

ICON Model Tutorial

April 2018



Working with the ICON Model

Practical Exercises for NWP Mode and ICON-ART

For NWP Mode: D. Reinert, F. Prill, G. Zängl

For ICON-ART: D. Rieger, J. Schröter, J. Förstner, S. Werchner,
M. Weimer, R. Ruhnke, B. Vogel

Acknowledgments

Many people contributed to this manuscript.

The section on ICON physics was partly provided by J. Helmert (land-soil model TERRA, Section 3.7.8), D. Klocke (convection parameterization, Section 3.7.3), M. Köhler (cloud-cover parameterization and turbulence, Sections 3.7.4, 3.7.5), D. Mironov (summary of sea-ice and lake model, Sections 3.7.7, 3.7.6), M. Raschendorfer (turbulence, Section 3.7.5), and A. Seifert (grid-scale microphysics parameterization, Section 3.7.2), DWD Physical Processes Division.

Section 9.3.3 (Visualization with R) has been contributed by J. Förstner, DWD Physical Processes Division.

Chapter 10 was provided by the Institute of Meteorology and Climate Research at the Karlsruhe Institute of Technology (KIT).

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

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. 8.

Contents

0. Preface	1
0.1. How This Document Is Organized	1
0.2. How to Obtain a Copy of the ICON Model Code	2
0.3. Further Documentation	3
1. Installation of the ICON Model Package	5
1.1. The ICON Model Package	5
1.1.1. Directory Layout	6
1.1.2. Libraries Needed for Data Input and Output	7
1.2. Configuring and Compiling the Model Code	10
1.2.1. Computer Platforms	10
1.2.2. Configuring and Compiling	11
1.3. The DWD ICON Tools	12
1.3.1. General Overview	12
1.3.2. Configuring and Compiling 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. Download of Predefined Grids	20
2.1.3. Grid Generator: Invocation from the Command Line	21
2.1.4. Grid Generator: Invocation via the Web Interface	23
2.1.5. Which Grid File is Related to My Simulation Data?	26
2.2. Initial Conditions	27
2.2.1. Obtaining DWD Initial Data	27
2.2.2. Obtaining ECMWF IFS Initial Data	34
2.2.3. Remapping Initial Data to Your Target Grid	35
2.3. Preparing Boundary Data for ICON-LAM	37
2.4. External Data Files	41
2.4.1. ExtPar Products	41
2.4.2. Additional Information for Surface Tiles	41
2.4.3. Parameter Files for Radiation	42
3. Model Description	47
3.1. Governing Equations	47
3.2. Simplifying Assumptions in the Recent Model Version	49
3.3. Vertical Coordinates	51
3.3.1. Terrain-following Hybrid Gal-Chen Coordinate	53
3.3.2. SLEVE Coordinate	54
3.4. ICON Time-Stepping	55

3.5.	Tracer Transport	57
3.5.1.	Directional Splitting	58
3.5.2.	Horizontal Transport	59
3.5.3.	Vertical Transport	61
3.5.4.	Reduced Calling Frequency	63
3.5.5.	Some Practical Advice	63
3.6.	Variable Resolution Modeling	66
3.6.1.	Parent-Child Coupling	66
3.6.2.	Processing Sequence	72
3.6.3.	Vertical Nesting	74
3.7.	ICON NWP-Physics in a Nutshell	75
3.7.1.	Radiation	75
3.7.2.	Cloud Microphysics	75
3.7.3.	Cumulus Convection	77
3.7.4.	Cloud Cover	78
3.7.5.	Turbulent Diffusion	79
3.7.6.	Lake Parameterization Scheme FLake	83
3.7.7.	Sea-Ice Parameterization Scheme	85
3.7.8.	Land-Soil Model TERRA	86
3.7.9.	Details of ICON's Physics-Dynamics Coupling	89
3.7.10.	Isobaric vs. Isochoric Coupling Strategies	90
3.7.11.	Reduced Model Top for Moist Physics	92
3.8.	Reduced Radiation Grid	93
4.	Running Idealized Test Cases	95
4.1.	Namelist Input for the ICON Model	95
4.2.	Jablonowski-Williamson Baroclinic Wave Test	95
4.2.1.	Main Switches for the Idealized Test Case	96
4.2.2.	Specifying the Computational Domain(s)	98
4.2.3.	Integration Time Step and Simulation Length	100
5.	Running Real Data Test Cases	101
5.1.	Model Initialization	101
5.1.1.	Basic Settings for Running Real Data Runs	101
5.1.2.	Starting from Uninitialized DWD Analysis	104
5.1.3.	Starting from Uninitialized DWD Analysis with IAU	105
5.1.4.	Starting from Initialized DWD Analysis	107
5.1.5.	Starting from IFS Analysis	107
5.2.	Starting or Terminating Nested Domains at Runtime	108
6.	Running ICON-LAM	109
6.1.	Limited Area Mode vs. Nested Setups	109
6.2.	Nudging in the Boundary Region	110
6.3.	Model Initialization	111
6.4.	Reading Lateral Boundary Data	113
6.4.1.	Naming Scheme for Lateral Boundary Data	114
6.4.2.	Pre-Fetching of Boundary Data (Mandatory)	115

7. Parallelization and Output	117
7.1. Settings for the Model Output	117
7.1.1. Output Rank Assignment	119
7.1.2. Output on Regular Grids and Vertical Interpolation	120
7.2. Checkpointing and Restart	120
7.3. Parallelization and Performance Aspects	122
7.3.1. Settings for Parallel Execution	122
7.3.2. Mixed Single/ Double Precision in ICON	125
7.3.3. Bit-Reproducibility	125
7.3.4. Basic Performance Measurement	126
8. Programming ICON	129
8.1. Representation of 2D and 3D Fields	129
8.2. Data Structures	132
8.2.1. Description of the Model Domain	132
8.2.2. Date and Time Variables	133
8.2.3. Data Structures for Physics and Dynamics Variables	134
8.2.4. Parallel Communication	135
8.3. NWP Call Tree	135
8.4. Implementing Own Diagnostics	136
9. Post-Processing and Visualization	141
9.1. Retrieving Data Set Information	141
9.1.1. ncdump	141
9.1.2. CDO – Climate Data Operators	142
9.2. Plotting Data Sets on Regular Grids	143
9.2.1. Ncview	143
9.2.2. Remapping to Regular Grids with CDO	143
9.3. Plotting Data Sets on the Triangular Grid	144
9.3.1. NCL – NCAR Command Language	144
9.3.2. NCL Step-by-step Tutorial	146
9.3.3. Visualization with R	151
9.3.4. GMT – Generic Mapping Tools	154
10. Running ICON-ART	157
10.1. General Remarks	157
10.2. ART Directory Structure	157
10.3. Installation	159
10.4. Configuration of an ART Job	160
10.4.1. Recommended ICON Namelist Settings	160
10.4.2. ART Namelists	160
10.4.3. Tracer Definition with XML Files	163
10.4.4. Modes Definition with XML Files	164
10.4.5. Point Source Module: pntSrc	165
10.4.6. Volcanic Ash Control	166
10.5. Output	167

11. ICON's Data Assimilation System	169
11.1. Data Assimilation	169
11.1.1. Variational Data Assimilation	169
11.1.2. Ensemble Kalman Filter	171
11.1.3. Hybrid Data Assimilation	171
11.1.4. Surface Analysis	172
11.2. Assimilation Cycle at DWD	172
Appendix A. The Computer System at DWD	175
Appendix B. Troubleshooting	177
Appendix C. Table of ICON Output Variables	181
Appendix D. Exercises	191
D.1. Installation of the ICON Model Package	192
D.2. Necessary Input Data	192
D.4. Running Idealized Test Cases	196
D.5. Running Real Data Test Cases	199
D.6. Running ICON-LAM	206
D.8. Programming ICON	210
D.10. Running ICON-ART	212
Bibliography	215
Index of Namelist Parameters	221

0. Preface

The ICON (ICOsahedral Nonhydrostatic) modeling framework (Zängl et al., 2015) is a joint project between the Deutscher Wetterdienst (DWD) and the Max-Planck-Institute for Meteorology (MPI-M) for developing a unified next-generation global numerical weather prediction (NWP) and climate modeling system.

The main goals formulated in the initial phase of the collaboration are

- 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.
- applicability on a wide range of scales down to $\mathcal{O}(1km)$ and beyond (which of course requires a nonhydrostatic dynamical core).

The ICON modeling framework 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 13km and 90 vertical levels. Starting from July 21st, 2015, model simulations have been complemented by a nesting region over Europe. Finally, in January 2018, the global 40 member ICON-EPS (Ensemble Prediction System) was released for the operational service at DWD.


The model source code has been made available for scientific use under an institutional license since 2015.

0.1. How This Document Is Organized

Not all topics in this manuscript are covered during the 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 8 – 11.

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

At the end of the document a number of exercises is provided (see page 191). These exercises revisit the topics of the individual chapters and range from easy tests to the setup of complex forecast simulations.

To some extent this document can also be used as a reference manual. We refer to the index on page 221 for a quick look-up of namelist parameters.

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

The ICON model is distributed under an **institutional license**¹. To obtain a grant of license that must be signed and returned to the DWD, please contact icon@dwd.de.

Additionally, we have established the mailing list icon-community@mpimet.mpg.de to stay in touch with the ICON community. Please visit <https://listserv.gwdg.de/mailman/listinfo/icon-community> and subscribe to this list in order to receive announcements about new releases and features.

Data Services

DWD has made a number of model forecast data sets publicly available, mostly free of charge (with a retention of 48 h). This service has started in July 2017 and can be reached under <https://opendata.dwd.de/weather/icon>. See the content description under <https://www.dwd.de/EN/ourservices/opendata/opendata.html> for a list of spatial data sets.

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

Access to the **grid generator web service** (see Section 2.1.4)

<https://webservice.dwd.de/cgi-bin/spp1167/webservice.cgi>

via the account **icon-web** requires a password. To this end, please contact icon@dwd.de.

¹An individual licensing procedure has not yet been released by April 2018.

0.3. Further Documentation

The ICON model is accompanied by various other manuals and documentation.

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. 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](http://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.pdf](#)

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 \(2014\)](#).

Scientific documentation.

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

Recent information on ICON's hydrostatic dynamical core 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 extended modules for Aerosols and Reactive Trace gases (ART) are described in [Rieger et al. \(2015\)](#).

A description of the ocean component ICON-O within the ICON modeling system (which is not covered by this tutorial) can be found in [Korn \(2017\)](#).

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. Exercises are carried out on the supercomputers at DWD but the principal steps of the installation can directly be transferred to other systems.

1.1. The ICON Model Package

The source code for the ICON model package consists of the following three components:

- **The ICOSahedral Nonhydrostatic model (ICON)**
The ICON code that is used for this tutorial has been derived from the ICON release v2.3.00 with additional enhancements for ICON-LAM (state *February 2018*). It also contains the ocean model developed at MPI-M which is, however, not covered by this tutorial.
- **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 to enable the simulation of gases, aerosol particles and related feedback processes in the atmosphere. The module is provided by the Karlsruhe Institute of Technology (KIT) and requires a separate license (see Section 10.1).
- **DWD ICON Tools**
The ICON Tools are a set of command-line tools for remapping, extracting and querying ICON data files. They are based on a common library and written in Fortran 90/95 and Fortran 2003.

