^{(\ell)}, \quad k \in \mathbb{N}.$$

This is leading us to the *Ensemble Kalman Filter* (EnKF), where an ensemble is employed for data assimilation and the covariance is estimated by (11.3). Here, we use the name EnKF (ensemble Kalman filter) as a generic name for all methods based on the above idea.

In principle, the EnKF carries out cycling as introduced above, just that the propagation step carries out propagation of a whole *ensemble* of L atmospheric states $x_k^{(a,\ell)}$ from time t_k to time t_{k+1} , and the analysis step has to generate L new analysis members, called the *analysis ensemble* based on the *first guess* or *background ensemble* $x^{(b,\ell)}$, $\ell = 1, \dots, L$.

Usually, the analysis is carried out in observation space, where a transformation is carried out. Also, working with a low number of ensemble members as it is necessary for large-scale data assimilation problems, we need to suppress spurious correlations which arise from a naive application of Eq. (11.3). This technique is known as *localization*, and the combined transform and localization method is called *localized ensemble transform Kalman filter* (LETKF), first suggested by Hunt et al. (2007).

The DWD data assimilation coding environment (DACE) provides a state-of-the-art implementation of the LETKF which is equipped with several important ingredients such as different types of covariance *inflation*. These are needed to properly take care of the *modeling error*. The original Kalman filter itself does not know what error the model has and thus by default under-estimates this error, which is counter-acted by a collection of tools.

11.1.3. Hybrid Data Assimilation

The combination of variational and ensemble methods provides many possibilities to further improve the state estimation of data assimilation. Based on the ensemble Kalman

filter LETKF the data assimilation coding environment provides a *hybrid system EnVar*, the *ensemble variational* data assimilation.

The basic idea of EnVar is to use the dynamical flow dependent ensemble covariance matrix B as a part of the three-dimensional variational assimilation. Here, localization is a crucial issue, since in the LETKF we localize in observation space, but 3D-VAR employs B in state space. Localization is carried out by a *diffusion*-type approximation in DACE.

The cycling for the EnVar needs to cycle both the ensemble $x^{(\ell)}$, $\ell = 1, \dots, L$ and one deterministic state x_{det} . The resolution of the ensemble can be lower than the full deterministic resolution. By default we currently employ a 40 km grid spacing for the ensemble and a 13 km global grid spacing for the deterministic state. The ensemble B matrix is then carried over to the finer deterministic resolution by interpolation. See Section 11.2 for more details on the operational assimilation system at DWD.

11.1.4. Surface Analysis

DACE provides additional modules for Sea Surface Temperature (SST) analysis, Soil Moisture Analysis (SMA) and snow analysis. Characteristic time scales of surface and soil processes are typically larger than those of atmospheric processes. Therefore, it is often sufficient to carry out surface analysis only every 6 to 24 hours.

11.2. Assimilation Cycle at DWD

The *assimilation cycle* iterates the steps described in Section 11.1: updating a short-range ICON forecast (first guess) using the observations available for that time window to generate an analysis, from which then a new updated first guess is started.

The core assimilation for atmospheric fields is based on a hybrid system (EnVar) as described in Section 11.1.3. At every assimilation step (every 3 h) an LETKF is ran using an ensemble of ICON first guesses. Currently, the ensemble consists of 40 members with a horizontal grid spacing of 40 km and a 20 km nest over Europe. A convex linear combination of the 3D-VAR climatological and the LETKF's (flow dependent) covariance matrix is then used to run a deterministic 3D-VAR analysis at 13 km horizontal grid spacing.

In addition, the above mentioned surface modules are run: Sea Surface Temperature (SST) analysis, Soil Moisture Analysis (SMA) and snow analysis.

Note that for the ICON-EU nest no assimilation of atmospheric fields is conducted. Instead the necessary atmospheric analysis increments are interpolated from the underlying global grid. Together with the available first guess fields on the nest they form the nest analysis. A separate surface analysis, however, is conducted.

The deterministic as well as the ensemble analysis is generated 8 times a day. Based on the former, deterministic forecasts are launched at approx. 13 km horizontal grid spacing globally with a 6.5 km nest over Europe. The maximum forecast time of the whole system is limited to +30 h lead time at 03/09/15/21 UTC. Otherwise, the system is integrated up

to +120 h while at 00/12 UTC the integration on the global domain (only) is prolonged to +180 h lead time.

Since the beginning of 2018 ICON ensemble forecasts are conducted as well. On the basis of the analysis ensemble 40 short to medium forecasts at 40 km globally and 20 km nested over Europe are run 8 times a day. The maximum forecast times are equivalent to those of the deterministic system (see above). The primary purpose of the ensemble forecasts is to estimate the forecast uncertainty, which arises due to uncertainties in the initial conditions and the model error.

The input, output and processes involved in the assimilation cycle are briefly described below:

Atmospheric Analysis

Fields modified by the atmospheric analysis: (see Appendix A for a description of each variable) t , p , u , v , qv .

Grid(s) on which it is performed: global

Carried out at every assimilation time step (3 h) using the data assimilation algorithms described in the previous sections.

Main input: First guess, observations, previous analysis error, online bias correction files.

Main output: Analysis of the atmospheric fields, analysis error, bias correction files, feedback files with information on the observation, its departures to first guess and analysis.

The system can make use of the following observations: radiosondes, weather stations, buoys, aircraft, ships, radio occultations, AMV winds and radiances. Available general features of the module are variational quality control and (variational) online bias correction. Regarding EnKF specifics, different types of inflation techniques, relaxation to prior perturbations and spread, adaptive localization, SST perturbations and SMA perturbations are available.

Snow Analysis

Fields modified by the snow analysis: (see Appendix A for a description of each variable) $freshsnow$, h_snow , rho_snow , t_snow , w_i , w_snow .

Grid(s) on which it is performed: global, EU-nest

Carried out at each assimilation time step (3 h).

Main input: SYNOP snow depth observations if the coverage is sufficient. If this is not the case, more sources of information are looked for until the number of observations is high enough, namely (and in this order), precipitation and 2 m temperature, direct observations (wwreports) and the NCEP external snow analysis.

Main output: Analysis of the snow fields.

Sea Surface Temperature Analysis

Fields modified by the SST analysis: (see Appendix A for a description of each variable) `fr_seaice`, `h_ice`, `t_ice`, `t_so`.

Grid(s) on which it is performed: global, EU-nest

Carried out only once a day, at 0 UTC.

Main input: NCEP analysis from the previous day (which uses satellite, buoy and ship observations, to be used as a first guess), ship and buoy observations available since the time of the NCEP analysis.

Main output: Sea surface temperature analysis and estimated error.

Soil Moisture Analysis

Fields modified by the SMA analysis: (see Appendix A for a description of each variable) `w_so`.

Grid(s) on which it is performed: global, EU-nest

Carried out only once a day, at 0 UTC.

Main input: Background fields for relevant fields at every hour since last assimilation, 2m-temperature analysis (see below) to be used as observations.

Main output: Soil moisture analysis and estimated error.

2m Temperature Analysis

Although carried out only at 0 UTC, it is run for several time steps in between to provide the output (2 m temperature) needed by the SMA analysis. Uses observations from SYNOP stations on land and METAR information from airports.

11.3. Incremental Analysis Update

Analysis fields are not perfectly balanced, which can lead to the generation of spurious sound and gravity waves in forecast runs. Thus, atmospheric models generally require some filtering procedure during model initialization, which prevents the accumulation of noise in the assimilation cycle and minimizes spin-up effects in forecast runs. In ICON a method known as *Incremental Analysis Update* (IAU) is applied, which was introduced by Bloom et al. (1996) and has become a standard method at operational NWP centers for controlling high-frequency noise originating from dynamic imbalances in the analysis state (Polavarapu et al., 2004, Buehner et al., 2015, Takacs et al., 2016, Ha et al., 2024).

The basic idea of the IAU is to add the analysis increment gradually over some time interval $\Delta\tau \gg \Delta t$, rather than adding the full increment at one particular point on time

(usually at model start). This method of tentatively pulling the model from its current state (first guess) towards the analyzed state acts as a low pass filter in the frequency domain, such that small scale non-balanced modes are effectively filtered (Bloom et al., 1996). The filter modifies only the analysis increment and not the background state, such that regions where the increment is zero remain unaffected.

Mathematically speaking, during forward integration the model is forced with appropriately weighted analysis increments:

$$\frac{d\mathbf{x}}{dt} = \mathcal{A}\mathbf{x} + g(t)\Delta\mathbf{x}^A \quad , \text{ with } \int g(t) dt = 1$$

\mathbf{x} denotes the discrete model state vector, \mathcal{A} is a matrix representing the (non)-linear dynamics of the system, $g(t)$ is a weighting (or filter) function which is non-zero over some time interval $\Delta\tau$, and $\Delta\mathbf{x}^A = \mathbf{x}^A - \mathbf{x}^{FG}$ is the analysis increment (i.e. the difference between the analysis \mathbf{x}^A and the model first guess \mathbf{x}^{FG}). In ICON, a top-hat filter of the form

$$g(t) = \begin{cases} 1/\Delta\tau & \text{for } -\Delta\tau/2 \leq t \leq \Delta\tau/2 \\ 0.0 & \text{otherwise,} \end{cases}$$

turned out to provide the most effective filtering. Note that by default the time window at which the filtering is active is symmetric w.r.t. to the nominal model start time $t_0 = 0$ s, implicating that the IAU method requires the model to be started $\Delta\tau/2$ ahead of the nominal start time. The filter width $\Delta\tau$ as well as the time interval Δt_{shift} by which the model start is shifted ahead of t_0 can be adjusted with the namelist parameters `dt_iau` and `dt_shift` < 0 (namelist `initicon_nml`), respectively. Typical values of the filter width are $\Delta\tau = 3$ h for global ICON forecasts, and $\Delta\tau = 10$ min for ICON-D2.

The ICON data assimilation system DACE provides atmospheric analysis increments for the variables u , v , T , p , and q_k with $k \in [v, c, i, r, s, g, h]$. Before these increments can be applied to the model, they must be transformed into increments of ICON's prognostic variables v_n , ρ , $\rho\theta_v$ (or π), and ρq_v . The variable transformation is performed inside the ICON model as part of the IAU process. It is described in Section 11.3.1.

Apart from atmospheric analysis increments, DACE also provides increments for several surface and soil variables, such as `FRESHSNW`, `H_SNOW`, and `W_SO`. As the potential for these variables to generate spurious atmospheric noise is relatively small, they are not treated with the IAU method, but added directly to the model first guess at $t = t_0 + \Delta t_{\text{shift}}$.

11.3.1. Variable Transformation

The atmospheric analysis increments provided by DACE (δu , δv , δT , δp , and δq_v) are transformed into increments of ICON's prognostic variables δv_n , $\delta \rho$, $\delta \pi$, and $\delta(\rho q_k)$ as follows:

Density increment $\delta\rho$ With the equation of state in the form $p\rho^{-1} = R_d T_v$, the density increment $\delta\rho$ can be written as a function of the known increments δp , δT , and $\delta\alpha = (R_v/R_d - 1)\delta q_v$:

$$\begin{aligned}\delta\rho &= \frac{1}{R_d T_v} \delta p - \frac{p}{R_d T_v^2} \delta T_v \\ &= \frac{1}{R_d T_v} \delta p - \frac{p}{R_d T_v^2} (1 + \alpha) \delta T - \frac{p T}{R_d T_v^2} \delta\alpha \\ &= \frac{1}{R_d T_v} \delta p - \frac{p}{R_d T_v T} \delta T - \frac{p}{R_d T_v (1 + \alpha)} \delta\alpha \\ &= \frac{\rho}{p} \delta p - \frac{\rho}{T} \delta T - \frac{\rho}{(1 + \alpha)} \delta\alpha\end{aligned}\tag{11.4}$$

Exner increment $\delta\pi$ Starting from the definition of the $\rho\theta_v$ increment

$$\delta(\rho\theta_v) = \theta_v \delta\rho + \rho \delta\theta_v$$

and inserting (11.4) as well as the virtual potential temperature increment derived from (3.11)

$$\begin{aligned}\delta\theta_v &= \frac{1}{\pi} \delta T_v + T_v \delta \left(\frac{1}{\pi} \right) \\ &= \frac{1}{\pi} \delta T_v - \frac{T_v}{\pi^2} \delta\pi \\ &= \frac{1}{\pi} \delta T_v - \frac{T_v}{\pi^2} \frac{\kappa}{p_{00}} \pi \left(\frac{p_{00}}{p} \right) \\ &= \frac{1 + \alpha}{\pi} \delta T + \frac{T}{\pi} \delta\alpha - \kappa \frac{\theta_v}{p} \delta p,\end{aligned}$$

it follows

$$\delta(\rho\theta_v) = \frac{\theta_v \rho}{p} \delta p - \frac{\theta_v \rho}{T} \delta T - \frac{\theta_v \rho}{(1 + \alpha)} \delta\alpha + \frac{\rho(1 + \alpha)}{\pi} \delta T + \frac{\rho T}{\pi} \delta\alpha - \kappa \frac{\rho\theta_v}{p} \delta p.\tag{11.5}$$

Recalling that $\theta_v = T_v/\pi$, terms 2 to 5 in (11.5) cancel, which results in

$$\delta(\rho\theta_v) = (1 - \kappa) \frac{\rho\theta_v}{p} \delta p.$$

The increment $\delta(\rho\theta_v)$ can finally be transformed into an exner pressure increment with the help of the equation of state (3.10).

$$\begin{aligned}\delta\pi &= \frac{R_d}{c_{vd}} \left(\frac{R_d}{p_{00}} \rho\theta_v \right)^{\frac{R_d}{c_{vd}} - 1} \frac{R_d}{p_{00}} \delta(\rho\theta_v) \\ &= \frac{R_d}{c_{vd}} \frac{\pi}{\rho\theta_v} \delta(\rho\theta_v) \\ &= \frac{R_d}{c_{vd}} (1 - \kappa) \frac{\pi}{p} \delta p \\ &= \frac{R_d}{c_{vd}} \left(\frac{c_{pd} - c_{pd} + c_{vd}}{c_{pd}} \right) \frac{\pi}{p} \delta p \\ &= \kappa \frac{\pi}{p} \delta p\end{aligned}\tag{11.6}$$

Water vapor mass fraction increment $\delta(\rho q_v)$ The increment of water vapor mass fraction $\delta(\rho q_v)$ is computed from

$$\delta(\rho q_v) = \rho \delta q_v + q_v \delta \rho . \quad (11.7)$$

If available, increments for condensates $\delta(\rho q_k)$, with $k \in [c, i, r, s, g, h]$ are computed accordingly.

Normal velocity increment δv_n At cell centers, the increments δu and δv are projected into the direction of v_n for each edge, and then linearly interpolated to the edge midpoint. In addition, the resulting edge-normal velocity increments δv_n are filtered by applying the ∇^2 operator in horizontal directions, followed by 2D divergence damping.



Current approximations: A rigorous implementation of the variable transformation (11.4), (11.6), and (11.7) requires that the analysis increments δp , δT , δq_k as well as the background state variables ρ , T , p , q_k (i.e. the first guess) are available at the desired analysis date.

The variable transformation is performed by ICON rather than DACE, the consequence being that the background state at the analysis date is not available. Please recall that the background state which is read by ICON is shifted ahead of the analysis date by Δt_{shift} . Notwithstanding this time error the time-shifted background state is used for the variable transformation.

As an additional approximation, the term $q_k \delta \rho$ in (11.7) is neglected when computing the mass fraction increments for all water constituents k .

11.3.2. Technical Hints

Technically, the application of the IAU method has some potential pitfalls, which the user should be aware of. In the following, let us assume that we want to start a global model forecast at 00 UTC.

- The validity time of the analysis file must be 00 UTC.
- The analysis file has to contain analysis increments (i.e. deviations from the first guess) rather than full fields. The only exceptions are `FR_ICE` and `T_SEA` (or alternatively `T_S0(0)`), which must be full fields (see Table 11.1).
- The model must be started from a first guess which is shifted ahead of the nominal start date by 1.5 h w.r.t. the analysis. Thus, in the given example, the validity time of the first guess must be 22:30 UTC of the previous day. This is because “dribbling” of the analysis increments is performed over the symmetric 3 h time window $[00 \text{ UTC} - 1.5 \text{ h}, 00 \text{ UTC} + 1.5 \text{ h}]$. See Figure 11.2 for an illustration of this process.

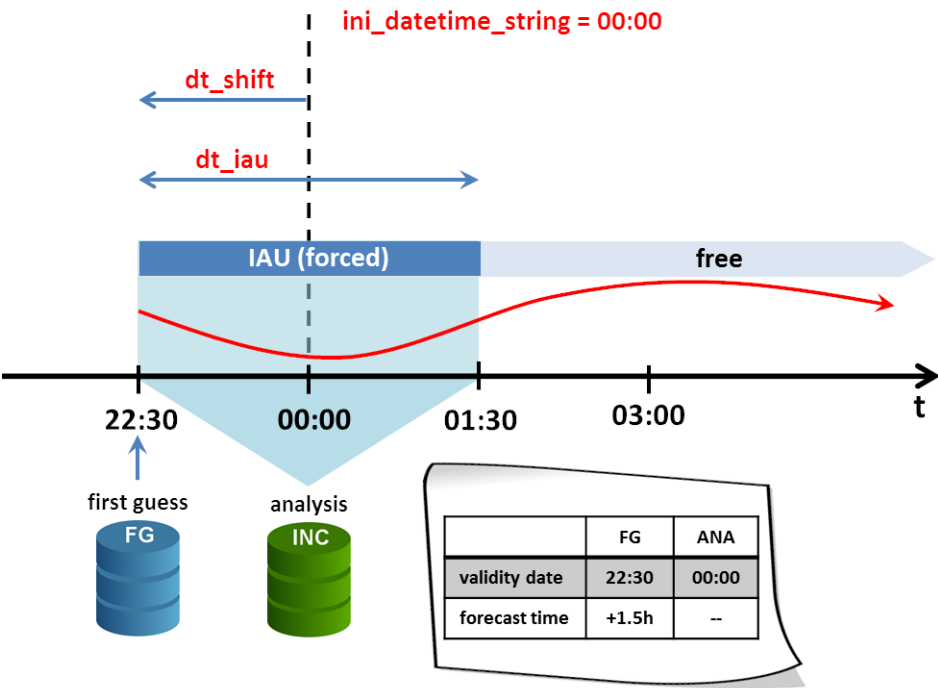


Figure 11.2.: Schematic illustrating typical settings for a global ICON forecast run starting from a DWD analysis **with IAU** at 00 UTC. IAU (i.e. analysis filtering by dribbling of analysis increments) is performed over a 3 h time interval (dt_iau), with the model start being shifted ahead of the nominal start date by 1.5 h (dt_shift). The validity date of the first guess and analysis is 22:30 UTC and 00 UTC, respectively.



The secret behind iterative IAU:

Some of you might have heard of an ICON feature named *iterative IAU*, though still wondering what's behind it. The *iterative IAU*, enabled by the namelist switch `iterate_iau = .TRUE.` (namelist `initicon_nml`), combines two model runs that serve two different purposes into a single model run. For convenience only, this is accomplished using an internal ICON loop structure.

We will explain the purpose of the two model runs in the context of a global model setup, which by default uses a **symmetric** IAU window of width $\Delta\tau = 3\text{ h}$, with the model start being shifted ahead of the nominal model start date at t_0 by $\Delta t_{\text{shift}} = -1.5\text{ h}$ (see also Figure 11.2):

The first model run is meant to generate the initialized (or filtered) analysis product out of the uninitialized analysis for IAU product. To this end an IAU run is launched which, in contrast to the standard IAU settings described above, uses a halved IAU window of width $\Delta\tau = 1.5\text{ h}$ and time shift $\Delta t_{\text{shift}} = -1.5\text{ h}$ which is, hence, **asymmetric** w.r.t. to the nominal model start date. The model integration stops after 1.5 h (at the nominal start date) and the model state is written to disk. For the example in Figure 11.2 the stop

date would be at 00 UTC. During the asymmetric IAU window, the analysis increments are fully incorporated. The resulting model state is equivalent to the *initialized analysis product* described in Section 2.2.1 and 11.4.3.

The second model run is a standard forecast run with a **symmetric** IAU window, starting from the uninitialized analysis for IAU product as described earlier in this Section.

The key point is that both runs are performed within a single model run by means of an internal loop structure. After ICON's read-in and initialization procedure, the first loop iteration stores the model's initial state, performs the asymmetric IAU run and saves the resulting initialized analysis to disk. During the second loop iteration, the model resets to the previously stored initial state and performs a standard forecast run with a symmetric IAU window.

The main benefit of merging these runs into a single model run is that the initial conditions (i.e. first guess and analysis increments) have to be read only once, which saves a decent amount of time in the operational forecast cycle.

11.4. Analysis Products

This section provides an overview of DWD's analysis products.

11.4.1. Uninitialized Analysis for IAU

This product is meant for starting the model in IAU mode, see Section 11.3 and Section 5.1.3. It consists of two files: a first guess file and an analysis file.

Table 11.1 provides an overview of the fields contained in the *uninitialized analysis product for IAU* for 00 UTC. Columns 1 to 3 show DWD's GRIB2 shortName, the unit and a short description of the fields, respectively. Columns 4 and 5 indicate, whether the field is part of the first guess file and/or analysis file. The marker ⊗ highlights analysis increments as opposed to full fields. First guess fields which are optional for starting the model with IAU are highlighted in blue, with their scope being indicated in the description. If one or more of these fields are unavailable, a cold-start of these fields is performed given that the parameterizations for which they are needed are activated.

As explained in Section 11.2, the atmospheric analysis is performed more frequently than the surface analysis. Therefore, the analysis product at times different from 00 UTC usually contains only a subset of the fields provided at 00 UTC. Consequently, Table 11.1 will look different for non-00 UTC runs in such a way that the fields

FR_ICE	T_SEA	W_SO	T_2M
--------	-------	------	------

will not be provided in the analysis file. The first three fields of this list will be contained in the first guess file instead.

Table 11.1.: Content of the **uninitialized analysis product for IAU**, separated into first guess (FG) and analysis (ANA). Optional fields for model initialization are marked in **blue**. The marker \otimes indicates analysis increments as opposed to full fields \times . Analysis fields highlighted in **red** are only available for 00 UTC. The validity date of the first guess is shifted back by 1.5 h w.r.t. the start date.

shortName	Unit	Description	Source	
			FG	ANA
DEN	kg m^{-3}	air density	\times	
P	Pa	pressure		\otimes
QC	kg kg^{-1}	cloud liquid water mass fraction	\times	
QI	kg kg^{-1}	cloud ice mass fraction	\times	
QR	kg kg^{-1}	rain water mass fraction	\times	
QS	kg kg^{-1}	snow mass fraction	\times	
QV	kg kg^{-1}	water vapor mass fraction	\times	\otimes
T	K	air temperature		\otimes
THETA_V	K	virtual potential temperature	\times	
TKE	$\text{m}^2 \text{s}^{-2}$	turbulent kinetic energy	\times	
U, V	m s^{-1}	horizontal velocity components		\otimes
VN	m s^{-1}	edge normal velocity component	\times	
W	m s^{-1}	vertical velocity	\times	
ALB_SEAICE	%	sea ice albedo scope: <code>lprog_albsi=.TRUE.</code> (namelist <code>lnd_nml</code>)	\times	
C_T_LK	1	shape factor w.r.t. temp. profile in the thermocline)	\times	
EVAP_PL	kg m^{-2}	evaporation of plants (integrated since "nightly reset") scope: <code>itype_trvg=3</code> (namelist <code>lnd_nml</code>)	\times	
FRESHSNW	1	age of snow indicator	\times	\otimes
FR_ICE	1	sea/lake ice fraction		\times
H_ICE	m	sea ice depth	\times	
H_ML_LK	m	mixed-layer thickness	\times	
H_SNOW	m	snow depth	\times	\otimes
HSNOW_MAX	m	maximum snow depth reached within current snow-cover period scope: <code>itype_snowevap=3</code> (namelist <code>lnd_nml</code>)	\times	
QV_S	kg kg^{-1}	surface specific humidity	\times	
RHO_SNOW	kg m^{-3}	snow density	\times	

Continued on next page

Table 11.1.: Continued from previous page

SKT	K	skin temperature	×	
		scope: <code>itype_canopy=2</code> (namelist <code>lnd_nml</code>)		
SNOAG	d	duration of current snow-cover period	×	
		scope: <code>itype_snowevap=3</code> (namelist <code>lnd_nml</code>)		
SNOWC	%	snow cover	×	
T_BOT_LK	K	temperature at water-bottom sediment interface	×	
T_G	K	surface temperature	×	
T_ICE	K	sea ice temperature	×	
T_MNW_LK	K	mean temperature of the water column	×	
T_SEA	K	sea surface temperature		×
T_SNOW	K	snow temperature	×	
T_WML_LK	K	mixed-layer temperature	×	
W_I	kg m ⁻²	water content of interception layer	×	
ZO	m	surface roughness length	×	
T_SO	K	soil temperature	×	
T_2M	K	2m temperature bias		⊗
		scope: <code>itype_vegetation_cycle=3</code> (namelist <code>extpar_nml</code>)		
T_2M_FILTBIAS	K	Time-filtered T_2M bias	×	
		scope: <code>itype_vegetation_cycle=3</code> (namelist <code>extpar_nml</code>)		
W_SO	kg m ⁻²	soil water content (liq. + ice)	×	⊗
W_SO_ICE	kg m ⁻²	soil ice content	×	

Please note that all GRIB2-keys of the fields T_2M and T_2M_FILTBIAS are identical, except for the key *typeOfGeneratingProcess*. In order to decode T_2M_FILTBIAS correctly and to distinguish it from T_2M, it is recommended to make use of the most recent DWD-specific GRIB definition files (see Section 1.2).

11.4.2. Uninitialized Analysis

The *uninitialized analysis* without IAU can be used if, for some reason, the model should be started without any noise filtering procedure. The first guess and analysis file are read in and merged by the model, i.e. the model state is abruptly pulled towards the analyzed state right before the first time integration step. This conceptually easy approach comes at the price of a massively increased noise level at the beginning of the simulation.

The validity time of the *first guess* and *analysis* must match the model’s start date. Table 11.2 provides an overview of the fields contained in the *uninitialized analysis product* for 00 UTC. Columns 4 and 5 again indicate, whether a specific field is contained in the first guess file and/or analysis file. Fields which are optional for starting the model are highlighted in blue, with their scope being indicated in the description. If one or more of these fields are unavailable, a cold-start of these fields is performed given that the parameterizations for which they are needed are activated.

As already explained in the previous section, the analysis at times different from 00 UTC will only contain a subset of the fields provided at 00 UTC. Table 11.2 will differ for non-00 UTC runs in the way that the fields

FR_ICE H_ICE T_ICE T_SEA W_SO

will not be available from the analysis file but from the first guess file.

Table 11.2.: Content of the **uninitialized analysis product**, separated into first guess (FG) and analysis (ANA). Optional fields for model initialization are marked in blue. Analysis fields highlighted in red are only available for 00 UTC.

shortName	Unit	Description	Source	
			FG	ANA
DEN	kg m ⁻³	air density	×	
P	Pa	pressure		×
QC	kg kg ⁻¹	cloud liquid water mass fraction	×	
QI	kg kg ⁻¹	cloud ice mass fraction	×	
QR	kg kg ⁻¹	rain water mass fraction	×	
QS	kg kg ⁻¹	snow mass fraction	×	
QV	kg kg ⁻¹	water vapor mass fraction		×
T	K	air temperature		×
THETA_V	K	virtual potential temperature	×	
TKE	m ² s ⁻²	turbulent kinetic energy	×	
U, V	m s ⁻¹	horizontal velocity components		×
VN	m s ⁻¹	edge normal velocity component	×	
W	m s ⁻¹	vertical velocity	×	
ALB_SEAICE	%	sea ice albedo scope: lprog_albsi=.TRUE. (namelist lnd_nml)	×	
C_T_LK	1	shape factor w.r.t. temp. profile in the thermocline)	×	

Continued on next page

Table 11.2.: Continued from previous page

EVAP_PL	kg m ⁻²	evaporation of plants (integrated since "nightly reset") scope: <code>itype_trvg=3</code> (namelist <code>lnd_nml</code>)	×	
FRESHSNW	1	age of snow indicator		×
FR_ICE	1	sea/lake ice fraction		×
H_ICE	m	sea ice depth		×
H_ML_LK	m	mixed-layer thickness	×	
H_SNOW	m	snow depth		×
HSNOW_MAX	m	maximum snow depth reached within current snow-cover period scope: <code>itype_snowevap=3</code> (namelist <code>lnd_nml</code>)	×	
QV_S	kg kg ⁻¹	surface specific humidity	×	
RHO_SNOW	kg m ⁻³	snow density	×	
SNOAG	d	duration of current snow-cover period scope: <code>itype_snowevap=3</code> (namelist <code>lnd_nml</code>)	×	
T_BOT_LK	K	temperature at water-bottom sediment interface	×	
T_G	K	surface temperature	×	
T_ICE	K	sea ice temperature		×
T_MNW_LK	K	mean temperature of the water column	×	
T_SEA	K	sea surface temperature		×
T_SNOW	K	snow temperature		×
T_WML_LK	K	mixed-layer temperature	×	
W_I	kg m ⁻²	water content of interception layer	×	
W_SNOW	kg m ⁻²	snow water equivalent	×	
ZO	m	surface roughness length	×	
T_SO	K	soil temperature	×	
W_SO	kg m ⁻²	soil water content (liq. + ice)		×
W_SO_ICE	kg m ⁻²	soil ice content	×	

11.4.3. Initialized Analysis

The *initialized analysis* is strongly related to the *uninitialized analysis for IAU*. It is a by-product of starting the model from the latter. E.g. the *initialized analysis* at 00 UTC is

generated by starting the model from the 22:30 UTC first guess, and adding the analysis increments over an asymmetric time window of 1.5 h width until 00 UTC. Therefore the noise level of the initialized analysis product is comparable to that of the *uninitialized analysis product for IAU*.

For model initialization the validity time of the *initialized analysis product* must match the model’s start date. Table 11.3 provides an overview of the fields contained in the *initialized analysis product* for 00 UTC. Fields which are optional for starting the model are highlighted in blue. If one or more of these fields are unavailable, a cold-start of these fields is performed given that the parameterizations for which they are needed are switched on.

Note that this product is also suitable for initializing COSMO limited area simulations. To this end it contains the atmospheric fields T, P, U, V rather than ICON’s set of prognostic atmospheric variables, i.e. THETA_V, RHO, VN. For ICON, the necessary transformation is performed automatically during startup.



For recent analysis dates (say August 2018 and later) this analysis product contains both SMI and W_S0. For previous analysis dates only W_S0 is available. ICON can read any of them, however SMI is preferred and W_S0 is the fallback. Whenever this analysis product needs to be remapped and SMI is not available, it is strongly recommended to manually convert W_S0 to SMI beforehand, since interpolating W_S0 can lead to strong numerical artifacts. See Section 2.2.3 for more details.

Table 11.3.: Content of the **initialized analysis product**. Fields which are optional for model initialization are marked in blue.

shortName	Unit	Description
P	Pa	pressure
QC	kg kg ⁻¹	cloud liquid water mass fraction
QI	kg kg ⁻¹	cloud ice mass fraction
QR	kg kg ⁻¹	rain water mass fraction
QS	kg kg ⁻¹	snow mass fraction
QV	kg kg ⁻¹	water vapor mass fraction
T	K	air temperature
TKE	m ² s ⁻²	turbulent kinetic energy
U, V	m s ⁻¹	horizontal velocity components
W	m s ⁻¹	vertical velocity
ALB_SEAICE	%	sea ice albedo scope: lprog_albsi=.TRUE. (namelist lnd_nml)

Continued on next page

Table 11.3.: *Continued from previous page*

C_T_LK	1	shape factor w.r.t. temp. profile in the thermocline)
EVAP_PL	kg m ⁻²	evaporation of plants (integrated since "nightly reset") scope: itype_trvg=3 (namelist lnd_nml)
FRESHSNW	1	age of snow indicator
FR_ICE	1	sea/lake ice fraction
H_ICE	m	sea ice depth
H_ML_LK	m	mixed-layer thickness
H_SNOW	m	snow depth
HSNOW_MAX	m	maximum snow depth reached within current snow-cover period scope: itype_snowevap=3 (namelist lnd_nml)
QV_S	kg kg ⁻¹	surface specific humidity
RHO_SNOW	kg m ⁻³	snow density
SNOAG	d	duration of current snow-cover period scope: itype_snowevap=3 (namelist lnd_nml)
T_BOT_LK	K	temperature at water-bottom sediment interface
T_G	K	surface temperature
T_ICE	K	sea ice temperature
T_MNW_LK	K	mean temperature of the water column
T_SNOW	K	snow temperature
T_WML_LK	K	mixed-layer temperature
W_I	kg m ⁻²	water content of interception layer
W_SNOW	kg m ⁻²	snow water equivalent
Z0	m	surface roughness length
SMI	1	soil moisture index
T_SO	K	soil temperature
W_SO	kg m ⁻²	soil water content (liq. + ice)
W_SO_ICE	kg m ⁻²	soil ice content
HHL	m	vertical coordinate half level heights

Please note that in contrast to the *uninitialized analysis for IAU* product (Table 11.1) the *initialized analysis* product does not contain the skin temperature T_SKT and the time-filtered 2 m temperature bias T_2M_FILTBIAS. If the skin temperature parameterization is activated (itype_canopy=2) but an initial T_SKT is missing, T_SKT is initialized with the surface temperature T_S. It is nevertheless recommended to switch on the skin temperature

parameterization. The lack of initial conditions for T_SKT does only have a marginal effect on the forecast quality.

Similarly, if the enhanced vegetation cycle parameterization is activated (`itype_vegetation_cycle=3`) but T_2M_FILTBIAS is missing, T_2M_FILTBIAS is initialized with zero (which effectively resembles `itype_vegetation_cycle=2`).

12. ICON-CLM

Section authors

S. Brien, V. Maurer, DWD Climate & Environment Consultancy
B. Geyer Helmholtz-Zentrum hereon

12.1. The CLM Community

The Climate Limited area Modeling Community (CLM Community) is an open international network of scientists working in the area of regional climate modeling and climate change. The members of the CLM Community develop the regional climate models COSMO-CLM and ICON-CLM together, apply the models for a wide range of applications, and collaborate on scientific projects. Together it is possible to achieve more than everybody could achieve alone. The activities are documented by the members themselves in the CLM Community portal <https://www.clm-community.eu>. It consists of several parts. Some parts, like the description of the working groups, and the member institutions, are public. The access to most of the information, like member contact information, minutes of internal meetings, and plans for projects, is restricted to community members. To become a member, follow the instructions at <https://www.clm-community.eu/Become-a-Member>.