The versioning of ICON is a bit complex and reflects the parallel development in several “flavors” like “atmosphere”, “ocean”, and several more. There are four important branches tagging versions that reached certain milestones: *icon-aes* (atmosphere in the Earth system, mainly echam physics in the atmosphere), *icon-nwp* (numerical weather prediction, mainly dynamics and physics of the LEM and NWP configurations of the atmospheric model), *icon-oes* (ocean in the Earth system), *icon-les* (land in the Earth system).

All tags contain all model components, but the latest tag of *icon-aes* may not contain the most recent developments of the ocean physics although these are already included into the latest *ocean-oes* tag. The release version integrates the stable components of all branches.

1.1.1. Directory Layout

Figure 1.1 shows a brief description of the directory structure of the ICON model and of the directories containing the test case data under the root tree.

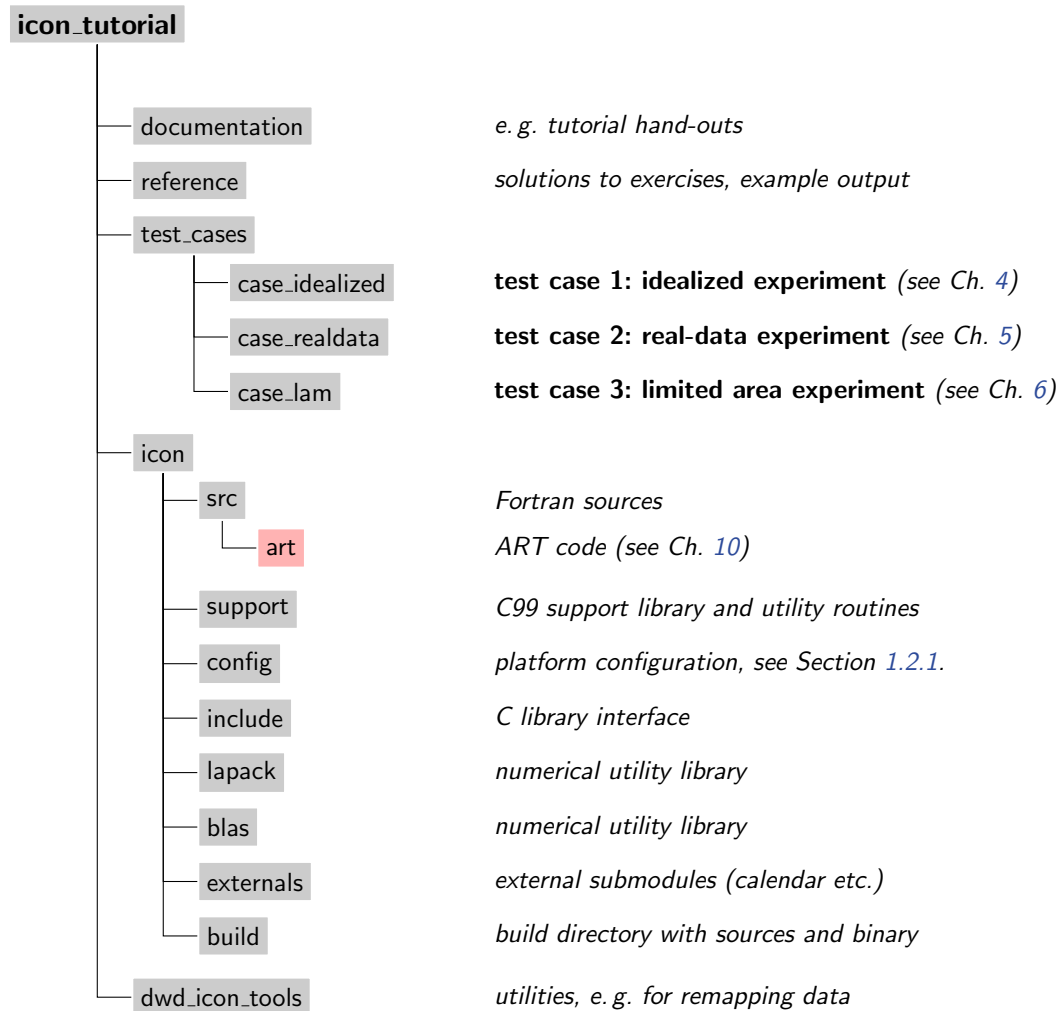


Figure 1.1.: Directory structure of the ICON model and of the directories containing the test case data under the root tree. Note that the ICON-ART source code modules are distributed in a separate tarball.

The ICON model code is located in the directory `icon`. The most important sub-directories are described in the following:

Sub-directory `build`

Within the `build` directory, a sub-directory with the name of your computer architecture is created during compilation. Within this sub-directory, a `bin` sub-directory containing the ICON binary `icon` and several other sub-directories containing the compiled module files are created.

Sub-directory config

Inside the `config` directory, different machine-dependent configurations are stored in configuration script files (see Section 1.2.1).

Sub-directory src

Within the `src` directory, the source code of ICON including the main program and ICON modules can be found. The modules are organized in several sub-directories:

The main program `icon.f90` can be found inside the sub-directory `src/drivers`. Additionally, this directory contains the modules for a hydrostatic and a 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 inside `src/atm_dyn_iconam` and the physical parameterizations inside `src/atm_phy_nwp`. Surface parameterizations can be found inside `src/lnd_phy_nwp`.

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 may be replaced by machine-dependent optimizations. However, please note that LAPACK and BLAS routines are *not* actively used by the *nonhydrostatic* model.

1.1.2. Libraries Needed for Data Input and Output

Especially for I/O tasks, the ICON model package requires external libraries. Two data formats are implemented in the package to read and write data from or to disk: GRIB and NetCDF.

- 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>.

To work with the formats described above the following libraries are utilized by the ICON model package. For this training course, the paths to access these libraries on the used computer system are already specified in the `Makefile`.

The Climate Data Interfaces (CDI) – `support/cdilib.c`

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. A condensed copy of the CDI is distributed together with the ICON model package. Note that the CDI are also used by the DWD ICON Tools.

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

The ECMWF GRIB-API/ecCodes package – `libgrib_api.a`, `libgrib_api_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 *GRIB-API/ecCodes* package uses the so-called key/value approach, which means that all the information contained in the GRIB message is retrieved through alphanumeric names. Indirect use of this GRIB-API/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 GRIB-API/ecCodes we refer to the ECMWF web page:

<https://software.ecmwf.int/wiki/display/GRIB/Home>



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

Please refer to the `README` for installing the GRIB-API/ecCodes libraries, which is done with a `configure` script. Check the following settings:

- The GRIB-API/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`.

After the configuration has finished, the GRIB-API/ecCodes library can be built with `make` and then `make install`.

GRIB Definition Files

An installation of the GRIB-API/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.

In contrast to the GRIB triplet, the short name, e.g., “t” for temperature, 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 `indicatorOfParameter=39`).

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 GRIB-API/ecCodes package can be obtained via

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

The new directory needs to be communicated to the GRIB-API/ecCodes package by setting the `GRIB_DEFINITION_PATH` environment variable (preceding the default definition files path).

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.

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 9.1.1).

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. 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 1.1.2 are present on your computer system. For convenience, the compiler version and the GRIB-API/ecCodes version are documented in the log output of each model run.

1.2.1. Computer Platforms

For a small number of HPC platforms settings are provided with the code, for example

Cray XC 40 cluster¹ (“xce.dwd.de“)
 432 compute nodes Intel Haswell (24 cores/node, 64 GiB memory)
 864 compute nodes Intel Broadwell (36 cores/node, 64 GiB memory)
 MPI: Cray MPICH 7.0.1
 NetCDF: Version 4.3.2
 Compiler: Cray Fortran v8.4.1

This compiler setup is defined in the configuration file `config/mh-linux`.

HLRE-3 cluster “Mistral“ (DKRZ Hamburg)
 1500 compute nodes Intel Haswell (2 CPUs/node, 12 cores/CPU)
 1500 compute nodes Intel Broadwell (2 CPUs/node, 18 cores/CPU)
 MPI: Intel MPI library 2017.0.098
 NetCDF: Version 4.4.2
 Compiler: GNU compiler gcc 6.2.0

This compiler setup is defined in `config/mh-linux`.

Other architecture-dependent configuration files may be added to the directory `icon/config`. Be warned that you need some knowledge about Unix / Linux, compilers and Makefiles to make the necessary adjustments w.r.t. the computing environment.

Due to the usage of modern Fortran 2003/2008 features, ICON places high demands on the compilers. Table 1.1 provides a list of compilers which are known to successfully build the recent ICON code. There’s a good chance that more recent compiler versions might work as well. However, be aware that this is not necessarily the case. E.g. many of the newer versions of the Cray compiler (> `ftn v8.4.1`) might have issues.

¹Currently position # 114 on Top500 List November 2017.

Fortran Compiler	Working Version
GNU	gcc v5.2.0
Cray	ftn v8.4.1
Intel	ifort v16.0.0
NAG	nagfor v6.0.1064

Table 1.1.: Compiler versions which are known to successfully build the ICON code (state February 2018)



Intel ifort compiler: When compiling with ifort before version 17.0.1, the default behaviour is for the compiler not to use the Fortran rules for automatic allocation on intrinsic assignment. You will need to use an option like `-assume realloc-lhs`.

The ICON model supports different modes of parallel execution, see Section 7.3 for details:

- 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>.
- 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>.

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.

1.2.2. Configuring and Compiling

A configure file is provided that takes over the main work of generating the compilation setup. This `autoconf` configuration is used to analyze the computer architecture (hardware and software) and sets user specified preferences, e.g. the compiler. As explained in Section 1.2.1 these preferences are read from `config/mh-<OS>`, where `<OS>` is the identified operating system.

To configure the source code, please log into the Cray XC 40 login node `xce` and change into the sub-directory `icon`. Then type:

```
./configure --with-fortran=compiler
```

where *compiler* is {gcc,nag,intel,pgi,cray}. The default is gcc. Here, for the DWD platform, please choose the option `--with-fortran=cray`.

With the Unix command `make` the programs are compiled and all object files are linked to create the binaries. 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).

During the compilation process, a sub-directory with the name of your computer architecture is created within the `build` directory. In this sub-directory, a `bin` sub-directory containing the binary `icon` and several further sub-directories containing the compiled module files are created.

If you wish to re-configure ICON it is advisable first to clean the old setup by giving:

```
make distclean
```

Some more details on configure options can be found in the help of the configure command:

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



Note for advanced users: Only the Cray XC 40 platform does not require an explicit “`--with-openmp`” option for hybrid parallel binaries. If one uses, e.g., the Intel Fortran compiler, then this option is explicitly needed in the configure process.

1.3. The DWD ICON Tools

1.3.1. General Overview

The DWD ICON Tools provide a number of utilities for the pre- and post-processing of ICON model runs. All of these tools can run in parallel on multi-core systems (OpenMP) and some offer an MPI-parallel execution mode in addition. We give a short overview over several tools in the following and refer to the documentation (Prill (2014)) for details.

ICONGRIDGEN

– Used in Section 2.1.3

The `icongridgen` tool is a simple grid generator. It can create 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.

ICONREMAP

– Used in Sections 2.2.3, 2.3

The `iconremap` utility is especially important for pre-processing the initial data for the basic test setups in this manuscript. `iconremap` (*ICO*sahedral *Non*hydrostatic 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-centred 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.

ICONGPI

`icongpi` (*ICO*sahedral *Non*hydrostatic model *Grid Point Information*) is a utility program for searching / accessing individual grid points of an ICON grid. It can be used to determine cells in a triangular grid corresponding to a given geographical position and to determine the geographical position for a given cell index.

1.3.2. Configuring and Compiling the DWD ICON Tools

To compile the DWD ICON Tools binaries, log into the Cray XC 40 login node `xce` and change into the sub-directory `icontools` by typing

```
cd dwd_icon_tools/icontools/
```

You get a list of available compile targets by typing `make`. The following output is exemplary and may differ from your current version:

```
-----
DWD ICONTOOLS

A set of command-line tools for remapping, extracting and querying
ICON data files.

Available Makefile targets:

target name          platform                parallelization         compiler
-----
local                local DWD workstation,  OpenMP                  gfortran
local_mpi            local DWD workstation,  OpenMP + MPI
ibmp7_mpi            IBM Power 7,            OpenMP + MPI            XLF
cray_mpi              Cray XE 6 / Cray XC 40, OpenMP + MPI            Cray FTN
lce_intel             lce                     OpenMP                  Intel
lce_intel_dbg         lce with debugging flags OpenMP                  Intel
lce_intel_mpi         lce                     OpenMP + MPI            Intel
lce_intel_mpi_dbg     lce with debugging flags OpenMP + MPI            Intel
lce_gfortran_mpi_dbg  lce with debugging flags OpenMP + MPI            gfortran

mistral_intel         Mistral DKRZ            OpenMP + MPI            Intel
mistral_gnu           Mistral DKRZ            OpenMP                  gfortran

fh2_intel_mpi         FH2 KIT                 OpenMP + MPI            Intel
uc1_intel_mpi         UC1 KIT                 OpenMP + MPI            Intel
knl_intel             DWD KNL                 OpenMP                  Intel

nag                   Mistral DKRZ (experimental) OpenMP                  NAGFOR

clean                 remove all object files, libraries and executables
-----
```

For example, the binary for the Cray XC 40 can be created by typing

```
make cray_mpi
```



ICON Tools Libraries: The DWD ICON Tools are divided into several independent libraries which can be linked against user applications. The purpose of this hierarchy of sub-libraries is to access the high-level API (`iconremap`, `iconsub`, `iconmpi`, `icongridgen`) which are part of the overarching `libicontools.a` sub-library, or alternatively call the low-level API

(interpolation, load grid, query point etc.) directly. For the latter case, it is not necessary to read namelists and data via the ICON Tools, since all the necessary data is provided via subroutine interfaces.

The DWD ICON utilities use the GRIB-API/ecCodes package for reading data in GRIB2 format. The GRIB-API/ecCodes package is indirectly accessed by the Climate Data Interface (CDI).

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 forecasts with a limited area model 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 3.8).

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 $n_{\text{cells}} := 20n^24^k$. 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

$$\overline{\Delta x} \approx 5050/(n2^k) \text{ [km]}, \quad (2.1)$$

where S_{earth} denotes the Earth's surface. 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

$$\overline{\Delta x} = \sqrt{S_{\text{earth}}/n_{\text{cells}}} = \sqrt{\frac{4R_{\text{earth}}^2\pi}{n_{\text{cells}}}} = \frac{R_{\text{earth}}}{n2^k} \sqrt{\frac{\pi}{5}} \approx 5050/(n2^k) \text{ [km]}.$$

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.

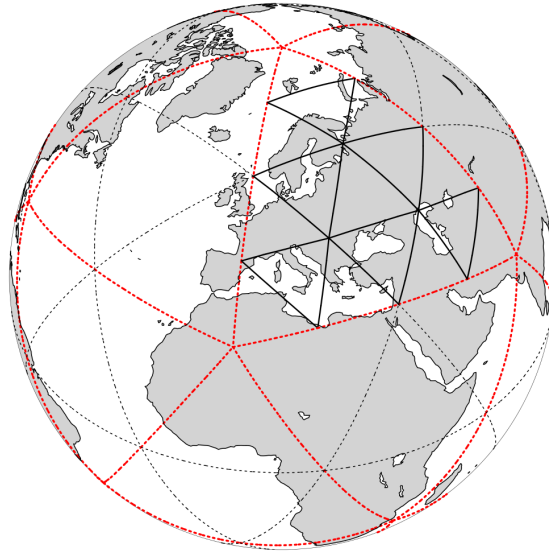


Figure 2.1.: Illustration of the grid construction procedure. The original spherical icosahedron is shown in red, denoted as R1B00 following the nomenclature described in the text. In this example, the initial division ($n=2$; black dotted), followed by one subsequent edge bisection ($k=1$) yields an R2B01 grid (solid lines).

Dual Hexagonal Grid

The centres 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 centres of the triangles form a hexagonal grid that is called to be dual to the grid of triangle vertices. On the ICON grid, the centres of the slightly distorted triangles form a dual grid of slightly distorted hexagons.

Spring Dynamics Optimization

This grid on the sphere will be 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 accurate 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 extend 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

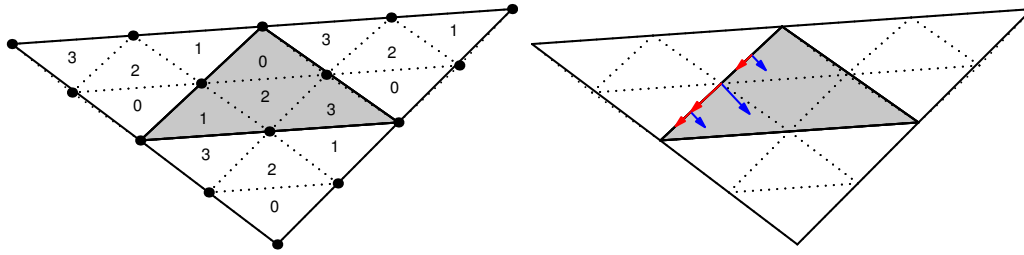


Figure 2.2.: Illustration of the parent-child relationship in refined grids. *Left:* Triangle subdivision and local cell indices. *Right:* The grids fulfill the ICON requirement of a right-handed coordinate system $[\vec{e}_t, \vec{e}_n, \vec{e}_w]$. Note: the primal tangent and the dual cell normal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

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.

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 3.6.1 explains the exchange of information between domains.

Additional topological information is required for ICON’s refined nests: Each “parent” triangle is split into four “child” cells. 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 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 closely located to nest data sites, but they *do not coincide* exactly.

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 circumcenters
- **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.
- **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.

2.1.2. 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.

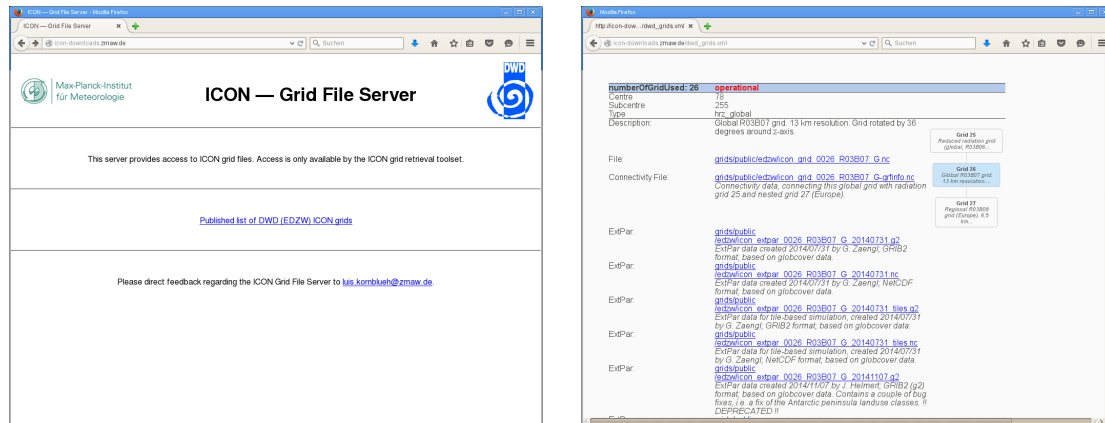


Figure 2.3.: Screenshots of the ICON download server hosted by the Max Planck Institute for Meteorology in Hamburg.

The contents of the primary storage directory are regularly mirrored to a public web site for download, see Figure 2.3 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.

Note that the grid information of some of the older grids (no. 23 – 40) is split over two files: The users need to download the main grid file itself *and* a *grid connectivity* file (suffix `-grfinfo.nc`).

2.1.3. 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. Linaridakis¹. 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 2.1.4. It is important to note, however, that this grid generator is not capable of generating non-spherical geometries like torus grids.

¹see the repository <https://code.mpimet.mpg.de/projects/icon-grid-generator>.



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 must be used together with ICON versions newer than ~ September 2016.

Grid Generator Namelist Settings

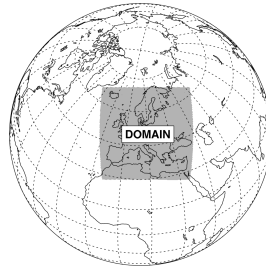
The DWD ICON Tools utility `icongridgen` is mainly controlled using a Fortran namelist.

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 \(2014\)](#)).

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.

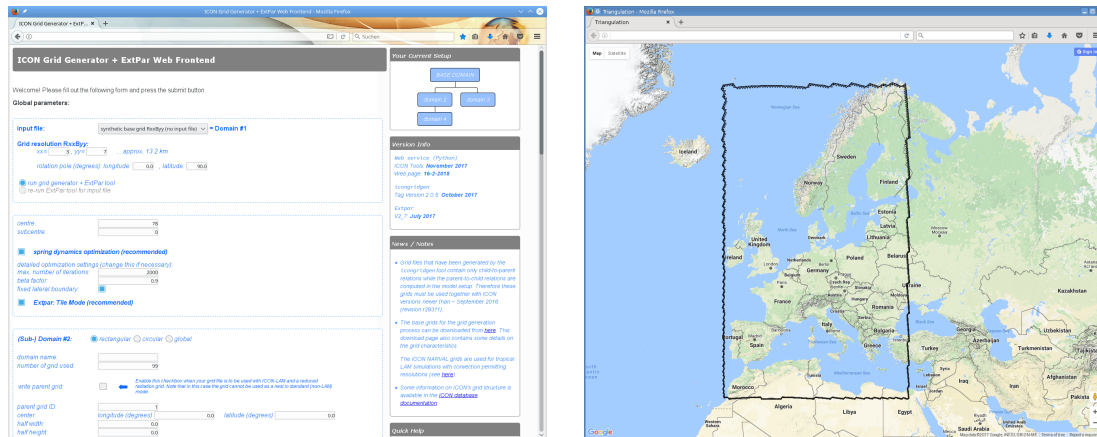


Figure 2.4.: Web browser screenshot of the web-based ICON grid generator tool. *Left:* Web form. *Right:* HTML visualization of the resulting grid based on Google Maps.