12.2. Activities of the CLM Community

12.2.1. CORDEX

The Coordinated Regional Climate Downscaling Experiment (CORDEX, <https://cordex.org>) is the most important international program for regional climate modeling. The initiative coordinates the downscaling of the global climate projections produced in the Coupled Model Intercomparison Project (CMIP), tries to initiate and coordinate research activities, and sets standards for simulation protocols and data that are produced and published with the framework of CORDEX.

CORDEX has defined 14 standard domains that cover all land areas on the globe, including Antarctica. The standard simulations are always produced for one of these domains and fulfill several other requirements concerning e.g. grid spacing, output variables, and the period that is covered by the simulations.

The CLM Community has extensively contributed to the CORDEX simulations based on CMIP5 with COSMO-CLM. For CORDEX-CMIP6 the community will contribute with two models, namely COSMO-CLM 6.0 and ICON-CLM. Traditionally, the European

domain is the main focus of the CLM Community activities within CORDEX, because many of the member institutions are located in Europe. But community members have also produced simulations for other domains in the past, e.g. for the Mediterranean domain, Australia, Africa, Antarctica, and South-East Asia.

Besides the standard simulations, which constitute an important contribution to the Assessment Reports of the Intergovernmental Panel on Climate Change (IPCC), CORDEX also organizes several more research-orientated projects, the so-called Flagship Pilot Studies (FPS). An FPS is usually organized for one or two CORDEX domains and tries to coordinate the simulations and research to address a specific scientific question, mainly in the context of the “CORDEX Scientific Challenges”. Members from the CLM Community contributed especially to the FPS Convection (Convective phenomena at high resolution over Europe and the Mediterranean), FPS LUCAS (Land Use & Climate Across Scales), and FPS ELVIC (Climate Extremes in the Lake Viktoria Basin), and are currently involved in FPS URB-RCC (Urban environments and Regional Climate Change).

12.2.2. COPAT

The aim of the “Coordinated Parameter Testing” (COPAT) is the provision of recommended model versions and setups. The CLM Community always tries to provide well-tested model versions and optimized configurations to its members. To achieve this goal, the new versions of the community model (formerly COSMO and now ICON) are always extensively analyzed and many tests to optimize the setup are conducted before a new recommended version and configuration of the model is suggested to the community members. These tests and improvements of the model were always done in collaboration between different community members over the last two decades, ensuring a high quality of the simulation results. The current effort is named COPAT2 - Coordinated parameter testing of the COSMO6.0 version and ICON-CLM. Up to now the main focus of the tests was on the European CORDEX domain, but other contributions are very welcome.

12.3. The ICON as Regional Climate Model - ICON-CLM

ICON-CLM is the Climate Limited area Mode of the ICON modeling framework. ICON-CLM uses the same software as ICON-NWP. The main differences for regional climate simulations are in the setup of the model and the forcing data that are used on the boundaries of the model domain. The members of the CLM Community jointly develop and use the ICON model as a regional climate model. A few settings are different from the usage in NWP, which will be explained later in this chapter.

For the production of regional climate simulations, the ICON model is run in limited area mode, i.e. the namelist switch `l_limited_area = .TRUE.` (namelist `grid_nml`) is set. ICON-CLM uses initial and boundary conditions from global climate models or reanalysis data as input data (or from coarser-scale regional models). These data often come from simulations that were produced in the framework of CMIP or CORDEX, but can, in theory, come from any climate simulation that provides the necessary variables.

12.3.1. Lateral Boundary Conditions

As climate modeling is a boundary value problem, the accuracy of regional climate projections depends to a certain extent on the quality and realism of lateral boundary conditions (LBCs) sampled at regular time intervals.

In general, three types of boundary data input formats can be used in the ICON model in limited area mode: ICON, COSMO, and IFS (see also Section 6.4). It has been decided in the CLM Community to use the IFS branch for all kinds of coarse-resolution initial and boundary data (except for ICON-to-ICON offline nesting) and pre-process all input data in a way that resembles the IFS structure. Thus, the namelist variable defining the type of model initialization needs to be set to the value for IFS data: `init_mode=2` in all ICON-CLM simulations (see also Section 6.3).

The data from the global simulations must be prepared for use in ICON-CLM. The data needs to be converted into the so-called caf format, which was used earlier in the INT2LM, the preprocessor of the COSMO model. The file names are composed of the header “caf” (or “cas” if the data is not global but on a subdomain) followed by the date in the form YYYYMMDDHH; e.g. `caf1992072100.nc`.

This first preparation step is done before the simulation itself by additional converter programs that are specifically developed for each GCM. Several boundary data sets are pre-processed for ICON-CLM by the CLM Community and are stored at the DKRZ data pool. These include the reanalysis data ERAInterim, ERA5, and NCEP, but also output from some CMIP6 GCMs. In the training course, ERA5 Reanalysis data for the simulation periods are made available.

In the SPICE scripting environment (see Section 12.4), the caf files are processed in such a way that they fit the IFS file format.

More information on the initial and boundary data, including a table of the necessary input fields can be found in Section 2.2.

12.3.2. Running with Restart Files

In general, simulations in weather forecast mode take only a few hours to run, whereas they might take up to weeks or months in climate mode. Therefore, ICON-CLM is generally operated using restarts. The main switch for this in the ICON model is called `lrestart` (`master_nml`) and has to be set to `.TRUE.`. Other parameters concerning the restart can be set in the namelists `time_nml` and `io_nml`. Note that `num_restart_proc` (`parallel_nml`) has to be set to a value > 0 to enable the asynchronous writing of restart files.

12.3.3. Calendar

The standard calendar for NWP applications is the proleptic Gregorian calendar. In global climate models, it is also quite common to use either a 365 or 360-day calendar. Both of these options are available as well in the ICON model and can be specified with the namelist switch `calendar` (`master_time_control_nml`), if necessary.

12.3.4. Land Surface Scheme

The default land surface scheme in ICON is TERRA (see Section 3.8.9). Its default settings for NWP applications are described by Schulz et al. (2016). These include a vertical discretization of eight layers for temperature and water content in the soil. The zone of hydrologically active layers reaches down to 2.5 m, which corresponds to six active layers, while for temperature there are seven active layers. In COSMO-CLM, instead, it was recommended to use a discretization of ten layers, reaching further down into the soil, with a better resolution in the middle layers. The hydrologically active zone is set to an increased depth of 4.5 m. The reason is that in contrast to weather forecasts, in climate simulations the model needs to better resolve hydrological processes reaching further down into the ground, in particular during extreme events such as droughts or floods, also under climate change. This is now part of the community internal ICON-CLM version, but not yet available in the official release version.

In addition, the land surface scheme JSBACH (Reick et al., 2021) has been implemented recently in ICON-CLM (coming from the MPI-M branch of the model, the ICON-ESM). However, it has not yet been thoroughly tested for the high-resolution RCM mode (only for low-resolution global simulations) and therefore it is, up to now, not recommended by the CLM Community to be used for ICON-CLM simulations.

12.3.5. Ocean and Sea Ice

For climate simulations, the conditions of ocean and sea ice are more important than in the weather forecast mode. For weather predictions, the sea-surface temperature (SST) and sea-ice concentration (CI) are kept constant throughout the simulation, or, as in the operational setup for the global model, a climatological trend can be added to the initial fields. In ICON-CLM, the Namelist setting `sstice_mode = 6` is used, which means that the fields are usually read with the same frequency as the LBC data. They are provided in separate files called `LOWBC_YYYY_MM.nc` in SPICE, but they could also be named differently with the help of the Namelist variables `sst_td_filename` and `sst_td_filename` (`lnd_nml`).

In different groups, model setups with a regional coupled ocean are used, but the OASIS interfaces necessary for these setups are not available in the official model versions.

12.3.6. Time-dependent External Forcings

For long-term integrations, the time dependency of external forcings such as aerosols, ozone, and solar irradiance has to be taken into account. In ICON, the following options have been implemented recently and can be used, if the necessary input files are provided for the simulation:

Aerosol

the application of different aerosol data sets, also covering volcanic and anthropogenic emissions, is implemented (namelist switch: `irad_aero` (`radiation_nml`)). Depending on your application you can choose one data set or combine them.

There are monthly mean information on aerosol from Kinne climatology and volcanic aerosol from CMIP6. The recommended setting for climate projections is `irad_aero = 18`, which means that the simple plumes are used, which mimic the transient Kinne climatology because they represent the CMIP6 scenarios. Only the Kinne fields for the year 1850 are used in this case. All Kinne fields have to be interpolated onto the ICON grid before the start of the simulation.

Ozone

three-dimensional volume mixing ratio for transient ozone (namelist switch: `irad_o3 (radiation_nml)`). Ozone fields also have to be interpolated offline onto the ICON grid.

Solar irradiance

monthly mean time series of total solar irradiance and spectral solar irradiance (namelist switch: `isolrad (radiation_nml)`).

12.3.7. Model Output

In contrast to most NWP applications, the regional climate model community mostly operates with the NetCDF file format. To write the ICON output files in NetCDF format, `filetype = 4` (namelist `output_nml`) needs to be set in the output namelist streams.

One particular feature of the climate mode is that the averaging of output variables is done via the setting `operation = "mean"` in each `output_nml`. In NWP mode, averages are calculated only for particular variables like fluxes, which have particular names (like `alhfl_s` or `acclhfl_s` for the averaged or accumulated latent heat flux at the surface, respectively). These are then averaged or accumulated from the start of the model simulation. Using `operation = "mean"` or `operation = "acc"`, they are averaged or accumulated over the output interval in the respective `output_nml`. Make sure that only instantaneous variables (e.g. `lhfl_s` for the latent heat flux or `tot_prec_rate` for precipitation) are used in combination with `operation = "mean"`. One drawback of this procedure is that variables averaged in this manner cannot be written out on regular lon-lat grids (`remap = 1` (namelist `output_nml`)). At the moment, the horizontal interpolation is done in the post-processing in SPICE. Another possibility to control the output of total precipitation is to use the Namelist switch `precip_interval (io_nml)`.

To make the climate model data available to a broad user group, e.g. by distributing it via Earth System Grid Federation (ESGF) nodes like <https://esgf-data.dkrz.de/>, the data must meet specific standards. The post-processing in SPICE takes care that the model output is ready for the CCLM2CMOR tool (<https://github.com/C2SM-RCM/CCLM2CMOR>) which can then be used to standardize the data and prepare it for publication in the ESGF.

Some post-processing to adapt the NetCDF structure of the the model output fields according to the CF conventions (<https://cfconventions.org/>) is also done in the “arch” part of SPICE.

12.4. Runtime Environment SPICE

Using the experience from common community activities, a run time environment for the COSMO model (COSMO starter package) was developed several years ago. This script environment was adjusted to the needs of ICON simulations and further developed as SPICE (“Starter Package for ICON-CLM Experiments”). It consists of a package of shell scripts and some auxiliary FORTRAN programs. The package steers the entire simulation process in 5 steps: input data gathering, input data preparation (including horizontal interpolation to the ICON grid), the ICON run itself, output data archiving, and output data post-processing. The processes are managed by one main script (**subchain**) and sent to the compute server as parallel jobs where possible. This is done in a loop, usually using monthly restarts (other frequencies are possible). The settings for some High Performing Computing systems are already implemented in SPICE. Optionally steps like the transfer to the DKRZ-archive or EvaSuite lite can be added.

The SPICE package also contains some grid files for the European domain at resolutions of 0.44 and 0.11 degrees.

In the ICON-CLM Training course exercises, especially the usage of ICON-CLM in the workflow of the SPICE scripting environment is practiced. A comprehensive documentation of SPICE can be found on the webpage: <https://spice.clm-community.eu/>.

A. Table of NWP Output Variables

The following table contains the NWP variables available for output¹. Please note that the field names are following an ICON-internal nomenclature, see Section 7 for details. The list also contains tile-based fields (suffix `_t_1`, `_t_2`, ...) which are summarized as `_t_*`.

The left column lists the so called "ICON-internal" variable names, which denotes those field names that are provided as the string argument `name` to the subroutine calls `CALL add_var(...)` and `CALL add_ref(...)` inside the ICON source code. These subroutine calls have the purpose to register new variables, to allocate the necessary memory, and to set the meta-data for these variables.

Variable Name	GRIB2 Name	Description
acdnc	NDCLLOUD	Cloud droplet number concentration
adrag_u_grid		Zonal resolved surface stress mean since model start
adrag_v_grid		Meridional resolved surface stress mean since model start
aer_bc	AOD_BCARB	Black carbon aerosol
aer_du	AER_DUST	Total soil dust aerosol
aer_or	AER_ORG	Organic aerosol
aer_ss	AER_SS	Sea salt aerosol
aer_su	AER_SO4	Total sulfate aerosol
aercl_bc		Black carbon aerosol climatology
aercl_du		Total soil dust aerosol climatology
aercl_or		Organic aerosol climatology
aercl_ss		Sea salt aerosol climatology
aercl_su		Total sulfate aerosol climatology
alb_dif	ALB_DIF	Shortwave albedo for diffuse radiation
alb_si	ALB_SEAICE	Sea ice albedo (diffuse)
albdif_t_*	ALB_RAD	Tile-based shortwave albedo for diffusive radiation
albdif	ALB_RAD	Shortwave albedo for diffuse radiation
albn_dif	ALB_NI	Near IR albedo for diffuse radiation
albnirdif_t_*	ALB_NI	Tile-based near IR albedo for diffuse radiation
albnirdif	ALB_NI	Near IR albedo for diffuse radiation
albnirdir	ALNIP	Near IR albedo for direct radiation
albu_v_dif	ALB_UV	UV visible albedo for diffuse radiation
albvisdif_t_*	ALB_UV	Tile-based UV visible albedo for diffusive radiation
albvisdif	ALB_UV	UV visible albedo for diffuse radiation
albvisdir	ALUVP	UV visible albedo for direct radiation
alhfl_bs	ALHFL_BS	Latent heat flux from bare mean since model start
alhfl_pl	ALHFL_PL	Latent heat flux from plantmean since model start
alhfl_s	ALHFL_S	surface latent heat flux mean since model start

Continued on next page

¹The Table A.1 in this appendix is based on the public ICON version 2024.01-1.

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
aqhfl_s	EVAPT	surface moisture flux mean since model start
ashfl_s	ASHFL_S	Surface sensible heat flux mean since model start
asob_s	ASOB_S	Surface net solar radiation mean since model start
asob_t	ASOB_T	TOA net solar radiation mean since model start
asobclr_s	ASOB_S_CS	Clear-sky surface net solar radiation mean since model start
asod_s	ASOD_S	Surface down solar rad. mean since model start
asod_t	ASOD_T	Top down solar radiation mean since model start
asodifd_s	ASWDIFD_S	Surface down solar diff. rad. mean since model start
asodifu_s	ASWDIFU_S	Surface up solar diff. rad. mean since model start
asodird_s	ASWDIR_S	Surface down solar direct rad.mean since model start
asou_t	USWRF	Top up solar radiation mean since model start
astr_u_sso	LGWS	Zonal sso surface stress mean since model start
astr_v_sso	MGWS	Meridional sso surface stress mean since model start
aswflx_par_sfc	APAB_S	Downward PAR flux mean since model start
athb_s	ATHB_S	Surface net thermal radiation mean since model start
athb_t	ATHB_T	TOA net thermal radiation mean since model start
athbclr_s	ATHB_S_CS	Clear-sky surface net thermal radiation mean since model start
athd_s	ATHD_S	Surface down thermal radiationmean since model start
athu_s	ATHU_S	Surface up thermal radiation mean since model start
aumfl_s	AUMFL_S	U-momentum flux flux at sumean since model start
avmfl_s	AVMFL_S	V-momentum flux flux at sumean since model start
bdy_halo_c_blk		Block lists for halo points belonging to the nest boundary region
bdy_halo_c_idx		Index lists for halo points belonging to the nest boundary region
c_t_lk	C_T_LK	Shape factor (temp. profile in lake thermocline)
cape_3km	CAPE_3KM	Mean Layer CAPE, with endpoint at 3000 m
cape_ml	CAPE_ML	Cape of mean surface layer parcel
cape_mu	CAPE_MU	Most unstable CAPE, approximated by parcel with largest Θ_e in 3000 m layer
cape	CAPE_CON	Conv avail pot energy
ceiling	CEILING	Ceiling height
cin_ml	CIN_ML	Convective inhibition of mean surface layer parcel
cin_mu	CIN_MU	Most unstable convective inhibition, approximated by parcel with largest Θ_e in 3000 m layer
clch	CLCH	High level clouds
clcl	CLCL	Low level clouds
clcm	CLCM	Mid level clouds
clct_mod	CLCT_MOD	Modified total cloud cover for media
clct	CLCT	Total cloud cover
clc	CLC	Cloud cover
cldepth	CLDEPTH	Modified cloud depth for media
cli_m	QI	Specific cloud ice content (time mean)
cloud_num		Cloud droplet number concentration
cloudtop	CDCT	Cloud top height
clw_m	QC	Specific cloud water content (time mean)