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 marked and can be identified in the ICON-LAM setup, which is described in Section 2.3. See Fig. 2.10 for an illustration of such a boundary zone.

- *Generation of a coarse-resolution radiation grid* (see Section 3.8 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 standard feedback-loop together with a global grid.

2.1.4. Grid Generator: Invocation via the Web Interface

A web service has been made available to help users with the generation of custom grid files. After entering grid coordinates through an online form, this web service creates a corresponding ICON grid file together with the necessary external parameter file.

You will need to log in via the user `icon-web`. For the necessary login password – or if you have trouble accessing the web service – please contact `icon@dwd.de`. Then, visit the web page of the grid generator

<https://webservice.dwd.de/cgi-bin/spp1167/webservice.cgi>

The web form is more or less self-explanatory. The settings reflect the namelist parameters of the `icongridgen` grid generator tool that runs as the first stage of the web service. These are explained in the following. The second stage, the `ExtPar` tool, does not require further settings (with the exception of the surface tiles setting, see below).

The tool is capable of generating multiple grid files at once. **Please note that the web-based grid generation submits a batch job to DWD’s HPC system and takes some time for processing!** Due to limited computing resources a threshold is imposed: the maximum grid size which can be generated is 3 000 000 cells. Of course, larger grid sizes are allowed when using the grid generator apart from the the web interface.

Finally all results (and log files) are packed together into a `*.tar.gz` archive and the user is informed via e-mail about its FTP download site. Additionally, a web browser visualization of the grids based on *Google Maps* is provided, see Fig. 2.4.

Step 1: Choosing the Base Grid

The web-based generation of ICON grids and their corresponding external parameter (`ExtPar`) data sets starts from an “input file”, which can be chosen from a pull-down menu with a pre-defined list of grids. These grids are identical to those of the download list described in Section 2.1.2.

It is also possible to start from a “synthetic” base grid $RnBk$ by specifying n and k , following the algorithm by [Sadourny et al. \(1968\)](#).

Note that when starting from an already existing file, the base grid will not be modified by the grid generator, and it will not be stored together with the generator output. The user merely chooses a sub-region on the globe where the base grid is extracted and refined. The result grid will then have half of the grid size of the base grid.

Step 2: Specify Global Options

A number of options will be applied to all produced data sets (“global options”):

- **“centre”, “subcentre”**
These numbers are stored in the output meta-data section for the identification of the originating / generating sub-centre. The values are defined by the WMO, e.g. DWD: 78/0.
- **“spring dynamics optimization”**
The ICON grids are based on the spherical icosahedron, but they are post-processed by an iterative algorithm inspired by elastostatics, see the explanation on “spring dynamics optimization” in Section 2.1.

- **”max. number of iterations”**
maximum number of pseudo-time stepping steps of the elastostatics model.
- **”beta factor”**
stiffness coefficient of the elastostatics model.
- **”fixed lateral boundary”**
If this checkbox is set, then the boundary vertices of regional grids are not moved during the optimization. Thus they will still coincide with vertices of the base grid.

Recommended: Do not alter the default settings unless you know the details of the underlying algorithm.

- **”ExtPar: Tile Mode”**

The web-based generator can produce ExtPar data with and without additional information for surface tiles (see the explanation on p. 41). The data files do not differ in the number of fields, but only in the way some fields are defined near coastal regions.

The – recommended – default has the tile data enabled. This setting can be changed by disabling the corresponding checkbox in the HTML form.

Step 3: Sub-domain Name and Parent Grid ID

By default, the web form contains only the input fields to specify a single grid. However, more domain specifications can be added to the generator by a click on the ”Add another domain” button (or removed by clicking ”Remove latest domain”).

Numbering: In the web form the base grid is denoted by ”#1” while the created domains are denoted by numbers ”#2”, ”#3” and so on.

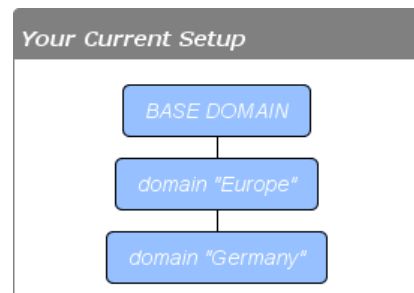
In the simplest case the domains specify multiple nests on the same refined level. In this case, the ”parent grid ID” is always set to ”1” (base grid).

Besides, domains can be nested, for example

Germany → Europe → Global.

Then, the grid for Germany has ”parent grid ID=2”, and the Europe grid has ”parent grid ID=1”.

The currently chosen hierarchy of grids is graphically depicted in the top right corner of the web form.



- **”domain name”**

Each domain requires a name (string) which will be used as a name prefix for the resulting files.

- **”number of grid used”**

This setting will be written into the grid meta-data. It is part of a GRIB2 mechanism to link data files to their underlying grid files, see also the explanation in

Section 2.1.2. For regional domains which are not used operationally, we suggest to choose arbitrary but distinguishable integer numbers.

- **”write parent grid”**

Enable this checkbox when your grid file is to be used with ICON-LAM and a reduced radiation grid. Note that in this case the grid cannot be used as a nest in standard (non-LAM) mode – see the corresponding remark in Section 2.1.3 above.

Step 4: Specify the Grid Type/Shape

- **”global”**

In this case, all cells of the base grid are refined, resulting in a grid with $(4N)$ triangles if the base grids consists of N triangles. No further settings are required to specify this grid.

- **”rectangular”**

This specifies a sub-region to be refined by a center latitude/longitude (in degrees) and a size of $2 \times$ ”half height” for the latitude and $2 \times$ ”half width” in terms of the longitude.

- **”rotate” / ”north-pole”**

By default, the latitude-longitude coordinates for the rectangular refinement area are based on the standard North Pole 90N0E. You may use, however, a rotated pole similar to the grids of the COSMO model.

- **”circular”**

This defines a circular-shaped refinement region with a given center and radius (in degrees).

Finally, having filled out all necessary fields of the web form, click on the “Proceed” button. The grid generation job is inserted into DWD’s processing queue and you will be informed via e-mail about its completion.

2.1.5. 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).

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. Running forecasts with a limited area model in addition requires accurate boundary conditions sampled at regular time intervals.

Initial conditions are usually generated by a process called *data assimilation*². 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 and discuss the data sets that can be used to drive the ICON model.

Each computational domain, i.e. also the nested sub-region, 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. The basic preprocessing steps for ICON are visualized in Figure 2.5.

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. They are available in GRIB2 format on the native ICON grid. The analysis products are generated by a hybrid system named *EnVar* which combines variational and ensemble methods. See Chapter 11 for more information on DWD’s data assimilation system.

²Note that for so-called *idealized test cases* no initial conditions must be read in. All necessary state variables are preset by analytical values.

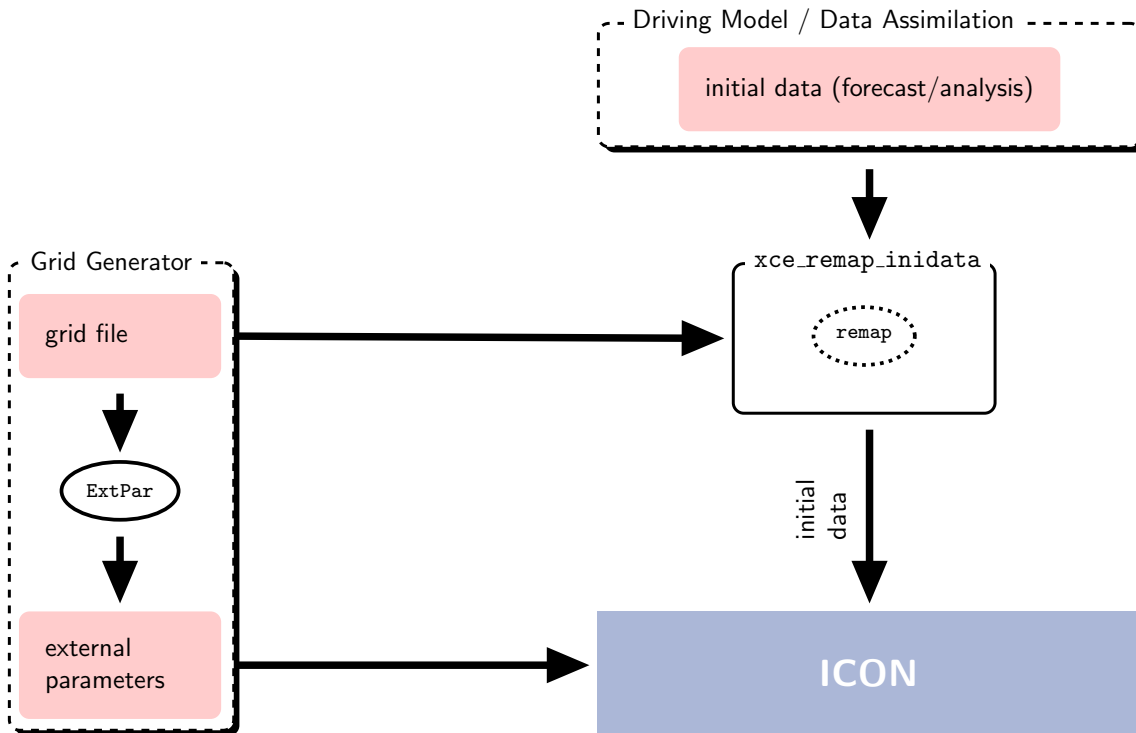


Figure 2.5.: Basic preprocessing steps for ICON (without limited area mode). The grid generation process is described in Sections 2.1 – 2.4. The initial data processing is covered by Section 2.2.3.

Choosing the Initial Data

Basically, what can be thought of as *the analysis* consists of two different data products: a first guess 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 those 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 consist of a first guess file and an analysis file, with the latter containing analysis *increments*. It is meant for starting the model in *incremental analysis update* (IAU) mode. IAU is a model internal filtering method for reducing spurious noise introduced by the analysis (see below).

While this initialization method performs best in terms of noise reduction, it bears the disadvantage that the corresponding analysis product contains surface-tile data (see Section 3.7.8). These data fields cannot be easily interpolated horizontally, which obstructs the use of the *uninitialized analysis for IAU* on custom target grids.

Uninitialized analysis

This product consists of a first guess file and an analysis file, with the latter containing *full* analysis fields instead of increments.

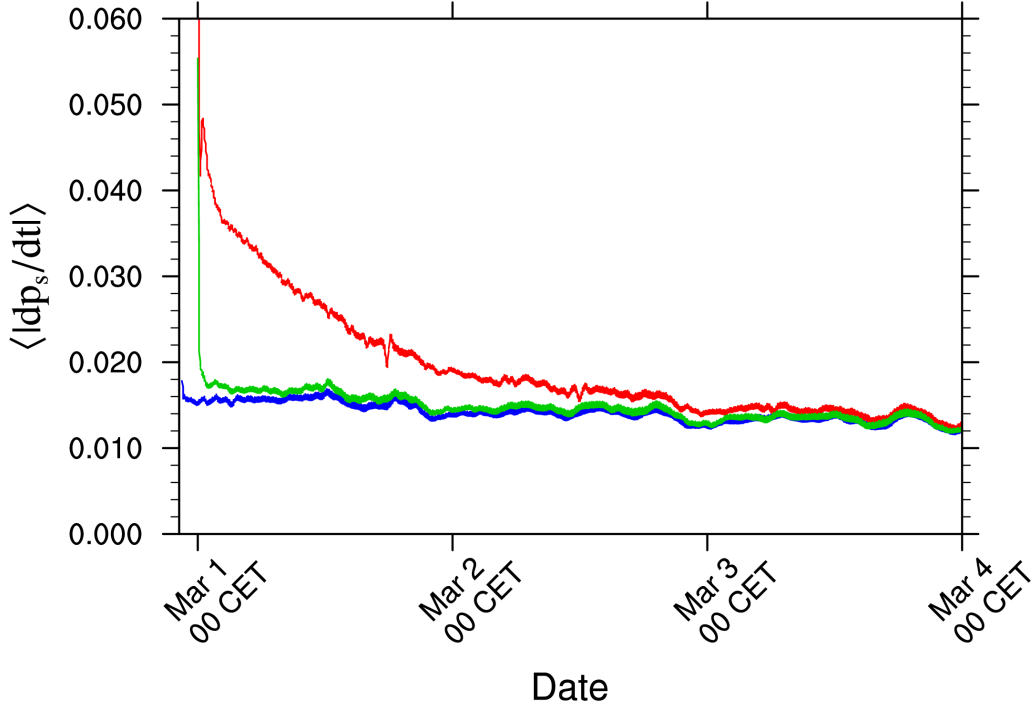


Figure 2.6.: Area averaged absolute surface pressure tendency in hPa as a function of simulation time. Curves differ in terms of the way the model is initialized, with the *uninitialized analysis for IAU* in blue, the *uninitialized analysis* in red and the *initialized analysis* in green.

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.

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. The noise level introduced by this product is very moderate (see below). In addition, this product can also be interpolated horizontally to arbitrary custom target grids.

The level of spurious noise that emerges from each of these analysis products can be deduced from Figure 2.6. It shows the area averaged absolute surface pressure tendency

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

as a function of simulation time, which is a measure of spurious gravity-noise induced by spurious imbalances in the initial conditions. The mathematical steps to obtain the

pressure tendency equation are shown at the end of Section 3.2. The blue, green and red curves show results for the *uninitialized analysis for IAU*, the *initialized analysis*, and the *uninitialized analysis*, respectively. It can be seen that for the *uninitialized analysis* the noise level at simulation start is significantly increased when compared to the other two modes. It takes about two days of model forecast for the noise levels to align. While the *uninitialized analysis for IAU* performs best in terms of noise-level, it has the disadvantage that some data fields cannot easily be interpolated in the horizontal, such that the application of this mode is typically restricted to the horizontal grid (13 km grid spacing) used operationally at DWD.



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.

Option 1: Uninitialized Analysis Product for IAU

In the *incremental analysis update* (IAU) method (Bloom et al., 1996, Polavarapu et al., 2004) the analysis increment is not added in one time step completely, but it is integrated into the model integration and added to the model states $x_k^{(b)}$ over an interval ΔT , which by default is $\Delta T = 3$ h. This method of tentatively pulling the model from its current state (first guess) towards the analyzed state acts as a low pass filter in frequency domain on the analysis increments, such that small scale unbalanced modes are effectively filtered.

In the following, let us assume that we want to start a model forecast at 00 UTC. Technically, the application of this method has some potential pitfalls, which the user should be aware of:

- The analysis file has to contain analysis increments (i.e. deviations from the first guess) instead of full fields, with validity time 00 UTC. The only exceptions are FR_ICE and T_SEA (or alternatively T_S0(0)), which must be full fields (see Table 2.1).
- The model must be started from a first guess which is shifted back in time by 1.5 h w.r.t. to 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 UTC – 1.5h, 00 UTC + 1.5h]. See Section 5.1.3 for an illustration of this process.

Table 2.1 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. In columns 3 and 4 it is indicated, whether the field is part of the first guess file and/or analysis file. The marker (I) highlights analysis increments as opposed to full fields. Fields which are optional for starting the model with IAU are highlighted in blue.

As will be explained in Section 11.2, the atmospheric analysis is performed more frequently than the surface analysis. Therefore, the analysis product provided at times different from 00 UTC usually contains only a subset of the fields provided at 00 UTC. Consequently, Table 2.1 will look different for non-00 UTC runs in such a way that the fields

FR_ICE T_SEA W_SO

will not be provided in the analysis file and will instead be contained in the first guess file.



Important note: In this context, the *surface tile approach*, that is explained in Section 3.7.8, deserves an additional remark. The tile approach is used operationally in ICON. However, due to the necessary remapping step, it is not possible to remap tiled surface data. Remapping of the tiled datasets makes no sense, since the tile-characteristics can differ significantly between a source and target grid cell. Instead, *aggregated* surface fields must be remapped.

Option 2: Uninitialized Analysis Product

The *uninitialized analysis* 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 2.2 provides an overview of the fields contained in the *uninitialized analysis product* for 00 UTC. Columns 3 and 4 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.

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 2.2 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.

Option 3: Initialized Analysis Product

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 created 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 2.3 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.

Downloading Initial Conditions

ICON initial conditions are stored within 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" (contact datenservice@dwd.de). An alternative way to get the data is to use DWD's open data portal (see Section 0.2).



DWD's operational analysis and forecast products for the ICON model are being stored to the SKY database since 2015-01-20. Nevertheless, 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 3.7.8) has been activated no earlier than December 2015.

Data retrieval with PAMORE. As a first option, the SKY data can be accessed via the high-level PAMORE tool (PAMORE: *PAR*allel *MO*del data *RE*trieve from Oracle databases).

We will describe the PAMORE command-line usage in the following. Please note that there also exists a web-based PAMORE service, see

<https://www.dwd.de/DE/leistungen/pamore/pamore.html>

The web-based PAMORE service requires a user account, to this end please contact Klima.Vertrieb@dwd.de.

In order to retrieve, for example, initial data from July 3, 2017 00 UTC, on the native ICON grid the following command lines can be used for the different analysis products:

Uninitialized analysis for IAU

Global domain with 13 km grid spacing

```
~/for0exp/bin/pamore -d 2017070300 -lt m -iglo_startdata -iau
```

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

```
~/for0exp/bin/pamore -d 2017070300 -lt m -iglo_eu_startdata -iau
```

Uninitialized analysis

Global domain with 13 km grid spacing

```
~/for0exp/bin/pamore -d 2017070300 -lt m -iglo_startdata
```

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

```
~/for0exp/bin/pamore -d 2017070300 -lt m -iglo_eu_startdata
```

Initialized analysis

Global domain with 13 km grid spacing

```
~for0exp/bin/pamore -d date -hstart 0 -hstop 0 -lt m \
    -model iglo -iglo_startdata_0
```

Nested EU domain (6.5 km):

```
VARLIST="hhl%genv,fr_land,w%genv,qc%genv,qi%genv,qr%genv,qs%genv, \
    qv%genv,u%genv,v%genv,t%genv,p%genv,tke%genv,z0,t_g, \
    t_mnw_lk,t_wml_lk,h_ml_lk,t_bot_lk,c_t_lk,qv_s,w_i,t_so, \
    w_so_ice,w_snow,rho_snow,t_ice,h_ice,fr_ice,w_so,t_snow, \
    h_snow,freshsnw"
```

```
~for0exp/bin/pamore -d date -vv 0 -lt main -model ieu \
    -ee "${VARLIST}" -hstart 0 -hstop 0
```

Here, *date* must be replaced by the desired date. Please note that for the nested domain Z0 and H_SNOW are only available since 2018-03-14.



Important note: When providing the additional options

- `-ires r2b06` 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 LETKF analysis ensemble consisting of 40 members with 40/20 km horizontal grid spacing. Currently this option is not available for the initialized analysis.

Data retrieval with sky4icon. Alternatively, there exists a Python script `sky4icon` for the automatic request of native (DWD) analysis data from DWD's meteorological data management system SKY. It is located in the sub-directory

```
icon/scripts/preprocessing/sky4icon
```

and directly generates the database requests underlying the above-mentioned PAMORE calls.

This script allows to import analysis data for both IAU and non-IAU runs on the native ICON grid, including data for the ICON-EU nest. However, it is *not* possible to download initialized analysis data with `sky4icon`.

In order to retrieve, for example, 6-hourly initial data from July 1, 2017 00 UTC until July 3, 2017 00 UTC for an IAU-based model initialization, the following command line is used:

```
./sky4icon 20170701000000 20170703000000 --increment 6
```

year
month
day
hour
minute
second

A full set of command-line options can be obtained via `sky4icon.py -h`, see Fig. 2.7.

```

SKY4ICON.PY
  Retrieve ICON data from the DWD "Sky" database.

usage: sky4icon.py [-h] [--increment INCREMENT] [--mode MODE] [--ensemble]
                  [--emember EMEMBER]
                  startdate enddate

positional arguments:
  startdate              start date [YYYYMMDDhhmmss]
  enddate                end date [YYYYMMDDhhmmss]

optional arguments:
  -h, --help            show this help message and exit
  --increment INCREMENT
                        increment time span [h] (default: 24)
  --mode MODE           mode: 1=IAU, 2=No IAU (default: 1)
  --ensemble            read ensemble data (default: False)
  --emember EMEMBER    ensemble member (default: 1)

```

Figure 2.7.: Available command-line options for the `sky4icon` script.



Ensemble data: Note that the script supports ensemble data as well. The command-line options `--ensemble` `--emember emember` allow you to pick one or more analysis from the LETKF analysis ensemble with 40/20 km horizontal grid spacing (member *emember*), as an alternative to the high-resolution (13/6.5 km) deterministic analysis produced by the EnVar system.

2.2.2. Obtaining ECMWF IFS Initial Data

Model runs can also be initialized by “external” analysis files produced by the Integrated Forecast 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. The Meteorological Archival and Retrieval System (MARS, see <https://software.ecmwf.int/wiki/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 2.4.

The script for importing from MARS is located in the sub-directory

```
icon/scripts/preprocessing/mars4icon_smi
```



Important note: The script `mars4icon_smi` must be executed on the ECMWF computer system!