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
con_gust	UVCSSZA	Convective contribution to wind gust
condhf_ice		Conductive heat flux at sea-ice bottom
conv_eis		Estimated inversion strength
cosmu0		Cosine of solar zenith angle
dbz_850	DBZ_850	Reflectivity in approx. 850 hPa
dbz_cmax	DBZ_CMAX	Column maximum reflectivity
dbz_ctmax	DBZ_CTMAX	Column and time maximum reflectivity since end of previous full 01H since model start
dbzlmx_low	DBZLMX_LOW	Reflectivity layer maximum 500 - 2500 m AGL
dbz	DBZ	Radar reflectivity in dBZ
ddqz_z_full_e	PP	Metrics functional determinant (edge)
ddqz_z_full		Metrics functional determinant
ddqz_z_half		Metrics functional determinant
ddt_adv_q*	DPSDT	Advective tracer tendency
ddt_exner_phy		Exner pressure physical tendency
ddt_grf_q*		Tracer tendency for grid refinement
ddt_pres_sfrc		Surface pressure tendency
ddt_qc_conv		Convective tendency of cloud water mass density
ddt_qc_gscp		Microphysics tendency of specific cloud water
ddt_qc_turb		Turbulence tendency of specific cloud water
ddt_qg_gscp		Microphysics tendency of graupel
ddt_qi_conv		Convective tendency of cloud ice mass density
ddt_qi_gscp		Microphysics tendency of specific cloud ice
ddt_qi_turb		Turbulence tendency of specific cloud ice
ddt_qr_conv		Convective tendency of rain mass density
ddt_qr_gscp		Microphysics tendency of rain
ddt_qs_conv		Convective tendency of snow mass density
ddt_qs_gscp		Microphysics tendency of snow
ddt_qv_conv		Convective tendency of absolute humidity
ddt_qv_gscp		Microphysics tendency of specific humidity
ddt_qv_turb		Turbulence tendency of specific humidity
ddt_temp_clcov		Sgs condensation temperature tendency
ddt_temp_drag		Sso + gwdrag temperature tendency
ddt_temp_dyn		Dynamical temperature tendency
ddt_temp_gscp		Microphysical temperature tendency
ddt_temp_pconv	DT_CON	Convective temperature tendency
ddt_temp_radlw	THHR_RAD	Long wave radiative temperature tendency
ddt_temp_radsw	SOHR_RAD	Short wave radiative temperature tendency
ddt_temp_turb	TTENDTS	Turbulence temperature tendency
ddt_tke_hsh	DTKE_HSH	TKE tendency horizontal shear production
ddt_tke_pconv	DTKE_CON	TKE tendency due to sub-grid scale convection
ddt_tke	TKETENS	Tendency of turbulent velocity scale
ddt_u_gwd	EWGD	GWD tendency of zonal wind
ddt_u_pconv	DU_CON	Convective tendency of zonal wind
ddt_u_sso	DU_SSO	Sso tendency of zonal wind
ddt_u_turb	UTENDTS	Turbulence tendency of zonal wind
ddt_ua_adv		Zonal wind tendency by advection
ddt_ua_cor		Zonal wind tendency by coriolis effect

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
ddt_ua_dmp	NSGD DV_CON DV_SSO VTENDTS	Zonal wind tendency by divergence damping
ddt_ua_dyn		Zonal wind tendency by dynamics
ddt_ua_grf		Zonal wind tendency by grid refinement
ddt_ua_hdf		Zonal wind tendency by horizontal diffusion
ddt_ua_iau		Zonal wind tendency by incremental analysis update
ddt_ua_pgr		Zonal wind tendency by pressure gradient
ddt_ua_phd		Zonal wind tendency by physics in dynamics
ddt_ua_ray		Zonal wind tendency by rayleigh damping
ddt_v_gwd		GWD tendency of meridional wind
ddt_v_pconv		Convective tendency of meridional wind
ddt_v_sso		Sso tendency of meridional wind
ddt_v_turb		Turbulence tendency of meridional wind
ddt_va_adv		Meridional wind tendency by advection
ddt_va_cor		Meridional wind tendency by coriolis effect
ddt_va_dmp		Meridional wind tendency by divergence damping
ddt_va_dyn		Meridional wind tendency by dynamics
ddt_va_grf		Meridional wind tendency by grid refinement
ddt_va_hdf		Meridional wind tendency by horizontal diffusion
ddt_va_iau		Meridional wind tendency by incremental analysis update
ddt_va_pgr		Meridional wind tendency by pressure gradient
ddt_va_phd		Meridional wind tendency by physics in dynamics
ddt_va_ray		Meridional wind tendency by rayleigh damping
ddt_vn_adv		Normal wind tendency by advection
ddt_vn_apc_pc		Advective+coriolis normal wind tendency, predictor/corrector
ddt_vn_cor_pc		Coriolis normal wind tendency, predictor/corrector
ddt_vn_cor		Normal wind tendency by coriolis effect
ddt_vn_dmp		Normal wind tendency by divergence damping
ddt_vn_dyn		Normal wind tendency by dynamics
ddt_vn_grf		Normal wind tendency by grid refinement
ddt_vn_hdf		Normal wind tendency by horizontal diffusion
ddt_vn_iau		Normal wind tendency by incremental analysis update
ddt_vn_pgr		Normal wind tendency by pressure gradient
ddt_vn_phd		Normal wind tendency by physics in dynamics
ddt_vn_phy		Normal wind physical tendency
ddt_vn_ray		Normal wind tendency by rayleigh damping
ddt_w_adv_pc		Advective vertical wind tendency, predictor/corrector
ddxn_z_full	DEPTH_LK	Terrain slope in normal direction
ddxt_z_full		Terrain slope in tangential direction
depth_lk	FI	Lake depth
dgeopot_mc		Geopotential difference between half levels
dhail_av	DHAIL_MX	Average expected hailsize since end of previous full 01H since model start
dhail_mx		Maximum expected hailsize since end of previous full 01H since model start
dhail_sd		Standard deviation of expected hailsize since end of previous full 01H since model start

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
div_ic	RDIV	Divergence at half levels
div		Divergence
dp_bs_lk		Depth of thermally active layer of bot. sediments.
dpres_mc		Pressure thickness
drag_u_grid		Zonal resolved surface stress
drag_v_grid	DURSUN_M	Meridional resolved surface stress
dursun_m		Possible astronomical maximum of sunshine
dursun_r		Relative duration of sunshine
dursun		Sunshine duration
dwdx		Zonal gradient of vertical wind
dwdy	DURSUN_R	Meridional gradient of vertical wind
dyn_gust		Dynamical gust
eai		(evaporative) earth area index
echotopinm		Maximum height of exceeding radar reflectivity threshold since end of previous full 01H since model start
echotop		Minimum pressure of exceeding radar reflectivity threshold since end of previous full 01H since model start
emis_rad	EMIS_RAD	Longwave surface emissivity
enhfac_diffu	RELHUM	Background nabla2 diffusion coefficient for upper sponge layer
evap_gmean	EXNER	Global mean evaporation flux
exner_dyn_incr		Exner dynamics increment
exner_pr		Exner perturbation pressure
exner_ref_mc		Reference atmosphere field exner
exner		Exner pressure
fac_ccqc	FETCH_LK	Factor for cloud cover - cloud water relationship
fetch_lk		Wind fetch over lake
fis		Geopotential (s)
for_d		Fraction of deciduous forest
fr_lake		Fraction lake
fr_land	FR_LAKE	Fraction land
fr_nir_sfc_diff		Diffuse fraction of downward near-infrared flux at surface
fr_par_sfc_diff		Diffuse fraction of downward photosynthetically active flux at surface
fr_seaice		Fraction of sea ice
fr_vis_sfc_diff		Diffuse fraction of downward visible flux at surface
frac_t_*	FR_LUC	Tile point area fraction list
freshsnow_t_*	FRESHSNW	Indicator for age of snow in top of snow layer
freshsnow	FRESHSNW	Weighted indicator for age of snow in top of snow layer
gamso_lk	FI	Attenuation coefficient of lake water with respect to sol. rad.
geopot_agl_ifc		Geopotential above groundlevel at cell center
geopot_agl		Geopotential above groundlevel at cell center
geopot		Geopotential at full level cell centre
graupel_gsp_rate		Gridscale graupel rate
graupel_gsp	PRG_GSP	Gridscale graupel
grf_tend_mflx	GRAU_GSP	Normal mass flux tendency (grid refinement)

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
grf_tend_rho	S_ORO_MAX	Density tendency (grid refinement)
grf_tend_thv		Virtual potential temperature tendency (grid refinement)
grf_tend_vn		Normal wind tendency (grid refinement)
grf_tend_w		Vertical wind tendency (grid refinement)
gust10	VMAX_10M	Gust at 10 m since end of previous full 01H since model start
gz0_t_*	Z0	Tile-based roughness length times gravity
gz0	Z0	Roughness length
h_b1_lk	H_B1_LK	Thickness of the upper layer of the sediments
h_ice	H_ICE	Sea/lake-ice depth
h_ml_lk	H_ML_LK	Mixed-layer thickness
h_snow_lk		Depth of snow on lake ice
h_snow_si		Depth of snow on sea ice
h_snow_t_*	H_SNOW	Snow height
h_snow	H_SNOW	Weighted snow depth
hbas_con	HBAS_CON	Height of convective cloud base
hbas_sc	HBAS_SC	Cloud base above msl, shallow convection
hdef_ic		Deformation
hfl_q*		Horizontal tracer flux
hmo3		Height of O3 maximum (Pa)
hpbl	HPBL	Boundary layer height above sea level
htop_con	HTOP_CON	Height of convective cloud top
htop_dc	HTOP_DC	Height of top of dry convection
htop_sc	HTOP_SC	Cloud top above msl, shallow convection
hus_m	QV	Specific humidity (time mean)
hzerocl	HZEROCL	Height of 0 deg C level
ice_gsp_rate	IPRATE	Gridscale ice rate
ice_gsp	IPRATE	Gridscale ice
inversion_height		Lowest inversion height
k400		Level index corresponding to the HAG of the 400hPa level
k650		Level index corresponding to the HAG of the 650hPa level
k700		Level index corresponding to the HAG of the 700hPa level
k800		Level index corresponding to the HAG of the 800hPa level
k850		Level index corresponding to the HAG of the 850hPa level
k950		Level index corresponding to the HAG of the 950hPa level
koi		Convection index
ktype		Type of convection
l_pat		Effective length scale of circulation patterns
lai	LAI	Leaf Area Index
lapse_rate	LAPSE_RATE	Temperature lapse rate 500hPa - 850hPa
lc_class_t_*	LUC	Tile point land cover class

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
lcl_ml	LCL_ML	Lifted Condensation Level of Mean Layer parcel (HAG)
lfc_ml	LFC_ML	Level of Free Convection of Mean Layer parcel (HAG)
lfd_con_max	LFD_TOT_MX	Maximum lightning flash density km-2 day-1 since end of previous full 01H (- +48h), 03H (+48 - +72h) and 06h (+72h -) since mode
lfd_con	LFD_TOT	Lightning flash density km-2 day-1
lhfl_bs_t_*		Tile-based latent heat flux from bare soil
lhfl_bs		Latent heat flux from bare soil
lhfl_pl_t_*		Tile-based latent heat flux from plants
lhfl_pl		Latent heat flux from plants
lhfl_s_t_*	LHFL_S	Tile-based surface latent heat flux
lhfl_s	LHFL_S	Surface latent heat flux
liqfl_turb		Vertical turbulent liquid water flux
low_ent_zone		Lowest limit of the entrainment zone corresponding to the lowest inversion
lpi_con_max	LPI_CON_MAX	Subgrid-scale lightning potential index, maximum since end of previous full 01H (- +48h), 03H (+48 - +72h) and 06h (+72h -) sinc
lpi_con	LPI_CON	Subgrid-scale lightning potential index
lpi_max	LPI_MAX	Lightning potential index, maximum since end of previous full 01H since model start
lpi	LPI	Lightning potential index (LPI)
lsm_ctr_c	WMB	Ocean model land-sea-mask
lsm_switch		Land-sea-mask switched by ocean
lw_emiss	EMIS_RAD	Longwave surface emissivity
lwflx_dn_clr		Longwave downward clear-sky flux
lwflx_dn		Longwave downward flux
lwflx_up_clr		Longwave upward clear-sky flux
lwflx_up		Longwave upward flux
lwflxall	NLWRF	Longwave net flux
mask_mtnpoints_g		Mask field for mountain points
mask_mtnpoints		Mask field for mountain points
mass_fl_e_sv		Storage field for horizontal mass flux at edges
mass_fl_e		Horizontal mass flux at edges
mflx_ic_ubic		Mass flux and tendency at child upper boundary
mlpi_con_max	LPI_CON_CI_MAX	Modified lightning potential index, maximum since end of previous full 01H (- +48h), 03H (+48 - +72h) and 06h (+72h -) since mod
mlpi_con	LPI_CON_CI	Modified lightning potential index
ndvi_max		NDVI yearly maximum
ndviratio	NDVIRATIO	(monthly) proportion of actual value/maximum NDVI (at init time)
o3	O3	Ozone mixing ratio
omega_z	RELV	Vertical vorticity
omega	OMEGA	Vertical velocity
pat_len		Length scale of sub-grid scale roughness elements
pfull_m	P	Pressure at full level (time mean)
phalf_m	P	Pressure at half level (time mean)

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
plcov_t_*	PLCOV	Plant covering degree in the vegetation phase
plcov	PLCOV	Plant covering degree in the vegetation phase
pme_gmean		Global mean P-E
prec_con_d		Convective precip since end of previous full 01H since model start
prec_con_rate_avg	CPRAT	Convective precip rate, time average
prec_con	PREC_CON	Convective precip
prec_gmean		Global mean precipitation flux
prec_gsp_d		Gridscale precip since end of previous full 01H since model start
prec_gsp_rate_avg	PR_GSP	Gridscale precip rate, time average
prec_gsp_rate	PR_GSP	Gridscale precipitation rate
prec_gsp	PREC_GSP	Gridscale precip
pref_aerdis		Reference pressure used for vertical distribution of aerosol optical depths
pres_ifc	P	Pressure at half level
pres_msl	PMSL	Mean sea level pressure
pres_sfc	PS	Surface pressure
pres	P	Pressure
ps_m	PS	Surface pressure (time mean)
psl_m	PMSL	Mean sea level pressure (time mean)
pv	POT_VORTIC	Potential vorticity
q_int1		Q at parent interface level
q_int2		Q at parent interface level
q_int3		Q at parent interface level
q_int4		Q at parent interface level
q_int5		Q at parent interface level
q_int6		Q at parent interface level
q_sedim	Q_SEDIM	Specific content of precipitation particles
q_ubc1		Q at child upper boundary
q_ubc2		Q at child upper boundary
q_ubc3		Q at child upper boundary
q_ubc4		Q at child upper boundary
q_ubc5		Q at child upper boundary
q_ubc6		Q at child upper boundary
qc_sgs		Subgrid-scale cloud water
qcfl_s		Surface cloud water deposition flux due to diffusion
qc	QC	Specific cloud water content
qg	QG	Specific graupel content
qhfl_s_t_*	EVAPT	Tile based surface moisture flux
qhfl_s	EVAPT	Surface moisture flux
qifl_s		Surface cloud ice deposition flux due to diffusion
qi	QI	Specific cloud ice content
qr	QR	Specific rain content
qs	QS	Specific snow content
qv_2m	QV_2M	Specific water vapor content in 2m
qv_s_t_*	QV_S	Specific humidity at the surface
qv_s	QV_S	Specific humidity at the surface

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
qv	QV	Specific humidity
r_bsmín		Minimum bare soil evaporation resistance
radtop_gmean		Global mean toa total radiation
rain_con_rate_3d	PRR_CON	3d convective rain rate
rain_con_rate	PRR_CON	Convective rain rate
rain_con	RAIN_CON	Convective rain
rain_gsp_rate	PRR_GSP	Gridscale rain rate
rain_gsp	RAIN_GSP	Gridscale rain
rain_upd		Rain in updrafts
rayleigh_vn		Rayleigh damping coefficient for vn
rayleigh_w		Rayleigh damping coefficient for w
rcld		Standard deviation of the saturation deficit
resid_wso_t_*	RESID_WSO	Residuum of the mass content of water in soil
resid_wso	RESID_WSO	Residuum of the mass content of water in soil
rh_2m_land	RELHUM_2M_L	Relative humidity in 2m over land fraction
rh_2m	RELHUM_2M	Relative humidity in 2m
rho_ic_ubc		Density and tendency at child upper boundary
rho_ic	DEN	Density at half level
rho_m	DEN	Density (time mean)
rho_ref_mc		Reference atmosphere field density
rho_ref_me		Reference atmosphere field density
rho_snow_t_*	RHO_SNOW	Snow density
rho_snow	RHO_SNOW	Weighted snow density
rho	DEN	Density
rh	RELHUM	Relative humidity
rlamh_fac_t_*		Scaling factor for rlam_heat
rlut_gmean		Global mean toa outgoing longwave radiation
rootdp	ROOTDP	Root depth of vegetation
rsdt_gmean		Global mean toa incident shortwave radiation
rsmin	RSMIN	Minimum stomatal resistance
rstom	RSTOM	Stomatal resistance
rsut_gmean		Global mean toa outgoing shortwave radiation
runoff_g_t_*	WATR	Soil water runoff
runoff_g	RUNOFF_G	Weighted soil water runoff
runoff_s_t_*	WATR	Surface water runoff
runoff_s	RUNOFF_S	Weighted surface water runoff
sai		Surface area index
scalfac_dd3d		Scaling factor for 3D divergence damping terms
sdi2	SDI_2	Supercell detection index (SDI2)
sfcfric_fac		Tuning factor for surface friction
shfl_s_t_*	SHFL_S	Tile-based surface sensible heat flux
shfl_s	SHFL_S	Surface sensible heat flux
si	SI	Showalter Index
skinc	SKC	Skin conductivity
sli	SLI	Surface Lifted Index
slope_angle		Slope angle
slope_azimuth		Slope azimuth

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
smi	SMI	Soil moisture index
snow_con_rate_3d	PRS_CON	3d convective snow rate
snow_con_rate	PRS_CON	Convective snow rate
snow_con	SNOW_CON	Convective snow
snow_gsp_rate	PRS_GSP	Gridscale snow rate
snow_gsp	SNOW_GSP	Gridscale snow
snow_melt_flux_t_*		Snow melt flux tile
snow_melt	SNOW_MELT	Snow melt amount
snowfrac_lc_t_*	SNOWC	Tile-based snow-cover fraction
snowfrac_lcu_t_*		Tile-based snow-cover fraction
snowfrac_lc	SNOWC	Snow-cover fraction
snowfrac_t_*		Local tile-based snow-cover fraction
snowfrac		Snow-cover fraction
snowlmt	SNOWLMT	Height of snow fall limit above MSL
sob_s_t_*	SOBS_RAD	Tile-based shortwave net flux at surface
sob_s	SOBS_RAD	Shortwave net flux at surface
sob_t	SOBT_RAD	Shortwave net flux at TOA
sobclr_s	SOBS_RAD_CS	Net shortwave clear-sky flux at surface
sod_t	SODT_RAD	Downward shortwave flux at TOA
sodifd_s	SWDIFDS_RAD	Shortwave diffuse downward flux at surface
soiltyp	SOILTYP	Soil type
sou_s	SWDIFUS_RAD	Shortwave upward flux at surface
sou_t	USWRF	Shortwave upward flux at TOA
sp_10m	SP_10M	Wind speed in 10m
srh	SRH	Storm relative helicity
sso_gamma	SSO_GAMMA	Anisotropy of sub-gridscale orography
sso_sigma	SSO_SIGMA	Slope of sub-gridscale orography
sso_stdh_raw		Standard deviation of sub-grid scale orography
sso_stdh	SSO_STDH	Standard deviation of sub-grid scale orography
sso_theta	SSO_THETA	Angle of sub-gridscale orography
str_u_sso	LGWS	Zonal sso surface stress
str_v_sso	MGWS	Meridional sso surface stress
swflx_dn_clr		Shortwave downward clear-sky flux
swflx_dn		Shortwave downward flux
swflx_nir_sfc		Downward near-infrared flux at surface
swflx_par_sfc	PABS_RAD	Downward photosynthetically active flux at surface
swflx_up_clr		Shortwave upward clear-sky flux
swflx_up		Shortwave upward flux
swflx_vis_sfc		Downward visible flux at surface
swiss00	SWISS00	SWISS00 Index
swiss12	SWISS12	SWISS12 Index
t_2m_land	T_2M_L	Temperature in 2m over land fraction
t_2m	T_2M	Temperature in 2m
t_b1_lk	T_B1_LK	Temperature at the bottom of the upper layer of the sediments
t_bot_lk	T_BOT_LK	Temperature at the water-bottom sediment interface
t_bs_lk		Clim. temp. at bottom of thermally active layer of sediments

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
t_cl	T_2M_CL	CRU near surface temperature climatology
t_g_t_*	T_G	Weighted surface temperature
t_g	T_G	Weighted surface temperature
t_ice	T_ICE	Sea/lake-ice temperature
t_mnw_lk	T_MNW_LK	Mean temperature of the water column
t_s_t_*	T_S	Temperature of ground surface
t_seasfc	T_SEA	Sea surface temperature
t_sk_t_*	SKT	Skin temperature
t_sk	SKT	Skin temperature
t_snow_lk		Temperature of snow on lake ice
t_snow_si		Temperature of snow on sea ice
t_snow_t_*	T_SNOW	Temperature of the snow-surface
t_snow	T_SNOW	Weighted temperature of the snow-surface
t_so_t_*	T_SO	Soil temperature (main level)
t_so	T_SO	Weighted soil temperature (main level)
t_s	T_S	Weighted temperature of ground surface
t_tilemax_inst_2m	T_2M	Instantaneous temperature in 2m, maximum over tiles
t_tilemin_inst_2m	T_2M	Instantaneous temperature in 2m, minimum over tiles
t_wml_lk	T_WML_LK	Mixed-layer temperature
ta_m	T	Temperature
tai		Transpiration area index
tas_gmean		Global mean temperature at 2m
tch_t_*	TCH	Tile-based turbulent transfer coefficients for heat
tch	TCH	Turbulent transfer coefficients for heat
tcm_t_*	TCM	Tile-based turbulent transfer coefficients for momentum
tcm	TCM	Turbulent transfer coefficients for momentum
tcond10_max	TCOND10_MX	Total column-integrated condensate above $z(T=-10$ degC), max. since end of previous full 01H since model start
tcond_max	TCOND_MAX	Total column-integrated condensate, max. since end of previous full 01H since model start
td_2m_land	TD_2M_L	Dew-point in 2m over land fraction
td_2m	TD_2M	Dew-point in 2m
temp_ifc	T	Temperature at half level
tempv	VTMP	Virtual temperature
temp	T	Temperature
tetfl_turb		Vertical turbulent theta flux
tfh		Factor of laminar transfer of scalars
tfm		Factor of laminar transfer of momentum
tfv_t_*	NSWRS	Tile-based laminar reduction factor for evaporation
tfv		Laminar reduction factor for evaporation
thb_s_t_*	THBS_RAD	Tile-based longwave net flux at surface
thb_s	THBS_RAD	Longwave net flux at surface
thb_t	THBT_RAD	Thermal net flux at TOA
thbclr_s	THBS_RAD_CS	Net longwave clear-sky flux at surface
theta_ref_ic		Reference atmosphere field theta
theta_ref_mc		Reference atmosphere field theta
theta_ref_me		Reference atmosphere field theta

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
theta_v_ic_abc		Potential temperature and tendency at child upper boundary
theta_v_ic	THETA_V	Virtual potential temperature at half levels
theta_v	THETA_V	Virtual potential temperature
thu_s	THUS_RAD	Longwave upward flux at surface
tke	TKE	Turbulent kinetic energy
tkr_t_*		Tile-based turbulent reference surface diffusion coefficient
tkred_sfc_h		Reduction factor for minimum diffusion coefficient for heat for model-DA coupling
tkred_sfc		Reduction factor for minimum diffusion coefficients for EPS perturbations
tkr		Turbulent reference surface diffusion coefficient
tkvh	TKVH	turbulent diffusion coefficients for heat
tkvm	TKVM	turbulent diffusion coefficients for momentum
tmax_2m	TMAX_2M	Max 2m temperature
tmin_2m	TMIN_2M	Min 2m temperature
topography_c	HSURF	Geometric height of the earths surface above sea level
tot_prec_d	TOT_PREC_D	Total precip since end of previous full 01H since model start
tot_prec_rate_avg	TOT_PR	Total precip rate, time average
tot_prec_rate	TOT_PR	Total precipitation rate
tot_prec	TOT_PREC	Total precip
tot_qc_dia	QC_DIA	Total specific cloud water content (diagnostic)
tot_qi_dia	QI_DIA	Total specific cloud ice content (diagnostic)
tot_qv_dia	QV_DIA	Total specific humidity (diagnostic)
tqc_dia	TQC_DIA	Total column integrated cloud water (diagnostic)
tqc	TQC	Total column integrated cloud water
tqg	TQG	Total column integrated graupel
tqi_dia	TQI_DIA	Total column integrated cloud ice (diagnostic)
tqi	TQI	Total column integrated cloud ice
tqr	TQR	Total column integrated rain
tqs	TQS	Total column integrated snow
tqv_dia	TQV_DIA	Total column integrated water vapour (diagnostic)
tqv	TQV	Total column integrated water vapour
trsola11		Shortwave net transmissivity
tsfc_ref		Reference surface temperature
tsfctrad		Surface temperature at trad
tvh_t_*		Tile-based turbulent transfer velocity for heat
tvh		Turbulent transfer velocity for heat
tvm_t_*		Tile-based turbulent transfer velocity for momentum
tvm		Turbulent transfer velocity for momentum
twater	TWATER	Total column integrated water
u_10m_t_*	U_10M	Tile-based zonal wind in 2m
u_10m	U_10M	Zonal wind in 10m
ua_m	U	Zonal wind (time mean)
uh_max_low	UH_MAX_LOW	Updraft helicity 0.0000000-3000.0000 m, max. since end of previous full 01H since model start

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
uh_max_med	UH_MAX_MED	Updraft helicity 2000.0000-5000.0000 m, max. since end of previous full 01H since model start
uh_max	UH_MAX	Updraft helicity 2000.0000-8000.0000 m, max. since end of previous full 01H since model start
umfl_s_t_*	UMFL_S	U-momentum flux at the surface
umfl_s	UMFL_S	U-momentum flux at the surface
urb_isa	FR_PAVED	Impervious surface area fraction
u	U	Zonal wind
v_10m_t_*	V_10M	Tile-based meridional wind in 2m
v_10m	V_10M	Meridional wind in 10m
va_m	V	Meridional wind (time mean)
vapfl_turb		Vertical turbulent water vapour flux
vfl_q*		Vertical tracer flux
vio3	TOZNE	Vertically integrated ozone amount
vis	VIS	Near surface visibility
vmfl_s_t_*	VMFL_S	V-momentum flux at the surface
vmfl_s	VMFL_S	V-momentum flux at the surface
vn_ie_int		Normal velocity and tendency at parent interface level
vn_ie_ubc		Normal velocity and tendency at child upper boundary
vn_ie	VN	Normal wind at half level
vn	VN	Velocity normal to edge
vorw_ctmax	VORW_CTMAX	Maximum rotation amplitude since end of previous full 01H since model start
vor	RELV	Vorticity
vt	VT	Tangential-component of wind
vwind_expl_wgt		Explicit weight in vertical wind solver
vwind_impl_wgt		Implicit weight in vertical wind solver
v	V	Meridional wind
w_concorr_c		Contravariant vertical correction
w_ctmax	W_CTMAX	Maximum updraft track since end of previous full 01H since model start
w_i_t_*	W_I	Weighted water content of interception water
w_i	W_I	Weighted water content of interception water
w_snow_t_*	W_SNOW	Water equivalent of snow
w_snow	W_SNOW	Weighted water equivalent of snow
w_so_ice_t_*	W_SO_ICE	Ice content
w_so_ice	W_SO_ICE	Ice content
w_so_t_*	W_SO	Total water content (ice + liquid water)
w_so	W_SO	Total water content (ice + liquid water)
w_ubc		Vertical velocity and tendency at child upper boundary
wa_m	W	Vertical velocity (time mean)
wap_m	OMEGA	Vertical velocity (time mean)
wdur	W_UP_DUR	Updraft duration
wshear_u	WSHEAR_U	U-component of vertical wind shear vector
wshear_v	WSHEAR_V	V-component of vertical wind shear vector
wup_mask	W_UP_MASK	Updraft mask
ww	WW	Significant weather

Continued on next page

Table A.1 – Continued from previous page

Variable Name	GRIB2 Name	Description
w	W	Vertical velocity
z_ifc	HHL	Geometric height at half level center
z_mc	H	Geometric height at full level center
zd_blklist		Missing description
zd_diffcoef		Missing description
zd_e2cell		Missing description
zd_edgeblk		Missing description
zd_edgeidx		Missing description
zd_geofac		Missing description
zd_indlist		Missing description
zd_intcoef		Missing description
zd_vertidx		Missing description

List of ICON Variable Groups

The "group:" keyword for the namelist parameters `ml_varlist`, `hl_varlist`, `pl_varlist` (namelist `output_nml`) can be used to activate a set of common variables for output at once. The following lists contain the variables for each of these groups (empty groups and groups with only a single entry are omitted).

- A. Output Variables

Group ADDITIONAL_PRECIP_VARS
cape, clct, clct_mod, cloudtop, prec_con_rate_avg, prec_gsp_rate_avg, si, sli, swiss*, tqc_dia, tqi_dia, tqv_dia

Group ATMO_DERIVED_VARS
div, omega, omega_z, vor

Group ATMO_ML_VARS
cli_m, clw_m, hus_m, pres, qc, qg, qi, qr, qs, qv, temp, tke, u, v, w

Group ATMO_PL_VARS
cli_m, clw_m, hus_m, qc, qg, qi, qr, qs, qv, temp, tke, u, v, w

Group ATMO_TIMEMEAN
cli_m, clw_m, hus_m, pfull_m, phalf_m, ps_m, psl_m, rho_m, ta_m, ua_m, va_m, wa_m, wap_m

Group ATMO_ZL_VARS
cli_m, clw_m, hus_m, pres, qc, qg, qi, qr, qs, qv, temp, tke, u, v, w

Group CLOUD_DIAG
clc, qc_sgs, tot_qc_dia, tot_qi_dia, tot_qv_dia

Group DWD_FG_ATM_VARS
pres, pres_sfc, qc, qg, qi, qr, qs, qv, rho, t_2m, td_2m, temp, theta_v, tke, u, u_10m, v, v_10m, vn, w, z_ifc

Group DWD_FG_SFC_VARS

alb_si, c_t_lk, fr_land, fr_seaice, freshsnow, gz0, h_ice, h_ml_lk, h_snow, qv_s,
rho_snow, snowfrac_lc, t_bot_lk, t_g, t_ice, t_mnw_lk, t_seasfc, t_sk, t_snow, t_so,
t_wml_lk, w_i, w_snow, w_so, w_so_ice

Group DWD_FG_SFC_VARS_T

freshsnow_t_*, h_snow_t_*, qv_s_t_*, rho_snow_t_*, snowfrac_lc_t_*, t_g_t_*,
t_sk_t_*, t_snow_t_*, t_so_t_*, w_i_t_*, w_snow_t_*, w_so_ice_t_*, w_so_t_*

Group IAU_INIT_VARS

ddt_temp_dyn, ddt_temp_radlw, ddt_temp_rads, ddt_temp_turb, ddt_vn_phy,
exner_dyn_incr, ice_gsp_rate, rain_gsp_rate, snow_gsp_rate

Group IAU_RESTORE_VARS

alb_si, c_t_lk, exner, fr_seaice, gz0, h_b1_lk, h_ice, h_ml_lk, h_snow, rho, t_b1_lk,
t_bot_lk, t_ice, t_mnw_lk, t_wml_lk, theta_v, tke, vn, w

Group ICON_LBC_VARS

pres, qc, qg, qi, qr, qs, qv, temp, tke, u, v, w, z_ifc

Group LAND_TILE_VARS

h_snow_t_*, qv_s_t_*, rho_snow_t_*, snowfrac_lc_t_*, snowfrac_t_*, t_g_t_*,
t_s_t_*, t_sk_t_*, t_snow_t_*, t_so_t_*, w_i_t_*, w_snow_t_*, w_so_ice_t_*,
w_so_t_*

Group LAND_VARS

qv_s, rho_snow, snowfrac, snowfrac_lc, t_g, t_snow, t_so, w_i, w_snow, w_so, w_so_ice

Group LATBC_PREFETCH_VARS

pres, pres_sfc, qc, qg, qi, qr, qs, qv, rho, temp, theta_v, u, v, vn, w, z_ifc

Group MODE_COMBINED_IN

alb_si, fr_seaice, freshsnow, h_ice, h_snow, qv_s, rho_snow, t_g, t_ice, t_sk,
t_snow, t_so, w_i, w_snow, w_so

Group MODE_COSMO_IN

alb_si, freshsnow, h_ice, qv_s, rho_snow, t_g, t_ice, t_snow, t_so, w_i, w_snow, w_so

Group MODE_DWD_ANA_IN

fr_seaice, freshsnow, h_ice, h_snow, pres, qv, t_ice, t_seasfc, t_snow, t_so, temp, u,
v, w_so

Group MODE_DWD_FG_IN

alb_si, c_t_lk, gz0, h_ml_lk, qc, qg, qi, qr, qs, qv_s, rho, rho_snow, t_bot_lk, t_g,
t_mnw_lk, t_sk, t_so, t_wml_lk, theta_v, tke, vn, w, w_i, w_snow, w_so_ice, z_ifc

Group MODE_IAU_ANAATM_IN

pres, qc, qg, qi, qr, qs, qv, temp, u, v

Group MODE_IAU_ANA_IN

fr_seaice, freshsnow, h_snow, pres, qc, qg, qi, qr, qs, qv, t_seasfc, t_so, temp, u, v,
w_so

Group MODE_IAU_FG_IN

alb_si, c_t_lk, freshsnow, gz0, h_ice, h_ml_lk, h_snow, qc, qg, qi, qr, qs, qv, qv_s,
rho, rho_snow, snowfrac_lc, t_bot_lk, t_g, t_ice, t_mnw_lk, t_sk, t_snow, t_so,
t_wml_lk, theta_v, tke, vn, w, w_i, w_so, w_so_ice

Group MODE_IAU_OLD_ANA_IN

fr_seaice, freshsnow, h_snow, pres, qv, rho_snow, t_seasfc, t_so, temp, u, v, w_snow, w_so

Group MODE_IAU_OLD_FG_IN

alb_si, c_t_lk, gz0, h_ice, h_ml_lk, qc, qg, qi, qr, qs, qv, qv_s, rho, t_bot_lk, t_g, t_ice, t_mnw_lk, t_snow, t_so, t_wml_lk, theta_v, tke, vn, w, w_i, w_so, w_so_ice

Group MODE_INIANA

alb_si, c_t_lk, fr_land, fr_seaice, freshsnow, gz0, h_ice, h_ml_lk, h_snow, pres, qc, qg, qi, qr, qs, qv, qv_s, rho_snow, smi, t_bot_lk, t_g, t_ice, t_mnw_lk, t_snow, t_so, t_wml_lk, temp, tke, u, v, w, w_i, w_snow, w_so_ice, z_ifc

Group NH_PROG_VARS

exner, rho, theta_v, vn

Group PBL_VARS

alhfl_s, aqhfl_s, ashfl_s, gust10, lhfl_bs, lhfl_s, qhfl_s, qv_2m, shfl_s, t_2m, t_2m_land, tch, tcm, td_2m, td_2m_land, tkr, tkvh, tkvm, tvh, tvn, u_10m, v_10m