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
```

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>

Note that 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 of SWV to SMI is currently performed as part of the MARS request (`mars4icon_smi`). However, this conversion is not reflected in the variable short names. The fields containing SMI for each surface layer are still termed `SWVLx`. The ICON model, however, expects them to be named `SMIx`. Therefore, the proper output name `SMIx` must be specified explicitly in the namelist `input_field_nml` of `iconremap` (see the example remapping script in Section 2.2.3).

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 initial data. One important application are high-resolution limited area runs, which start from operational ICON forecasts or analysis. In this case horizontal remapping of the initial ICON 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 interpolation of the initial data 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. To this end, the `iconremap` utility from the DWD ICON Tools will be used in batch mode. It is important to note that very large grids should always be processed MPI-parallel in batch mode.

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

```
&remap_nml
  in_grid_filename = "${INPUT_GRID_FILENAME_NC}"
  in_filename      = "${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 and v
  inputname      = "U", "V"
  outputname     = "VN"
/
...

```

For each of the variables to be remapped, the script contains a namelist `input_field_nml` which specifies details of the interpolation methods and the output name. The u and v wind components require special treatment. These must be interpolated to edge-normal wind components v_n (see the namelist above). A detailed documentation of the ICON remap namelist parameters can be found under `dwd_icon_tools/doc/icon_tools_doc.pdf`, i.e. Prill (2014).

For both, DWD analysis data and ECMWF IFS data, the DWD ICON Tools contain example run scripts which performs the remapping. These files are

DWD initial data: `dwd_icon_tools/icon_tools/xce_remap_inidata`

ECMWF IFS data: `dwd_icon_tools/example/runscripts/xce_ifs2icon.run`

After adjusting the necessary file and directory names the scripts can be submitted to the PBS batch system of the Cray XC 40.



Important note: When remapping land surface fields, it is advisable to make use of the land sea mask information (`var_in_mask="FR_LAND"` in `input_field_nml`). By doing so, we mask out water points so that only land points contribute to the interpolation stencil.



Important note: For simplicity, the script `xce_remap_inidata` interpolates the soil water content `W_SO` directly. A more accurate and advisable approach would be to convert `W_SO` into the soil moisture index `SMI` beforehand. A short Fortran program `smi_w_so.f90` which performs this task ships with the ICON code and can be found in the sub-directory `icon/scripts/postprocessing/tools`.

If `SMI` instead of `W_SO` is provided to ICON, it will automatically be converted to `W_SO` during setup¹.

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

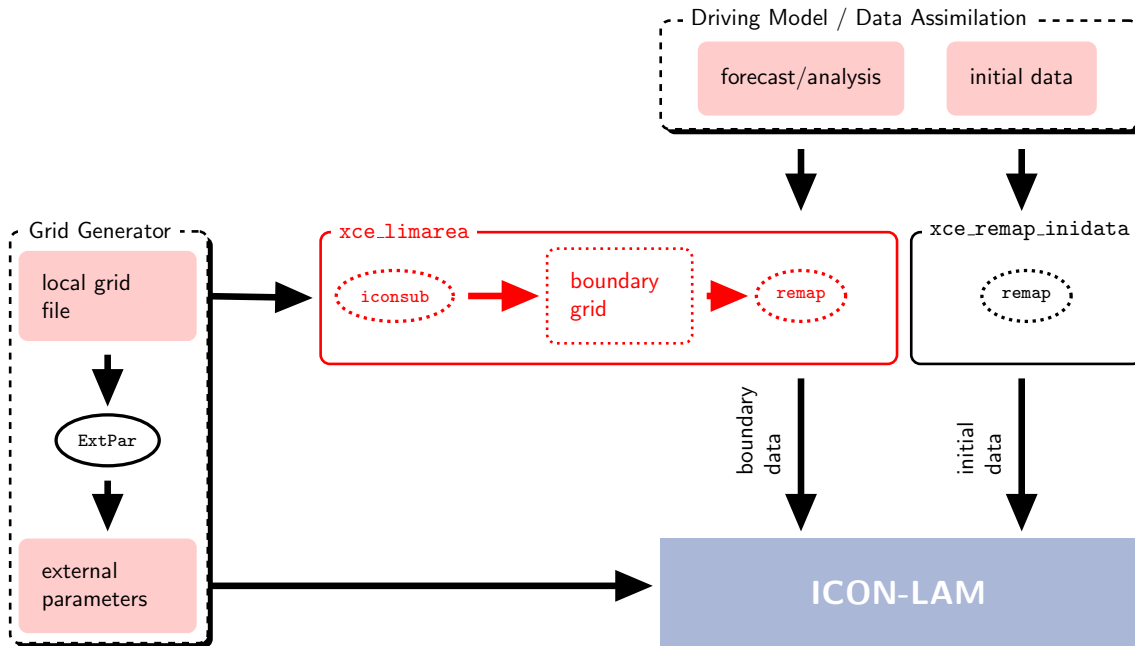


Figure 2.8.: Basic preprocessing steps for ICON-LAM (compare to Fig. 2.5). The grid generation process is described in Sections 2.1 – 2.4. The initial data processing is covered by Section 2.2.3. Finally, the script `xce_limarea` for extracting the boundary data is described in Section 2.3. This preprocessing step is necessary for the limited area model ICON-LAM.

2.3. Preparing Boundary Data 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 preprocessing steps for ICON are visualized in Figure 2.8, where the additional preprocessing of boundary data constitutes the main difference to Figure 2.5. The raw data files which are intended to be used as LBCs must contain one of the sets of variables depicted in Figure 2.9, on either a triangular ICON grid, or a regular latitude-longitude grid.

Optional fields are marked in light-gray. The fact that different sets of variables can be used, provides some flexibility in terms of the driving model. As indicated in Figure 2.9, 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,	PS,	QV,	QC,	QI,	QR,	QS,	GEOSP

Figure 2.9.: Sets of variables that may serve as lateral boundary conditions for ICON-LAM. DWD GRIB2 short names are used. Optional fields are marked in gray. PS denotes the surface pressure, or its logarithm, and GEOSP denotes the surface geopotential. Blending of the sets is not allowed. Examples of driving models are given in brackets.



Important note: The 3D field HHL field (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 envisaged model start date.

ICON-LAM Preprocessing Script

The DWD ICON Tools contain a run script `xce_limarea` which processes a whole directory of data raw files (script variable “DATADIR”), mapping the fields onto the boundary zone of a limited area grid.

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 grid file names are specified by

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

After adjusting the necessary filenames `INGRID` and `LOCALGRID` and the input directory name `DATADIR` in `dwd_icon_tools/icontools/xce_limarea`, this script performs the steps described in the following two sections. The `xce_limarea` script can be submitted to the PBS batch system of the Cray XC 40.

Step 1: Extract Boundary Region from the Local Grid File

In the first step the above ICON-LAM preprocessing script creates 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.

We use the `iconsub` program from the collection of ICON Tools, see Section 1.3, 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
/
```

Then 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 value in a special meta-data field, the `refin_c_ctrl` index, e.g. `refin_c_ctrl = 1, ..., 14`, see Figure 2.10.