Group PHYS_TENDENCIES

ddt_qc_conv, ddt_qc_gscp, ddt_qc_turb, ddt_qg_gscp, ddt_qi_conv, ddt_qi_gscp, ddt_qi_turb, ddt_qr_conv, ddt_qr_gscp, ddt_qs_conv, ddt_qs_gscp, ddt_qv_conv, ddt_qv_gscp, ddt_qv_turb, ddt_temp_clcov, ddt_temp_drag, ddt_temp_gscp, ddt_temp_pconv, ddt_temp_radlw, ddt_temp_rads, ddt_temp_turb, ddt_tke, ddt_tke_hsh, ddt_tke_pconv, ddt_u_gwd, ddt_u_pconv, ddt_u_sso, ddt_u_turb, ddt_v_gwd, ddt_v_pconv, ddt_v_sso, ddt_v_turb

Group PRECIP_VARS

graupel_gsp, prec_con, prec_gsp, rain_con, rain_gsp, snow_con, snow_gsp, tot_prec

Group PROG_TIMEMEAN

pfull_m, phalf_m, ps_m, psl_m, rho_m, ta_m, ua_m, va_m, wa_m, wap_m

Group RAD_VARS

albdif, albnirdif, albvisdif, asob_s, asob_t, asod_t, asodifd_s, asodifu_s, asodird_s, asou_t, aswflx_par_sfc, athb_s, athb_t, athd_s, athu_s, cosmu0, fr_nir_sfc_diff, fr_par_sfc_diff, fr_vis_sfc_diff, sob_s, sob_s_t_*, sob_t, sod_t, sodifd_s, sou_s, sou_t, swflx_nir_sfc, swflx_par_sfc, swflx_vis_sfc, thb_s, thb_s_t_*, thb_t, thu_s

Group SNOW_VARS

rho_snow, t_snow

Group TRACER_TIMEMEAN

cli_m, clw_m, hus_m

Bibliography

- Asensio, H., and M. Messmer, 2014: *External Parameters for Numerical Weather Prediction and Climate Application: EXTPAR v2.0.2 User and Implementation Guide*. Consortium for Small-scale Modeling (COSMO), URL http://www.cosmo-model.org/content/model/modules/Extpar_201408_user_and_implementation_manual.pdf.
- Avissar, R., and R. Pielke, 1989: A parameterization of heterogeneous land surfaces for atmospheric numerical models and its impact on regional meteorology. *Mon. Weather Rev.*, **117**, 2113–2136.
- Baines, P., and T. Palmer, 1990: Rationale for a new physically based parameterization of sub-grid scale orographic effects. Tech. Rep. 169, European Centre for Medium-Range Weather Forecasts, 11 pp. URL <http://www.ecmwf.int>.
- Baldauf, M., A. Seifert, J. Förstner, D. Majewski, M. Raschendorfer, and T. Reinhardt, 2011: Operational Convective-Scale Numerical Weather Prediction with the COSMO Model: Description and Sensitivities. *Mon. Weather Rev.*, **139** (12), 3887–3905, doi:10.1175/MWR-D-10-05013.1.
- Barker, H. W., G. L. Stephens, P. T. Partain, and Coauthors, 2003: Assessing 1D atmospheric solar radiative transfer models: Interpretation and handling of unresolved clouds. *J. Clim.*, **16** (16), 2676–2699, doi:10.1175/1520-0442(2003)016<2676:ADASRT>2.0.CO;2.
- Bechtold, P., 2017: Atmospheric moist convection. *Meteorological Training Course Lecture Series*, ECMWF, 1–78, URL <https://www.ecmwf.int/sites/default/files/elibrary/2017/16953-atmospheric-moist-convection.pdf>.
- Bechtold, P., M. Köhler, T. Jung, F. Doblas-Reyes, M. Leutbecher, M. J. Rodwell, F. Vitart, and G. Balsamo, 2008: Advances in simulating atmospheric variability with the ECMWF model: From synoptic to decadal time-scales. *Q. J. R. Meteorol. Soc.*, **134** (634), 1337–1351, doi:10.1002/qj.289.
- Bechtold, P., N. Semane, P. Lopez, J.-P. Chaboureaud, A. Beljaars, and N. Bormann, 2014: Representing equilibrium and nonequilibrium convection in large-scale models. *J. Atmos. Sci.*, **71** (2), 734–753, doi:10.1175/JAS-D-13-0163.1.
- Blackadar, A. K., 1962: The vertical distribution of wind and turbulent exchange in a neutral atmosphere. *J. Geophys. Res.*, **67**, 3095–3102.
- Bloom, S. C., L. L. Takacs, A. M. D. Silva, and D. Ledvina, 1996: Data assimilation using incremental analysis updates. *Mon. Weather Rev.*, **124**, 1256–1270.

- Bonanni, A., J. Hawkes, and T. Quintino, 2023: Plume: A plugin mechanism for numerical weather prediction models. doi:<https://doi.org/10.5194/egusphere-egu23-7944>, EGU General Assembly 2023, Vienna, Austria, 24–28 Apr 2023.
- Borchert, S., G. Zhou, M. Baldauf, H. Schmidt, G. Zängl, and D. Reinert, 2019: The upper-atmosphere extension of the ICON general circulation model (version: ua-icon-1.0). *Geosci. Model Dev.*, **12** (8), 3541–3569, doi:10.5194/gmd-12-3541-2019, URL <https://gmd.copernicus.org/articles/12/3541/2019/>.
- Buehner, M., and Coauthors, 2015: Implementation of Deterministic Weather Forecasting Systems Based on Ensemble-Variational Data Assimilation at Environment Canada. Part I: The Global System. *Mon. Weather Rev.*, **143** (7), 2532–2559, doi:10.1175/MWR-D-14-00354.1, URL <https://journals.ametsoc.org/view/journals/mwre/143/7/mwr-d-14-00354.1.xml>.
- Choulga, M., E. Kourzeneva, E. Zakharova, and A. Doganovsky, 2014: Estimation of the mean depth of boreal lakes for use in numerical weather prediction and climate modelling. *Tellus A*, **66**, 21 295, doi:10.3402/tellusa.v66.21295.
- Colella, P., and P. R. Woodward, 1984: The piecewise parabolic method (ppm) for gas-dynamical simulations. *J. Comput. Phys.*, **54**, 174–201.
- Crueger, T., and Coauthors, 2018: ICON-A, the atmosphere component of the ICON earth system model: II. model evaluation. *J. Adv. Model Earth Sy.*, **10** (7), 1638–1662, doi:10.1029/2017MS001233.
- Davies, H., 1976: A lateral boundary formulation for multi-level prediction models. *Q. J. R. Meteorol. Soc.*, **102**, 405–418, doi:10.1002/qj.49710243210.
- Davies, H., 1983: Limitations of some common lateral boundary schemes used in regional NWP models. *Mon. Weather Rev.*, **111**, 1002–1012, doi:10.1175/1520-0493(1983)111<1002:LOSCLB>2.0.CO;2.
- Dipankar, A., B. Stevens, R. Heinze, C. Moseley, G. Zängl, M. Giorgetta, and S. Brdar, 2015: Large eddy simulation using the general circulation model ICON. *J. Adv. Model Earth Sy.*, **7** (3), 963–986, doi:10.1002/2015MS000431.
- Doms, G., and M. Baldauf, 2018: *A Description of the Nonhydrostatic Regional COSMO-Model. Part I: Dynamics and Numerics*. Consortium for Small-Scale Modelling, URL <http://www.cosmo-model.org/public/documentation.htm>.
- Doms, G., U. Schättler, and J.-P. Schulz, 2003: *Kurze Beschreibung des Lokal-Modells LM und seiner Datenbanken auf dem Datenserver des DWD*. Deutscher Wetterdienst (DWD), URL https://www.dwd.de/EN/ourservices/reports_on_icon/reports_on_icon.html.
- Doms, G., and Coauthors, 2011: *A Description of the Nonhydrostatic Regional COSMO Model. Part II: Physical Parameterization*. Consortium for Small-Scale Modelling, URL <http://www.cosmo-model.org>.
- Easter, R. C., 1993: Two modified versions of Bott’s positive-definite numerical advection scheme. *Mon. Weather Rev.*, **121**, 297–304.

- ECMWF, 2017: *PART IV: PHYSICAL PROCESSES*, chap. Convection, 75–95. IFS Documentation, ECMWF.
- ECMWF, 2018a: *PART IV: PHYSICAL PROCESSES*, chap. Non-ogographic gravity wave drag, 69–74. IFS Documentation, ECMWF.
- ECMWF, 2018b: *PART IV: PHYSICAL PROCESSES*, chap. Subgrid-scale orographic drag, 59–67. IFS Documentation, ECMWF.
- Ern, M., P. Preusse, and C. D. Warner, 2006: Some experimental constraints for spectral parameters used in the warner and mcintyre gravity wave parameterization scheme. *Atmos. Chem. Phys.*, **6** (12), 4361–4381, doi:10.5194/acp-6-4361-2006.
- Gal-Chen, T., and R. Somerville, 1975: On the use of a coordinate transformation for the solution of the Navier-Stokes equations. *J. Comput. Phys.*, **17**, 209–228.
- Gallus, W. A., and M. Rančić, 1996: A non-hydrostatic version of the NMC’s regional Eta model. *Q. J. R. Meteorol. Soc.*, **122**, 495–513.
- Gassmann, A., 2013: A global hexagonal C-grid non-hydrostatic dynamical core (ICON-IAP) designed for energetic consistency. *Q. J. R. Meteorol. Soc.*, **139** (670), 152–175, doi:10.1002/qj.1960.
- Gassmann, A., and H.-J. Herzog, 2008: Towards a consistent numerical compressible non-hydrostatic model using generalized Hamiltonian tools. *Q. J. R. Meteorol. Soc.*, **134** (635), 1597–1613.
- Giorgetta, M., and Coauthors, 2018: ICON-A, the Atmosphere Component of the ICON Earth System Model: I. Model Description. *J. Adv. Model Earth Sy.*, **10** (7), 1613–1637, doi:10.1029/2017MS001242.
- Giorgetta, M. A., and Coauthors, 2022: The icon-a model for direct qbo simulations on gpus (version icon-cscs:baf28a514). *Geosci. Model Dev.*, **15** (18), 6985–7016, doi:10.5194/gmd-15-6985-2022.
- GLOBE-Task-Team, 1999: The Global Land One-kilometer Base Elevation (GLOBE) Digital Elevation Model, version 1.0. Tech. rep., National Oceanic and Atmospheric Administration. URL <http://www.ngdc.noaa.gov/mgg/topo/globe.html>.
- Grabowski, W. W., and H. Morrison, 2021: Supersaturation, buoyancy, and deep convection dynamics. *Atmospheric Chemistry and Physics*, **21** (18), 13997–14018, doi:10.5194/acp-21-13997-2021, URL <https://acp.copernicus.org/articles/21/13997/2021/>.
- Grell, G. A., J. Dudhia, and D. Stauffer, 1994: A description of the fifth-generation Penn State/NCAR Mesoscale Model (MM5). Tech. Rep. NCAR/TN-398+STR, University Corporation for Atmospheric Research. doi:10.5065/D60Z716B.
- Guerra, J. E., and P. A. Ullrich, 2016: A high-order staggered finite-element vertical discretization for non-hydrostatic atmospheric models. *Geosci. Model Dev.*, **9** (5), 2007–2029, doi:10.5194/gmd-9-2007-2016, URL <https://gmd.copernicus.org/articles/9/2007/2016/>.

- Ha, S., J. J. Guerrette, I. Hernández Baños, W. C. Skamarock, and M. G. Duda, 2024: Incremental analysis update (IAU) in the Model for Prediction Across Scales coupled with the Joint Effort for Data assimilation Integration (MPAS-JEDI 2.0.0). *Geosci. Model Dev.*, **17** (10), 4199–4211, doi:10.5194/gmd-17-4199-2024, URL <https://gmd.copernicus.org/articles/17/4199/2024/>.
- Harris, L. M., and P. H. Lauritzen, 2010: A flux-form version of the conservative semi-lagrangian multi-tracer transport scheme (CSLAM) on the cubed sphere grid. *J. Comput. Phys.*, **230**, 1215–1237.
- Hartung, K., and Coauthors, 2023: Poster: The ICON community interface ComIn. *24th ICON/COSMO/CLM/ART User Seminar, ICCARUS conference*.
- Heinze, R., and Coauthors, 2017: Large-eddy simulations over Germany using ICON: a comprehensive evaluation. *Q. J. R. Meteorol. Soc.*, **143** (702), 69–100.
- Heise, E., B. Ritter, and R. Schrodin, 2006: Operational implementation of the multilayer soil model. *COSMO Technical Reports No. 9*, Consortium for Small-Scale Modelling, 19pp, URL <http://www.cosmo-model.org>.
- Hesselberg, A., 1925: Die Gesetze der ausgeglichenen atmosphärischen Bewegungen. *Beitr. Phys. Atmos.*, **12**, 141–160.
- Heymsfield, A. J., and L. J. Donner, 1990: A scheme for parameterizing ice-cloud water content in general circulation models. *J. Atmos. Sci.*, **47** (15), 1865–1877.
- Hogan, R. J., and A. Bozzo, 2018: A flexible and efficient radiation scheme for the ecmwf model. *J. Adv. Model Earth Sy.*, **10** (8), 1990–2008.
- Hogan, R. J., and A. J. Illingworth, 2000: Deriving cloud overlap statistics from radar. *Q. J. R. Meteorol. Soc.*, **126** (569), 2903–2909.
- Hohenegger, C., and Coauthors, 2023: ICON-Sapphire: simulating the components of the Earth system and their interactions at kilometer and subkilometer scales. *Geosci. Model Dev.*, **16** (2), 779–811, doi:10.5194/gmd-16-779-2023.
- Hunt, B. R., E. Kostelich, and I. Szunyogh, 2007: Efficient data assimilation for spatiotemporal chaos: A Local Ensemble Transform Kalman Filter. *Physica D*, **230**, 112–126.
- ICON Press Release, 2024: Milestone in climate and weather research: Weather and climate model icon published under open source license. Deutsches Klimarechenzentrum, Presse- und Öffentlichkeitsarbeit, URL <https://idw-online.de/de/news827865>.
- Jablonowski, C., P. Lauritzen, R. Nair, and M. Taylor, 2008: *Idealized test cases for the dynamical cores of Atmospheric General Circulation Models: A proposal for the NCAR ASP 2008 summer colloquium*. National Center for Atmospheric Research (NCAR).
- Jablonowski, C., and D. L. Williamson, 2006: A baroclinic instability test case for atmospheric model dynamical cores. *Q. J. R. Meteorol. Soc.*, **132**, 2943–2975.
- Khain, A., A. Pokrovsky, M. Pinsky, A. Seifert, and V. Phillips, 2004: Simulation of Effects of Atmospheric Aerosols on Deep Turbulent Convective Clouds Using a Spectral Microphysics Mixed-Phase Cumulus Cloud Model. Part I: Model Description and Possible Applications. *J. Atmos. Sci.*, **61** (24), 2963–2982, doi:10.1175/JAS-3350.1.

- Khain, A. P., and I. Sednev, 1996: Simulation of precipitation formation in the eastern mediterranean coastal zone using a spectral microphysics cloud ensemble model. *Atmos. Res.*, **43** (1), 77–110, doi:10.1016/S0169-8095(96)00005-1.
- Klemp, J., 2011: A terrain-following coordinate with smoothed coordinate surfaces. *Mon. Weather Rev.*, **139**, 2163–2169.
- Klemp, J., J. Dudhia, and A. Hassiotis, 2008: An upper gravity-wave absorbing layer for NWP applications. *Mon. Weather Rev.*, **136** (10), 3987–4004.
- Klemp, J. B., W. C. Skamarock, and J. Dudhia, 2007: Conservative split-explicit time integration methods for the compressible nonhydrostatic equations. *Mon. Weather Rev.*, **135** (8), 2897 – 2913.
- Kogan, Y. L., and W. J. Martin, 1994: Parameterization of bulk condensation in numerical cloud models. *Journal of the Atmospheric Sciences*, **51** (17), 1728–1739, doi:10.1175/1520-0469(1994)051<1728:POBCIN>2.0.CO;2.
- Kolmogorov, A. N., 1968: Local structure of turbulence in an incompressible viscous fluid at very high reynolds numbers. *Sov. Phys. Usp.*, **10** (6), 734.
- Korn, P., 2017: Formulation of an unstructured grid model for global ocean dynamics. *J. Comput. Phys.*, **339** (C), 525–552, doi:10.1016/j.jcp.2017.03.009.
- Korn, P., and S. Danilov, 2017: Elementary dispersion analysis of some mimetic discretizations on triangular C-grids. *J. Comput. Phys.*, **330**, 156 – 172, doi:https://doi.org/10.1016/j.jcp.2016.10.059.
- Korn, P., and Coauthors, 2022: ICON-O: The Ocean Component of the ICON Earth System Model–Global Simulation Characteristics and Local Telescoping Capability. *J. Adv. Model Earth Sy.*, **14** (10), e2021MS002952, doi:10.1029/2021MS002952.
- Korolev, A., and I. Mazin, 2003: Supersaturation of water vapor in clouds. *Journal of the Atmospheric Sciences*, **60** (24), 2957–2974, doi:10.1175/1520-0469(2003)060<2957:SOWVIC>2.0.CO;2.
- Kourzeneva, E., 2010: External data for lake parameterization in Numerical Weather Prediction and climate modeling. *Boreal Env. Res.*, **15**, 165–177.
- Kourzeneva, E., H. Asensio, E. Martin, and S. Faroux, 2012: Global gridded dataset of lake coverage and lake depth for use in numerical weather prediction and climate modelling. *Tellus A*, **64**, 15 640, doi:10.3402/tellusa.v64i0.15640.
- Lauritzen, P. H., C. Erath, and R. Mittal, 2011a: On simplifying ‘incremental remap’-based transport schemes. *J. Comput. Phys.*, **230**, 7957–7963.
- Lauritzen, P. H., C. Jablonowski, M. A. Taylor, and R. D. Nair, 2011b: *Numerical Techniques for Global Atmospheric Models*. 1st ed., Springer, 556 pp.
- Lauritzen, P. H., R. D. Nair, and P. A. Ullrich, 2010: A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid. *J. Comput. Phys.*, **229**, 1401–1424.