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 preprocessing 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
/
```

Here, the data file `input_data_file` automatically iterates over all files in `$(DATADIR)`. 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 preprocessed variables.

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.

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

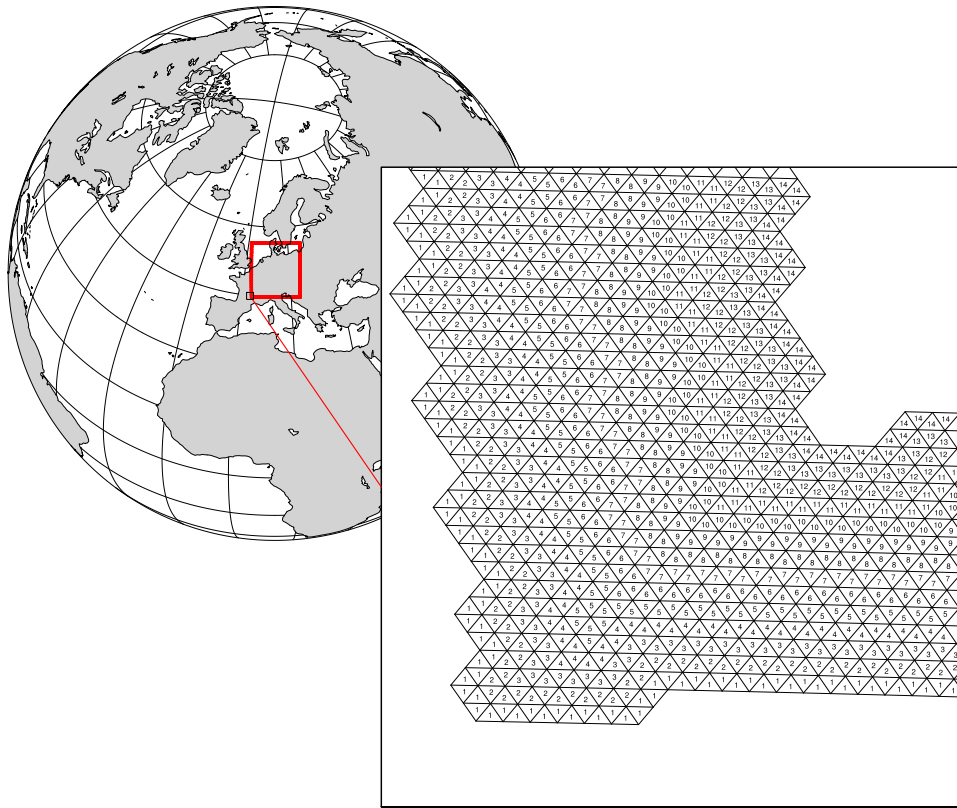


Figure 2.10.: Illustration of the ICON-LAM boundary zone. The cells in this boundary zone are identified by their `refin_c_ctrl` index, e.g. `refin_c_ctrl = 1, ..., 14`.

process the whole list of boundary data samples (this is automatically done within the `xce_limarea` script).

After the specification of the filenames, the `xce_limarea` script automatically adds remapping parameters for all variables of the Set I in Fig. 2.9. 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 above), we set this field as optional:

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

Afterwards, the `xce_limarea` script launches the call to the `iconremap` binary.

In this context the following technical detail may considerably speed up the preprocessing: 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 Data Files

External parameters are used to describe the properties of the Earth's surface. These data include the topography, the land-sea mask, and several parameters which are needed to specify the dominant land use of a grid box like the soil type or the plant cover fraction.

2.4.1. ExtPar Products

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 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 preprocessor is a COSMO software and not part of the ICON training course release. Still, the ExtPar tool can be accessed via the ICON grid generator web service (see Section 2.1.4). Similar as for the grid files, for fixed domain sizes and resolutions some external parameter files for the ICON model are available for download. For the NWP release these data are provided in the NetCDF file format and GRIB2.



Topography information: Please note the following remark:

The topography contained in the ExtPar data files is *not identical* to the topography data which is eventually 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. Therefore, for post-processing purposes it is necessary to specify and use the topography height `topography_c` (GRIB2 short name `HSURF`) from the model output (cf. Section 7.1 and Appendix C).

2.4.2. Additional Information for Surface Tiles

ExtPar data is available with and without additional information for surface tiles. See Section 3.7.8 for details on the tile approach.

ExtPar data 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 `lnd_nml`).

The data files do not differ in the number 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. In files without the `_tiles` suffix, it is only defined at dominant lake points.

2.4.3. Parameter Files for Radiation

In addition to the ExtPar products, input fields for radiation are loaded into the ICON model. These constants fields are distributed together with the model code in the sub-directory `icon/data`.

`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 2.1.: 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 (I) indicates analysis increments as opposed to full fields. Analysis fields highlighted in **red** are only available for 00 UTC. The validity date of the first guess is shifted by 1.5 h w.r.t. the start date.

shortName	Unit	Description	Source	
			FG	ANA
VN	m s^{-1}	edge normal velocity component	x	
W	m s^{-1}	vertical velocity	x	
DEN	kg m^{-3}	air density	x	
THETA_V	K	virtual potential temperature	x	
QV	kg kg^{-1}	water vapour mass fraction	x	x (I)
QC	kg kg^{-1}	cloud liquid water mass fraction	x	
QI	kg kg^{-1}	cloud ice mass fraction	x	
QR	kg kg^{-1}	rain water mass fraction	x	
QS	kg kg^{-1}	snow mass fraction	x	
TKE	$\text{m}^2 \text{s}^{-2}$	turbulent kinetic energy	x	
U, V	m s^{-1}	horizontal velocity components		x (I)
T	K	air temperature		x (I)
P	Pa	pressure		x (I)
ZO	m	surface roughness length	x	
QV_S	kg kg^{-1}	surface specific humidity	x	
W_I	kg m^{-2}	water content of interception layer	x	
T_G	K	surface temperature	x	
RHO_SNOW	kg m^{-3}	snow density	x	
T_SNOW	K	snow temperature	x	
H_SNOW	m	snow depth	x	x (I)
FRESHSNW	1	age of snow indicator	x	x (I)
SNOWC	%	snow cover	x	
T_ICE	K	sea ice temperature	x	
H_ICE	m	sea ice depth	x	
FR_ICE	1	sea/lake ice fraction		x
ALB_SEAICE	%	sea ice albedo	x	
T_MNW_LK	K	mean temperature of the water column	x	
T_WML_LK	K	mixed-layer temperature	x	
H_ML_LK	m	mixed-layer thickness	x	
T_BOT_LK	K	temperature at water-bottom sediment interface	x	
C_T_LK	1	shape factor w.r.t. temp. profile in the thermocline)	x	
T_SEA	K	sea surface temperature		x
T_SO	K	soil temperature	x	
W_SO	kg m^{-2}	soil water content (liq. + ice)	x	x (I)
W_SO_ICE	kg m^{-2}	soil ice content	x	

Table 2.2.: 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
VN	m s^{-1}	edge normal velocity component	x	
W	m s^{-1}	vertical velocity	x	
DEN	kg m^{-3}	air density	x	
THETA_V	K	virtual potential temperature	x	
QV	kg kg^{-1}	water vapour mass fraction		x
QC	kg kg^{-1}	cloud liquid water mass fraction	x	
QI	kg kg^{-1}	cloud ice mass fraction	x	
QR	kg kg^{-1}	rain water mass fraction	x	
QS	kg kg^{-1}	snow mass fraction	x	
TKE	$\text{m}^2 \text{s}^{-2}$	turbulent kinetic energy	x	
U, V	m s^{-1}	horizontal velocity components		x
T	K	air temperature		x
P	Pa	pressure		x
ZO	m	surface roughness length	x	
QV_S	kg kg^{-1}	surface specific humidity	x	
W_I	kg m^{-2}	water content of interception layer	x	
T_G	K	surface temperature	x	
W_SNOW	kg m^{-2}	snow water equivalent	x	
RHO_SNOW	kg m^{-3}	snow density	x	
T_SNOW	K	snow temperature		x
H_SNOW	m	snow depth		x
FRESHSNW	1	age of snow indicator		x
T_ICE	K	sea ice temperature		x
H_ICE	m	sea ice depth		x
FR_ICE	1	sea/lake ice fraction		x
ALB_SEAICE	%	sea ice albedo	x	
T_MNW_LK	K	mean temperature of the water column	x	
T_WML_LK	K	mixed-layer temperature	x	
H_ML_LK	m	mixed-layer thickness	x	
T_BOT_LK	K	temperature at water-bottom sediment interface	x	
C_T_LK	1	shape factor w.r.t. temp. profile in the thermocline)	x	
T_SEA	K	sea surface temperature		x
T_SO	K	soil temperature	x	
W_SO	kg m^{-2}	soil water content (liq. + ice)		x
W_SO_ICE	kg m^{-2}	soil ice content	x	

Table 2.3.: Content of the **initialized analysis product**. Optional fields for model initialization are marked in **blue**.

shortName	Unit	Description
U, V	m s^{-1}	horizontal velocity components
W	m s^{-1}	vertical velocity
T	K	air temperature
P	Pa	pressure
QV	kg kg^{-1}	water vapour mass fraction
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
TKE	$\text{m}^2 \text{s}^{-2}$	turbulent kinetic energy
Z0	m	surface roughness length
QV_S	kg kg^{-1}	surface specific humidity
W_I	kg m^{-2}	water content of interception layer
T_G	K	surface temperature
W_SNOW	kg m^{-2}	snow water equivalent
RHO_SNOW	kg m^{-3}	snow density
T_SNOW	K	snow temperature
H_SNOW	m	snow depth
FRESHSNW	1	age of snow indicator
T_ICE	K	sea ice temperature
H_ICE	m	sea ice depth
FR_ICE	1	sea/lake ice fraction
T_MNW_LK	K	mean temperature of the water column
T_WML_LK	K	mixed-layer temperature
H_ML_LK	m	mixed-layer thickness
T_BOT_LK	K	temperature at water-bottom sediment interface
C_T_LK	1	shape factor w.r.t. temp. profile in the thermo-cline)
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

Table 2.4.: Recommended **IFS analysis fields** on a regular lat-lon grid, as retrieved by the script `mars4icon_smi`. Optional fields are marked in blue.

shortName	Unit	Description
ECMWF		
U, V	m s^{-1}	horizontal velocity components
W	Pa s^{-1}	vertical velocity
T	K	Temperature
FI	$\text{m}^2 \text{s}^{-2}$	geopotential (at model levels)
QV	kg kg^{-1}	specific humidity
CLWC	kg kg^{-1}	cloud liquid water content
CIWC	kg kg^{-1}	cloud ice content
CRWC	kg kg^{-1}	rain water content
CSWC	kg kg^{-1}	snow water content
SST	K	sea surface temperature
CI	[0,1]	sea-ice cover
LNSP	-	logarithm of surface pressure
Z	$\text{m}^2 \text{s}^{-2}$	surface geopotential
TSN	K	snow temperature
SD	m of water eqv.	water content of snow
RSN	kg m^{-3}	density of snow
ASN	[0,1]	snow albedo
SKT	K	skin temperature
STL1	K	soil temperature level 1
STL2	K	soil temperature level 2
STL3	K	soil temperature level 3
STL4	K	soil temperature level 4
SWVL1	$\text{m}^3 \text{m}^{-3}$	soil moisture index (SMI) layer 1
SWVL2	$\text{m}^3 \text{m}^{-3}$	soil moisture index (SMI) layer 2
SWVL3	$\text{m}^3 \text{m}^{-3}$	soil moisture index (SMI) layer 3
SWVL4	$\text{m}^3 \text{m}^{-3}$	soil moisture index (SMI) layer 4
SRC	m of water eqv.	water content of interception storage
LSM	[0,1]	land/sea mask

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 gives a formulation of the governing equations in the ICON model. The vertical coordinate is explained and some details of the variable resolution modeling are discussed. We also present a comprehensive overview of the physics parameterizations available in ICON (NWP-mode).

3.1. Governing Equations

The equation system of the ICON model is based upon the prognostic variables suggested by [Gassmann and Herzog \(2008\)](#). It describes a two-component system consisting of dry air and water, where water is allowed to occur in all three phases, including precipitating drops and ice particles.

As described in [Wacker and Herbert \(2003\)](#), an equation set for the mixture can be derived by first introducing a reference velocity into the governing equations. 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 reference velocity.

Here we have chosen the barycentric velocity as reference velocity. It is defined as

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

with the partial density ρ_k of constituent k and its advective velocity \mathbf{v}_k . For simplicity, \mathbf{v}_b will be denoted as \mathbf{v} in the following.

In order to separate turbulent fluctuations from the mean flow, a density weighted averaging (known as Hesselberg averaging) is applied. Every field ϕ is decomposed into a density-weighted mean and a deviation ([Hesselberg, 1925](#))

$$\phi = \hat{\phi} + \phi''$$

with

$$\hat{\phi} = \frac{\overline{\rho\phi}}{\bar{\rho}}$$

and subsequent averaging. $\bar{\phi}$ denotes the classical Reynolds average. More details on density-weighted average calculus can be found e.g. in [Zdunkowski and Bott \(2003\)](#).

The basic Hesselberg-averaged equation system, including the shallow atmosphere approximations, reads as follows

$$\frac{\partial \hat{v}_n}{\partial t} + \frac{\partial \hat{K}_h}{\partial n} + (\hat{\zeta} + f)\hat{v}_t + \hat{w} \frac{\partial \hat{v}_n}{\partial z} = -c_{pd}\hat{\theta}_v \frac{\partial \bar{\pi}}{\partial n} - F(v_n) \quad (3.2)$$

$$\frac{\partial \hat{w}}{\partial t} + \hat{\mathbf{v}}_h \cdot \nabla \hat{w} + \hat{w} \frac{\partial \hat{w}}{\partial z} = -c_{pd}\hat{\theta}_v \frac{\partial \bar{\pi}}{\partial z} - g \quad (3.3)$$

$$\frac{c_{vd}c_{pd}}{R_d} \bar{\rho} \hat{\theta}_v \frac{\partial \bar{\pi}}{\partial t} = c_{pd}\bar{\pi} \frac{\partial \bar{\rho} \hat{\theta}_v}{\partial t} = -c_{pd}\bar{\pi} \nabla \cdot (\bar{\rho} \hat{\mathbf{v}} \hat{\theta}_v) + \bar{Q} \quad (3.4)$$