- Lebo, Z. J., H. Morrison, and J. H. Seinfeld, 2012: Are simulated aerosol-induced effects on deep convective clouds strongly dependent on saturation adjustment? *Atmospheric Chemistry and Physics*, **12**, 9941–9964, doi:10.5194/acp-12-9941-2012.
- Leuenberger, D., M. Koller, and C. Schär, 2010: A generalization of the SLEVE vertical coordinate. *Mon. Weather Rev.*, **138**, 3683–3689.
- Lilly, D. K., 1962: On the numerical simulation of buoyant convection. *Tellus*, **14** (2), 148–172, doi:10.1111/j.2153-3490.1962.tb00128.x.
- Lin, S.-J., and R. B. Rood, 1996: Multidimensional flux-form semi-lagrangian transport schemes. *Mon. Weather Rev.*, **124** (9), 2046–2070, doi:10.1175/1520-0493(1996)124<2046:MFFSLT>2.0.CO;2.
- Lipscomb, W. H., and T. D. Ringler, 2005: An incremental remapping transport scheme on a spherical geodesic grid. *Mon. Weather Rev.*, **133**, 2335–2350.
- Lott, F., and M. J. Miller, 1997: A new subgrid-scale orographic drag parametrization: Its formulation and testing. *Q. J. R. Meteorol. Soc.*, **123** (537), 101–127, doi:10.1002/qj.49712353704.
- Matsuno, T., 1966: Numerical integrations of the primitive equations by a simulated backward difference method. *J. Meteorolog. Soc. Jpn.*, **44** (1), 76–84, doi:10.2151/jmsj1965.44.1_76.
- McLandress, C., and J. F. Scinocca, 2005: The GCM response to current parameterizations of nonorographic gravity wave drag. *J. Atmos. Sci.*, **62** (7), 2394–2413, doi:10.1175/JAS3483.1.
- Mellor, G. L., and T. Yamada, 1982: Development of a turbulence closure model for geophysical fluid problems. *Rev. Geophys.*, **20** (4), 851–875, doi:10.1029/RG020i004p00851.
- Mironov, D., E. Heise, E. Kourzeneva, B. Ritter, N. Schneider, and A. Terzhevik, 2010: Implementation of the lake parameterisation scheme FLake into the numerical weather prediction model COSMO. *Boreal Env. Res.*, **15**, 218–230.
- Mironov, D., B. Ritter, J.-P. Schulz, M. Buchhold, M. Lange, and E. Machulskaya, 2012: Parameterisation of sea and lake ice in numerical weather prediction models of the German weather service. *Tellus A*, **64** (0), doi:10.3402/tellusa.v64i0.17330.
- Mironov, D. V., 2008: Parameterization of lakes in numerical weather prediction. Description of a lake model. COSMO Technical Report, Consortium for Small-Scale Modelling, 41 pp. URL <http://www.cosmo-model.org>.
- Miura, H., 2007: An upwind-biased conservative advection scheme for spherical hexagonal-pentagonal grids. *Mon. Weather Rev.*, **135**, 4038–4044.
- Mlawer, E. J., S. J. Taubman, P. D. Brown, M. J. Iacono, and S. A. Clough, 1997: Radiative transfer for inhomogeneous atmospheres: RRTM, a validated correlated-k model for the longwave. *J. Geophys. Res.: Atmos.*, **102** (D14), 16 663–16 682, doi:10.1029/97JD00237.
- Narcowich, F. J., and J. D. Ward, 1994: Generalized Hermite interpolation via matrix-valued conditionally positive definite functions. *Math. Comp.*, **63**, 661–687, doi:10.1090/S0025-5718-1994-1254147-6.

- Orr, A., P. Bechtold, J. Scinocca, M. Ern, and M. Janiskova, 2010: Improved middle atmosphere climate and forecasts in the ECMWF model through a nonorographic gravity wave drag parameterization. *J. Clim.*, **23** (22), 5905–5926, doi:10.1175/2010JCLI3490.1.
- Pincus, R., H. W. Barker, and J.-J. Morcrette, 2003: A fast, flexible, approximate technique for computing radiative transfer in inhomogeneous cloud fields. *J. Geophys. Res.: Atmos.*, **108** (D13).
- Polavarapu, S., S. Ren, A. M. Clayton, D. Sankey, and Y. Rochon, 2004: On the relationship between incremental analysis updating and incremental digital filtering. *Mon. Weather Rev.*, **132**, 2495–2502.
- Politovich, M. K., and W. A. Cooper, 1988: Variability of supersaturation in a continental cloud. *Journal of the Atmospheric Sciences*, **45** (11), 1651–1664, doi:10.1175/1520-0469(1988)045<1651:VOSIAC>2.0.CO;2.
- Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, 2007: *Numerical Recipes*. 3rd ed., Cambridge University Press, 1235 pp.
- Prill, F., 2020: *DWD ICON Tools Documentation*. Deutscher Wetterdienst (DWD), [dwd_icon_tools/doc/icontools_doc.pdf](https://dwd-icon-tools/doc/icontools_doc.pdf).
- Prill, F., 2023: The icon community interface. URL <https://www.nat-esm.de/services/trainings/techworkshop2>, natESM training "Software-engineering aspects of composed Earth System Models", Hamburg, 14–15 Nov 2023.
- Pruppacher, H. R., and J. D. Klett, 1998: *Microphysics of Clouds and Precipitation*. 2nd ed., Kluwer Academic Publishers.
- Randall, D. A., 2017: A survey of time-differencing schemes for the oscillation and decay equations. *An Introduction to Numerical Modelling of the Atmosphere*, Colorado State University, 63–104, URL http://kiwi.atmos.colostate.edu/group/dave/at604pdf/AT604_LaTeX_Book.pdf.
- Raschendorfer, M., 2001: The new turbulence parameterization of LM. *COSMO News Letter No. 1*, Consortium for Small-Scale Modelling, 89–97, URL <http://www.cosmo-model.org>.
- Rast, S., 2017: Using and programming ICON – a first introduction. Max Planck Institute for Meteorology, URL https://code.mpimet.mpg.de/attachments/download/15898/icon_lecture_2016.pdf, course at Hamburg University 2017.
- Reick, C. H., V. Gayler, D. Goll, S. Hagemann, M. Heidkamp, and J. E. M. S. Nabel, 2021: JSBACH 3 - The land component of the MPI Earth System Model: documentation of version 3.2. Reports on earth system science, Max Planck Institute for Meteorology, 287 pp. doi:10.17617/2.3279802, URL <https://hdl.handle.net/21.11116/0000-0008-098B-2>.
- Reinert, D., 2020: The Tracer Transport Module Part I: A Mass Consistent Finite Volume Approach with Fractional Steps. Reports on icon, Deutscher Wetterdienst, 19 pp. doi:10.5676/DWD_pub/nwv/icon_005, URL https://www.dwd.de/DE/leistungen/reports_on_icon/reports_on_icon.html.

- Reinert, D., 2021: The Tracer Transport Module Part II: Description and Validation of the Vertical Transport Operator. Reports on icon, Deutscher Wetterdienst, 42 pp. doi:10.5676/DWD_pub/nwv/icon_007, URL https://www.dwd.de/DE/leistungen/reports_on_icon/reports_on_icon.html.
- Reinert, D., and Coauthors, 2024: *DWD Database Reference for the Global and Regional ICON and ICON-EPS Forecasting System*. Deutscher Wetterdienst (DWD), URL https://www.dwd.de/SharedDocs/downloads/DE/modelldokumentationen/nwv/icon/icon_dbbeschr_aktuell.html.
- Rieger, D., M. Köhler, R. J. Hogan, S. A. K. Schäfer, A. Seifert, A. de Lozar, and G. Zängl, 2019: ecRad in ICON - Details on the Implementation and First Results. Reports on ICON 4, Deutscher Wetterdienst. doi:10.5676/DWD_pub/nwv/icon_004.
- Rieger, D., and Coauthors, 2015: ICON-ART 1.0 - a new online-coupled model system from the global to regional scale. *Geosci. Model Dev.*, **8** (6), 1659–1676, doi:10.5194/gmd-8-1659-2015.
- Rood, R. B., 2010: A Perspective on the Role of the Dynamical Core in the Development of Weather and Climate Models. *Numerical Techniques for Global Atmospheric Models*, P. Lauritzen, C. Jablonowski, M. A. Taylor, and R. D. Nair, Eds., Springer.
- Rotta, J., 1951a: Statistische Theorie nichthomogener Turbulenz. *J. Z. Physik*, **129** (6), 547–572, doi:10.1007/BF01330059.
- Rotta, J., 1951b: Statistische Theorie nichthomogener Turbulenz. *J. Z. Physik*, **131** (1), 51–77, doi:10.1007/BF01329645.
- Ruppert, T., 2007: Diplomarbeit: Vector field reconstruction by radial basis functions. M.S. thesis, Department of Mathematics, Technical University Darmstadt.
- Satoh, M., 2002: Conservative scheme for the compressible nonhydrostatic models with the horizontally explicit and vertically implicit time integration scheme. *Mon. Weather Rev.*, **130**, 1227–1245.
- Schär, C., D. Leuenberger, O. Fuhrer, D. Lüthi, and C. Girard, 2002: A new terrain-following vertical coordinate formulation for atmospheric prediction models. *Mon. Weather Rev.*, **130**, 2459–2480.
- Schrodin, R., and E. Heise, 2001: The Multi-Layer Version of the DWD Soil Model TERRA_LM. Tech. Rep. 2, Consortium for Small-Scale Modelling, 1–16 pp. URL <http://www.cosmo-model.org>.
- Schröter, J., and Coauthors, 2018: ICON-ART 2.1: a flexible tracer framework and its application for composition studies in numerical weather forecasting and climate simulations. *Geosci. Model Dev.*, **11** (10), 4043–4068, doi:10.5194/gmd-11-4043-2018.
- Schulz, J.-P., 2006: The new Lokal-Model LME of the German Weather Service. *COSMO News Letter No. 6*, Consortium for Small-Scale Modelling, 210–212, URL <http://www.cosmo-model.org>.

- Schulz, J.-P., 2008: Introducing sub-grid scale orographic effects in the cosmo model. *COSMO News Letter No. 9*, Consortium for Small-Scale Modelling, 29–36, URL <http://www.cosmo-model.org>.
- Schulz, J.-P., G. Vogel, C. Becker, S. Kothe, U. Rummel, and B. Ahrens, 2016: Evaluation of the ground heat flux simulated by a multi-layer land surface scheme using high-quality observations at grass land and bare soil. *Meteorol. Z.*, **25** (5), 607–620, doi:10.1127/metz/2016/0537.
- Schulzweida, U., 2024: *Climate Data Operators (CDO)*. Max Planck Institute for Meteorology, Hamburg, URL <https://code.mpimet.mpg.de/projects/cdo>.
- Scinocca, J. F., 2003: An accurate spectral nonorographic gravity wave drag parameterization for general circulation models. *J. Atmos. Sci.*, **60** (4), 667–682, doi:10.1175/1520-0469(2003)060<0667:AASNGW>2.0.CO;2.
- Seifert, A., 2008: A revised cloud microphysical parameterization for COSMO-LME. *COSMO News Letter No. 7, Proceedings from the 8th COSMO General Meeting in Bucharest, 2006*, Consortium for Small-Scale Modelling, 25–28, URL <http://www.cosmo-model.org>.
- Seifert, A., and K. D. Beheng, 2006: A two-moment cloud microphysics parameterization for mixed-phase clouds. Part 1: Model description. *Meteorol. Atmos. Phys.*, **92** (1), 45–66, doi:10.1007/s00703-005-0112-4.
- Siebert, H., and R. A. Shaw, 2017: Supersaturation fluctuations during the early stage of cumulus formation. *Journal of the Atmospheric Sciences*, **74** (3), 975–988, doi:10.1175/JAS-D-16-0180.1.
- Simmons, A., and D. Burridge, 1981: An energy and angular-momentum conserving finite-difference scheme and hybrid vertical coordinates. *Mon. Weather Rev.*, **109**, 758–766.
- Skamarock, W., J. B. Klemp, M. G. Duda, L. D. Fowler, S. Park, and T. D. Ringler, 2012: A Multiscale Nonhydrostatic Atmospheric Model Using Centroidal Voronoi Tessellations and C-Grid Staggering. *Mon. Weather Rev.*, **140**, 3090–3105, doi:10.1175/MWR-D-11-00215.1.
- Skamarock, W. C., and J. B. Klemp, 2008: A time-split nonhydrostatic atmospheric model for weather research and forecasting applications. *J. Comput. Phys.*, **227**, 3465–3485.
- Skamarock, W. C., and M. Menchaca, 2010: Conservative transport schemes for spherical geodesic grids: High-order reconstructions for forward-in-time schemes. *Mon. Weather Rev.*, **138**, 4497–4508.
- Skamarock, W. C., and Coauthors, 2019: A description of the advanced research WRF model version 4. No. ncar/tn-556+str, National Center For Atmospheric Research, Boulder, CO. doi:10.5065/1dfh-6p97.
- Smagorinsky, J., 1963: General Circulation Experiments with the Primitive Equations. *Mon. Weather Rev.*, **91**, 99, doi:10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

- Smiatek, G., J. Helmert, and E.-M. Gerstner, 2016: Impact of land use and soil data specifications on COSMO-CLM simulations in the CORDEX-MED area. *Meteorol. Z.*, **25** (2), 215–230, doi:10.1127/metz/2015/0594.
- Smiatek, G., B. Rockel, and U. Schättler, 2008: Time invariant data preprocessor for the climate version of the COSMO model (COSMO-CLM). *Meteorol. Z.*, **17** (4), 395–405, doi:10.1127/0941-2948/2008/0302.
- Snyder, J., 1987: *Map Projections – a Working Manual*. No. 1395, U. S. Geological Survey professional paper, U. S. Government Printing Office.
- Sommeria, G., and J. W. Deardorff, 1977: Subgrid-Scale Condensation in Models of Non-precipitating Clouds. *J. Atmos. Sci.*, **34** (2), 344–355, doi:10.1175/1520-0469(1977)034<0344:SSCIMO>2.0.CO;2.
- Straka, J. M., R. B. Wilhelmson, and K. K. Droegemeier, 1993: Numerical solutions of a non-linear density current: A benchmark solution and comparisons. *Int. J. Numer. Methods Fluids*, **17**, 1–22.
- Strang, G., 1968: On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, **5** (3), 506–517, doi:10.1137/0705041.
- Strassmann, D., D. Reinert, and A. Dipankar, 2025: Truncation error analysis of the Piecewise Parabolic Method. *Q. J. R. Meteorol. Soc.*, accepted.
- Takacs, L. L., M. J. Suárez, and R. Todling, 2016: Maintaining atmospheric mass and water balance in reanalyses. *Q. J. R. Meteorol. Soc.*, **142** (697), 1565–1573, doi:10.1002/qj.2763, URL <https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.2763>, <https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.2763>.
- Thuburn, J., 2008: Some conservation issues for the dynamical cores of NWP and climate models. *Journal of Computational Physics*, **227** (7), 3715–3730, doi:10.1016/j.jcp.2006.08.016, URL <https://www.sciencedirect.com/science/article/pii/S0021999106003755>.
- Tibaldi, S., 1986: Envelope orography and maintenance of quasi-stationary waves in the ECMWF model. *Adv. Geophys.*, **29**, 339–374.
- Tiedtke, M., 1989: A comprehensive mass flux scheme for cumulus parameterization in large-scale models. *Mon. Weather Rev.*, **117** (8), 1779–1800, doi:10.1175/1520-0493(1989)117<1779:ACMFSF>2.0.CO;2.
- Tomita, H., M. Satoh, and K. Goto, 2002: An optimization of the icosahedral grid modified by spring dynamics. *J. Comput. Phys.*, **183** (1), 307–331, doi:10.1006/jcph.2002.7193.
- Ullrich, P. A., and Coauthors, 2017: DCMIP2016: a review of non-hydrostatic dynamical core design and intercomparison of participating models. *Geosci. Model Dev.*, **10** (12), 4477–4509, doi:10.5194/gmd-10-4477-2017.
- Wacker, U., and F. Herbert, 2003: Continuity equations as expressions for local balances of masses in cloudy air. *Tellus A*, **55**, 247–254.
- Wallace, J., S. Tibaldi, and A. Simmons, 1983: Reduction of systematic forecast errors in the ECMWF model through the introduction of an envelope orography. *Q. J. R. Meteorol. Soc.*, **109**, 683–717.

- Wan, H., and Coauthors, 2013: The ICON-1.2 hydrostatic atmospheric dynamical core on triangular grids – Part 1: Formulation and performance of the baseline version. *Geosci. Model Dev.*, **6** (3), 735–763, doi:10.5194/gmd-6-735-2013.
- Warner, C. D., and M. E. McIntyre, 1996: On the propagation and dissipation of gravity wave spectra through a realistic middle atmosphere. *J. Atmos. Sci.*, **53** (22), 3213–3235, doi:10.1175/1520-0469(1996)053<3213:OTPAO>2.0.CO;2.
- Yeh, K.-S., 2007: The streamline subgrid integration method: I. quasi-monotonic second order transport schemes. *J. Comput. Phys.*, **225**, 1632–1652.
- Zalesak, S. T., 1979: Fully multidimensional flux-corrected transport algorithms for fluid. *J. Comput. Phys.*, **31**, 335–362.
- Zängl, G., 2012: Extending the Numerical Stability Limit of Terrain-Following Coordinate Models over Steep Slopes. *Mon. Weather Rev.*, **140**, 3722–3733, doi:10.1175/MWR-D-12-00049.1.
- Zängl, G., D. Reinert, and F. Prill, 2022: Grid Refinement in ICON v2.6.4. *Geosci. Model Dev.*, **15** (18), 7153–7176, doi:10.5194/gmd-15-7153-2022.
- Zängl, G., D. Reinert, P. Ripodas, and M. Baldauf, 2015: The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core. *Q. J. R. Meteorol. Soc.*, **141**, 563–579.
- Zarzycki, C., M. N. Levy, C. Jablonowski, J. R. Overfelt, M. A. Taylor, and P. Ullrich, 2014: Aquaplanet experiments using CAM’s variable resolution dynamical core. *J. Clim.*, **27**, 5481–5503, doi:10.1175/JCLI-D-14-00004.1.
- Zdunkowski, W., and A. Bott, 2003: *Dynamics of the atmosphere: A course in theoretical meteorology*. 1st ed., Cambridge University Press, 719 pp.
- Zerroukat, M., N. Wood, and A. Staniforth, 2002: SLICE: A Semi-Lagrangian Inherently Conserving and Efficient scheme for transport problems. *Q. J. R. Meteorol. Soc.*, **128**, 2801–2820, doi:10.1256/qj.02.69.
- Zerroukat, M., N. Wood, and A. Staniforth, 2005: A monotonic and positive-definite filter for Semi-Lagrangian Inherently Conserving and Efficient (SLICE) scheme. *Q. J. R. Meteorol. Soc.*, **131**, 2923–2936.
- Zerroukat, M., N. Wood, and A. Staniforth, 2006: The Parabolic Spline (PSM) Method for conservative transport problems. *Int. J. Numer. Meth. Fluids*, **51**, 1297–1318.

Index of Namelist Parameters

The following index contains only namelist parameters covered by this tutorial. Please take a look at the document

`icon/doc/Namelist_overview/Namelist_overview.pdf`

for a complete list of available namelist parameters for the ICON model.

- `initicon_nml` (Namelist), [152](#)
- `albedo_type`, [54](#)
- `ana_varnames_map_file`, [151](#)
- `bdy_indexing_depth`, [25](#), [25](#)
- `bub_amp`, [145](#), [146](#)
- `bub_hor_width`, [145](#), [146](#)
- `bub_ver_width`, [145](#), [146](#)
- `bubctr_z`, [145](#), [146](#)
- `calendar`, [251](#)
- `checkpointTimeIntVal`, [176](#), [176](#), [177](#)
- `cldopt_filename`, [52](#)
- `comin_nml` (Namelist), [207](#)
- `const_z0`, [146](#)
- `damp_height`, [140](#), [161](#), [167](#), [167](#), [168](#), [168](#)
- `diffusion_nml` (Namelist), [136](#), [140](#), [146](#)
- `dom`, [170](#), [171](#)
- `dt_checkpoint`, [176](#), [177](#), [177](#)
- `dt_conv`, [91](#), [92](#)
- `dt_gwd`, [91](#)
- `dt_iau`, [153](#), [237](#), [240](#)
- `dt_rad`, [91](#), [92](#), [99](#)
- `dt_restart`, [177](#), [177](#)
- `dt_shift`, [153](#), [155](#), [237](#), [240](#)
- `dt_sso`, [91](#)
- `dtime`, [90](#), [137](#), [140](#), [146](#), [148](#), [160](#), [176](#)
- `dtime_latbc`, [165](#)
- `dwdana_filename`, [152](#), [152–154](#)
- `dwdfg_filename`, [152](#), [152–154](#), [163](#)
- `dynamics_grid_filename`, [22](#), [121](#), [137](#), [140](#), [146](#), [149](#), [150](#), [196](#)
- `dynamics_nml` (Namelist), [136](#), [140](#), [146](#)
- `dynamics_parent_grid_id`, [21](#), [22](#), [22](#)
- `ecrad_data_path`, [52](#), [96](#)
- `ecrad_isolver`, [189](#), [189](#)
- `end_datetime_string`, [148](#), [148](#)
- `end_time`, [148](#), [155](#), [155](#), [176](#)
- `exner_expol`, [74](#), [140](#)
- `experimentStartDate`, [148](#), [155](#), [165](#), [171](#), [176](#)
- `experimentStopDate`, [148](#), [176](#)
- `extpar_filename`, [149](#), [149–151](#)
- `extpar_nml` (Namelist), [51](#), [54](#), [55](#), [136](#), [140](#), [146](#), [149](#), [151](#), [243](#)
- `extpar_varnames_map_file`, [151](#)
- `fbk_relax_timescale`, [127](#)
- `filename_format`, [170](#), [170](#), [171](#)
- `filetype`, [170](#), [170](#), [253](#)
- `flat_height`, [69](#), [129](#)
- `frlake_thrhd`, [116](#)
- `grid_nml`, [68](#)
- `grid_nml` (Namelist), [22](#), [70](#), [121](#), [125](#), [133](#), [137](#), [140](#), [146](#), [148](#), [149](#), [155](#), [157](#), [176](#), [250](#)
- `gridgen_nml` (Namelist), [25](#), [26](#)
- `gridref_nml` (Namelist), [127](#)
- `h_levels`, [172](#)
- `hbot_qvsubstep`, [86](#), [118](#), [118](#), [119](#)
- `hdiff_efdt_ratio`, [140](#), [146](#)
- `hdiff_order`, [140](#), [146](#)
- `hdiff_smag_fac`, [146](#)

hl_varlist, [172](#), [172](#)
 htop_moist_proc, [88](#), [118](#), [118](#), [119](#)
 i_levels, [172](#)
 icapdcycle, [107](#), [107](#)
 icld_overlap, [98](#)
 iforcing, [136](#), [136](#), [140](#), [146](#), [149](#)
 ifs2icon_filename, [152](#), [154](#), [162](#), [163](#)
 igradp_method, [65](#)
 ihadv_tracer, [86](#), [86–88](#), [119](#)
 iice_scatter, [98](#)
 il_varlist, [172](#), [172](#)
 iliquid_scatter, [98](#)
 in_filename, [50](#)
 in_grid_filename, [50](#)
 in_mask_below, [42](#)
 in_mask_threshold, [42](#)
 in_type, [50](#)
 inextra_2d, [204](#)
 inextra_3d, [204](#)
 ini_datetime_string, [148](#), [148](#), [152–155](#),
 [165](#), [171](#)
 init_mode, [38](#), [150](#), [150](#), [152–154](#), [162](#), [162](#),
 [163](#), [251](#)
 initicon_nml (Namelist), [114](#), [116](#), [150–155](#),
 [162](#), [163](#), [237](#), [240](#)
 input_field_nml (Namelist), [37](#), [40–43](#), [50](#)
 interpol_nml (Namelist), [86](#), [129](#), [159](#), [173](#)
 inwp_cldcover, [97](#)
 inwp_convection, [97](#), [106](#), [167](#), [167](#)
 inwp_gsep, [85](#), [97](#), [97](#), [104](#), [105](#), [105](#), [106](#)
 inwp_gwd, [97](#)
 inwp_radiation, [96](#), [97](#), [189](#)
 inwp_sso, [97](#)
 inwp_surface, [97](#)
 inwp_turb, [97](#), [110](#), [146](#)
 io_nml, [251](#)
 io_nml (Namelist), [151](#), [170](#), [177](#), [178](#), [204](#),
 [253](#)
 irad_aero, [54](#), [252](#), [253](#)
 irad_o3, [253](#)
 is_dry_cbl, [146](#)
 is_plane_torus, [146](#)
 isolrad, [253](#)
 isrfc_type, [146](#)
 iterate_iau, [240](#)
 itopo, [136](#), [136](#), [140](#), [146](#), [149](#)
 itype_canopy, [243](#), [247](#)
 itype_hlimit, [86](#), [88](#)
 itype_latbc, [158](#), [165](#)
 itype_lwemiss, [54](#)
 itype_sher, [111](#)
 itype_snowevap, [242](#), [243](#), [245](#), [247](#)
 itype_t_diffu, [140](#)
 itype_trvg, [242](#), [245](#), [247](#)
 itype_vegetation_cycle, [55](#), [243](#), [248](#), [248](#)
 itype_vlimit, [82](#), [86](#), [88](#)
 itype_vn_diffu, [140](#)
 itype_z0, [55](#)
 ivadv_tracer, [86](#), [87](#), [88](#)
 ivctype, [68](#), [69](#), [167](#)
 ivlimit_selective, [87](#), [87](#)
 jw_temp0, [138](#), [140](#)
 jw_u0, [138](#), [140](#)
 jw_up, [138](#), [140](#)
 Kh_ext, [146](#)
 Km_ext, [146](#)
 l_limited_area, [157](#), [160](#), [250](#)
 latbc_boundary_grid, [161](#), [166](#)
 latbc_filename, [165](#), [165](#)
 latbc_path, [165](#), [165](#)
 latbc_varnames_map_file, [151](#), [164](#), [166](#)
 latm_above_top, [161](#)
 layer_thickness, [68](#)
 lconst_z0, [146](#)
 lcoriolis, [136](#), [140](#), [146](#)
 ldynamics, [135](#), [140](#), [146](#), [149](#)
 les_nml (Namelist), [111](#), [146](#)
 lfeedback, [125](#)
 lgrayzone_deepconv, [107](#)
 limarea_nml (Namelist), [151](#), [158](#), [159](#), [161](#),
 [165](#), [166](#)
 linside_domain_test, [42](#)
 linvert_dict, [151](#)
 llake, [97](#), [116](#), [116](#)
 lmeteogram_enabled, [178](#)
 lnd_nml (Namelist), [52](#), [54](#), [55](#), [113](#), [116](#),
 [118](#), [242–247](#), [252](#)
 lprog_albsi, [242](#), [244](#), [246](#)
 lread_ana, [152](#), [152](#), [153](#), [163](#), [163](#)
 lredgrid_phys, [133](#)
 lrestart, [176](#), [177](#), [251](#)
 lrtm_filename, [52](#)
 lseaice, [97](#), [118](#)

- lshallowconv_only, [107](#), [107](#), [167](#), [167](#)
- lsnowtile, [113](#)
- lsq_high_ord, [86](#)
- ltestcase, [68](#), [135](#), [140](#), [146](#), [148](#)
- ltile_coldstart, [114](#)
- ltile_init, [114](#)
- ltimer, [185](#)
- ltkeshs, [111](#), [111](#)
- ltransport, [85](#), [136](#), [140](#), [146](#), [149](#)
- ltransport=.TRUE., [139](#)
- lvert_nest, [67](#), [129](#)
- lwrite_parent, [26](#), [133](#)
- m_levels, [170](#)
- master_nml (Namelist), [150](#), [152](#), [171](#), [176](#), [177](#), [251](#)
- master_time_control_nml (Namelist), [148](#), [176](#), [177](#), [251](#)
- max_nudge_coeff_thermdyn, [159](#)
- max_nudge_coeff_vn, [159](#)
- max_refin_c_ctrl, [48](#), [48](#)
- max_time_stamps, [179](#)
- merge_domain, [132](#)
- meteoqram_output_nml (Namelist), [178](#), [179](#)
- min_lay_thckn, [69](#), [69](#), [146](#)
- min_refin_c_ctrl, [48](#)
- ml_varlist, [170](#), [170](#), [172](#)
- model_base_dir, [150](#), [152](#), [171](#)
- modelTimeStep, [148](#)
- msg_level, [136](#)
- n0_mtgrm, [178](#)
- n_flat_lev, [68](#)
- n_iter_smooth_topo, [51](#)
- nadv_substeps, [86](#)
- nblocks_c, [184](#), [188](#), [188](#)
- nblocks_sub, [188](#), [188](#)
- ncstorage_file, [50](#)
- ndyn_substeps, [90](#), [90](#), [91](#), [123](#)
- netcdf_dict, [151](#)
- nh_brunt_vais, [145](#), [146](#)
- nh_test_name, [138](#), [140](#), [146](#)
- nh_testcase_nml (Namelist), [135](#), [138](#), [140](#), [144–146](#)
- nh_testcases_nml, [68](#)
- nh_u0, [145](#)
- ninc_mtgrm, [178](#)
- nonhydrostatic_nml (Namelist), [65](#), [68](#), [69](#), [72–74](#), [86](#), [88](#), [90](#), [118](#), [123](#), [136](#), [140](#), [161](#), [167](#)
- north_pole, [174](#), [175](#)
- nproma, [184](#), [188](#), [188](#), [192](#), [193](#)
- nproma_sub, [188](#), [188](#)
- nsteps, [137](#), [140](#), [146](#), [148](#), [176](#)
- ntiles, [52](#), [54](#), [55](#), [113](#), [113](#), [114](#)
- ntracer, [85](#), [140](#)
- nudge_efold_width, [129](#), [159](#)
- nudge_hydro_pres, [159](#)
- nudge_max_coeff, [129](#), [159](#)
- nudge_start_height, [159](#)
- nudge_type, [48](#), [158](#), [160](#), [161](#)
- nudge_zone_width, [129](#), [159](#)
- nudging_nml (Namelist), [48](#), [158–161](#)
- num_io_procs, [175](#), [176](#), [182](#)
- num_lev, [66](#), [69](#), [129](#), [137](#), [140](#), [146](#), [168](#)
- num_prefetch_proc, [166](#), [166](#)
- num_restart_proc, [178](#), [251](#)
- num_restart_procs, [177](#), [182](#)
- nwp_phy_nml (Namelist), [52](#), [55](#), [91](#), [96](#), [97](#), [105–107](#), [110](#), [136](#), [146](#), [161](#), [167](#)
- nwp_tuning_nml (Namelist), [99](#), [101](#), [105](#), [167](#)
- out_grid_filename, [50](#)
- out_mask_below, [42](#)
- out_mask_filename, [42](#)
- out_mask_threshold, [42](#)
- out_type, [50](#)
- output, [169](#), [178](#)
- output_bounds, [170](#)
- output_filename, [169](#), [170](#)
- output_nml (Namelist), [137](#), [140](#), [169–172](#), [174–176](#), [253](#), [268](#)
- output_nml_dict, [151](#), [170](#)
- p_levels, [172](#)
- parallel_nml (Namelist), [166](#), [176–178](#), [182](#), [184](#), [188](#), [192](#), [251](#)
- pe_placement_ml, [176](#)
- pl_varlist, [172](#), [172](#)
- precip_interval, [253](#)
- proc0_shift, [184](#), [184](#), [189](#)
- radiation_grid_filename, [133](#), [149](#)
- radiation_nml (Namelist), [52](#), [54](#), [96](#), [98](#), [189](#), [252](#), [253](#)

rat_sea, [167](#), [167](#)
 rayleigh_coeff, [140](#)
 rbf_scale, [174](#)
 reg_lat_def, [171](#)
 reg_lon_def, [171](#)
 remap, [170](#), [171](#), [253](#)
 remap_nml (Namelist), [42](#), [50](#)
 restart_filename, [177](#), [177](#)
 restart_write_mode, [178](#), [178](#)
 restartTimeIntVal, [176](#), [177](#)
 rhotheta_offctr, [73](#)
 run_nml (Namelist), [66–68](#), [85](#), [129](#), [135–137](#), [139](#), [140](#), [146](#), [148](#), [149](#), [169](#), [176](#), [177](#), [185](#)

 sleeve_nml, [70](#)
 sleeve_nml (Namelist), [69](#), [129](#), [137](#), [140](#), [146](#), [159](#), [167](#)
 smag_coeff_type, [146](#)
 smag_constant, [111](#)
 sst_td_filename, [252](#)
 sstice_mode, [55](#), [252](#)
 start_time, [148](#), [155](#), [155](#), [176](#)
 stationlist_tot, [178](#)
 steps_per_file, [171](#)
 stream_partitions_ml, [175](#), [176](#), [176](#)
 support_baryctr_intp, [173](#)

 time_nml, [251](#)
 time_nml (Namelist), [148](#), [177](#)
 timers_level, [185](#)
 top_height, [69](#), [140](#), [146](#), [159](#), [167](#), [167](#), [168](#), [168](#)
 tracer_inidist_list, [140](#), [141](#)
 tracer_names, [141](#), [141](#)
 transport_nml (Namelist), [82](#), [85–87](#), [119](#), [136](#), [141](#), [149](#)
 tune_gfluxlaun, [99](#)
 tune_gfrcrit, [101](#)
 tune_gkdrag, [101](#), [101](#)
 tune_gkwake, [101](#), [101](#)
 tune_grcrit, [101](#)
 tune_supsat_limfac, [105](#)
 tune_zvz0i, [166](#), [167](#)
 turb_prandtl, [111](#)
 turbdiff_nml (Namelist), [110](#), [146](#), [167](#)

 ufri, [146](#)
 use_lakeiceana, [116](#)

 var_in_mask, [41](#), [42](#), [42](#)
 var_list, [179](#)
 var_out_mask, [42](#)
 vct_filename, [68](#), [70](#)
 veladv_offctr, [73](#)
 vwind_offctr, [72](#), [72](#), [140](#)

 zprefix, [179](#)