$$\frac{\partial \bar{\rho}}{\partial t} + \nabla \cdot (\bar{\rho} \hat{\mathbf{v}}) = \sum_k \bar{\sigma}_k^{conv} \quad (3.5)$$

$$\frac{\partial \bar{\rho} \hat{q}_k}{\partial t} + \nabla \cdot (\bar{\rho} \hat{q}_k \hat{\mathbf{v}}) = -\nabla \cdot \left(\bar{J}_k^z \mathbf{k} + \overline{\rho q_k'' \mathbf{v}''} \right) + \bar{\sigma}_k \quad (3.6)$$

Prognostic equations are solved for the horizontal velocity component normal to the triangle edges \hat{v}_n (3.2), the vertical wind component \hat{w} (3.3), virtual potential temperature $\hat{\theta}_v$ (3.4), the total density of the air mixture $\bar{\rho}$ (3.5), with

$$\bar{\rho} = \sum_k \bar{\rho}_k, \quad (3.7)$$

and mass fractions (3.6)

$$\hat{q}_k = \hat{\rho}_k / \bar{\rho}. \quad (3.8)$$

The index $k \in \{v, c, i, r, s, g\}$ represents a specific constituent of the mixture. We use

$$\begin{aligned} k = d & \quad \text{for dry air,} \\ k = v & \quad \text{for water vapour,} \\ k = c & \quad \text{for cloud water,} \\ k = i & \quad \text{for cloud ice,} \\ k = r & \quad \text{for rain,} \\ k = s & \quad \text{for snow, and} \\ k = g & \quad \text{for graupel.} \end{aligned}$$

Further explanation of symbols and variables is given in [Table 3.1](#). Note that the corresponding data structures containing the physics and dynamics variables are outlined in [Section 8.2](#).

The equation system (3.2)–(3.10) is supplemented by the lower boundary conditions

$$\hat{w}|_s = \frac{\bar{E}_v - \sum_{k_{prec}} \bar{S}_k|_s}{\bar{\rho}|_s - \sum_{k_{prec}} \bar{\rho}_k|_s} \quad (3.9)$$

$$\bar{J}_k^z|_s = \begin{cases} \bar{E}_k - \bar{\rho}_k \hat{w}|_s, & \text{if } k \equiv v \\ -\bar{\rho}_k \hat{w}|_s, & \text{if } k \equiv \text{non-prec. constituent} \\ -\bar{S}_k|_s, & \text{if } k \equiv \text{prec. constituent} \end{cases} \quad (3.10)$$

and the equation of state

$$\bar{p} = R_d \bar{\rho} \hat{T} (1 + \alpha) ,$$

with

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

In contrast to the original formulation by Gassmann and Herzog (2008), we make use of the two-dimensional rather than the three-dimensional Lamb transformation to convert the nonlinear momentum advection term into a vector-invariant form. Vector-invariant means that no gradients of vectors appear in this equation, which avoids derivatives of the coordinate basis that would otherwise arise in an arbitrary coordinate frame from the nonlinear momentum advection term.

Note that we do not explicitly solve a prognostic equation for the density of dry air. From Equation (3.7) it becomes clear that the partial density of one constituent (here $\bar{\rho}_d$) can be diagnosed, given that a prognostic equation for the total density and all but one partial densities is solved. The reconstructed tangential velocity component is denoted as \hat{v}_t , and in accordance with the model code, it is assumed here that $(\hat{v}_t, \hat{v}_n, \hat{w})$ form a right-handed system.

From the Equations (3.5), (3.6) for total density and partial densities some important constraints can be derived which must hold in the discretized analogue in order to achieve mass conservation. First of all, the total density is defined as the sum of all partial densities, as shown by Eq. (3.7). From Eq. (3.8) it follows that

$$\sum_k \hat{q}_k = 1 .$$

Likewise, the prognostic equation for total density (3.5) should be obtained by summing the budget equations of all constituents. As a consequence the following constraints hold:

$$\sum_k \bar{J}_k = 0, \quad \sum_k \overline{\rho q_k'' v''} = 0$$

3.2. Simplifying Assumptions in the Recent Model Version

The recent model version does not account for mass loss/gain due to precipitation/evaporation. In particular, the following assumptions are made:

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
$\widehat{\zeta} = (\nabla \times \widehat{\mathbf{v}}) \cdot \mathbf{k}$	vertical component of relative vorticity
$f = 2\Omega \sin \phi$	Coriolis parameter
π	Exner function
c_{pd}, c_{vd}	specific heat capacity for dry air at constant pressure/volume
$F(v_n)$	turbulent momentum fluxes
g	acceleration of gravity
\overline{Q}	adiabatic heat source
$\overline{J}_k^z = \overline{\rho}_k (\widehat{w}_k - \widehat{w})$	vertical diffusion flux for constituent k
$\overline{\sigma}_k$	internal conversion rate for k th constituent (i.e. conversion among different phases or particle forms)
$\overline{\sigma}_k^{conv}$	internal conversion rate for k th constituent due to convection only
$\overline{\rho q_k'' \mathbf{v}''}$	turbulent flux of k th partial mass fraction
$\overline{E}_v = \overline{\rho q_v'' \mathbf{v}''} _s$	surface evaporation flux
$\overline{S}_k = \overline{\rho} \widehat{q}_k \widehat{v}_k^T$	sedimentation flux of k th constituent
\widehat{v}^T	terminal fall velocity of k th constituent

The term $\sum_k \overline{\sigma}_k^{conv}$ on the r.h.s. of Eq. (3.5), which describes the net mass tendency of the convection parameterization is neglected. Thus the model neither accounts for mass loss due to convective precipitation, nor mass re-distribution due to convective motions. The mass continuity equation takes the form

$$\frac{\partial \overline{\rho}}{\partial t} + \nabla \cdot (\overline{\rho} \widehat{\mathbf{v}}) = 0. \quad (3.11)$$

At the surface, the boundary condition for \widehat{w} , Eq. (3.9), is approximated as

$$\overline{\rho} \widehat{w}|_s = \sum_k \overline{\rho}_k \widehat{w}_k|_s = 0, \quad (3.12)$$

which in terrain following coordinates translates to $\overline{\rho} \widehat{w}|_s = \overline{\rho} \widehat{\mathbf{v}} \cdot \nabla h$. h denotes the topography height. Thus, the simulated atmospheric system is assumed to be closed w.r.t. total mass. The sedimentation fluxes at the surface $\overline{S}_k|_s$ as well as the surface evaporation flux $\overline{E}_v|_s$, which appear in the partial mass continuity equations, are retained. In order for Eq. (3.12) to hold, the (implicit) assumption is that the corresponding mass loss/gain due to sedimentation/evaporation is compensated by a fictitious flux of dry air across the surface.

Away from the surface, the diffusion fluxes of all airborne constituents (except for dry air) are neglected. The diffusive fluxes of all precipitating constituents, however, are taken into account. The continuity equation for the total mass is used in the form (3.11). Since Eq. (3.11) only holds if the constraint $\sum_k \bar{J}_k^z = 0$ holds, again, the (implicit) assumption made is 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 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 affect the surface pressure.

The latter point becomes obvious, when writing down the (hydrostatic) pressure tendency equation and applying the approximations (3.11)–(3.12) (see also Wacker and Herbert (2003)). Starting from the hydrostatic equation $\frac{\partial \bar{p}}{\partial z} = -\bar{\rho}g$ it follows:

$$\begin{aligned} \frac{\partial}{\partial t} \int_{\bar{p}_s}^0 dp &= -g \frac{\partial}{\partial t} \int_{z_s}^{\infty} \bar{\rho} dz \\ \frac{\partial \bar{p}_s}{\partial t} &= g \int_{z_s}^{\infty} \frac{\partial \bar{\rho}}{\partial t} dz \\ \frac{\partial \bar{p}_s}{\partial t} &= -g \int_{z_s}^{\infty} \nabla \cdot (\bar{\rho} \hat{v}) - \sum_k \bar{\sigma}_k^{conv} dz \\ \frac{\partial \bar{p}_s}{\partial t} &= -g \int_{z_s}^{\infty} \nabla_h \cdot (\bar{\rho} \hat{v}_h) dz + g (\bar{\rho} w)|_s + g \bar{P}^{conv}|_s \\ \frac{\partial \bar{p}_s}{\partial t} &= -g \int_{z_s}^{\infty} \nabla_h \cdot (\bar{\rho} \hat{v}_h) dz + g \frac{\bar{\rho}|_s}{\bar{\rho}|_s - \sum_{kprec} \bar{\rho}_k|_s} \left(\bar{E}_v - \sum_{kprec} \bar{S}_k|_s \right) + g \bar{P}^{conv}|_s \end{aligned}$$

Any effect of precipitation/evaporation on the dynamics is neglected.

3.3. Vertical Coordinates

In a nonhydrostatic model, a vertical coordinate in terms of pressure does not make sense since it cannot be taken for granted that the pressure is monotonously decreasing with increasing altitude. In general, the air mass of an air column above a certain point can not be calculated from the pressure at that point anymore. Instead, a geometric altitude grid has to be used.

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

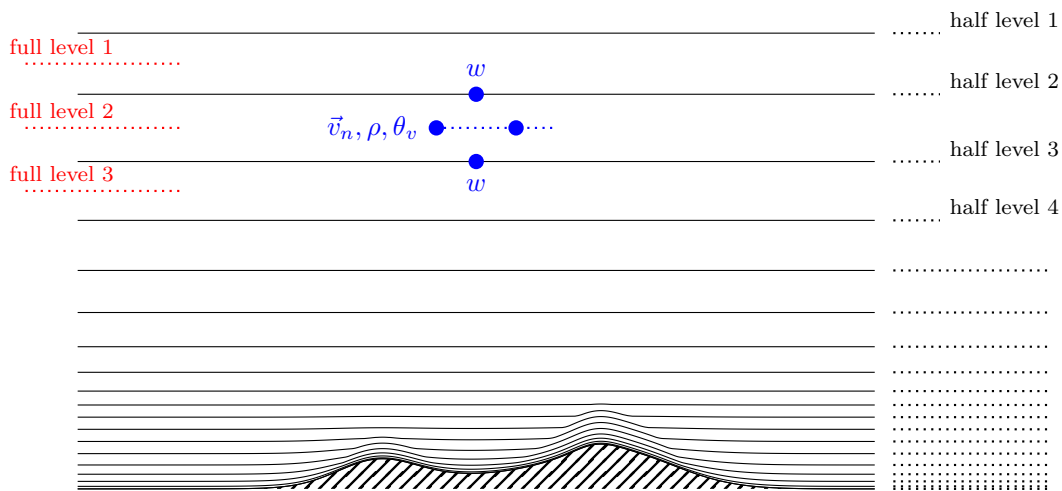


Figure 3.1.: Illustration of ICON’s vertical levels. With `num_lev` layers, there are `num_lev + 1` so-called *half levels*. The half levels $l, l + 1$ enclose layer l at the centres of which are the corresponding full levels l , for $l = 1, \dots, \text{num_lev}$. Layer 1 is at the top of the atmosphere and layer n at the bottom of the atmosphere. Half level `num_lev + 1` coincides with the Earth’s surface.

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 centre of the box is said to be at the *full level* of the vertical grid, see Fig. 3.1 for an illustration. Note that the numbering of full and half levels is top-down, starting with $l = 1$ 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.

When setting up an ICON simulation, the total number of vertical levels has to be specified for each domain via

`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 3.6.3 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 below.

3.3.1. 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

$$\begin{aligned} z(x, y, \eta) &= \frac{(H - B'(\eta) h(x, y))}{H} \eta + B'(\eta) h(x, y) \\ &= \eta + B'(\eta) \left(1 - \frac{\eta}{H}\right) h(x, y), \end{aligned} \quad (3.13)$$

where z represents the height of the coordinate surfaces 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 $z = z_{flat}$. Sometimes, Equation (3.13) is also written in the discretized form

$$z_h(x, y, k) = A(k) + B(k) h(x, y), \quad k = 1, \dots, \text{num_lev} + 1 \quad (3.14)$$

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)
```

In that case the user has to provide the vertical coordinate table (vct) as an input file. The table consists of the A and B values (see Equation (3.14)) from which the half level heights $z_h(x, y, k)$ can be deduced. $A(k)[m]$ are fixed height values, with $A(1)$ defining the model top height H and $A(\text{num_lev} + 1) = 0$ m. 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, at each horizontal grid point $z_h(x, y, 1)$ is 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`. The file must obey the following naming rule: `atm_hyb_sz_[nlev]`, where `[nlev]` must be replaced by the total number of full levels. ICON expects the file to be located in the base directory from which the model is started. Note that the filename specification must not be confused with another parameter which has a similar name, `vertical_grid_filename (grid_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.

3.3.2. 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 now 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`.

3.4. ICON Time-Stepping

For efficiency reasons, different integration time steps are applied depending on the process under consideration. In the following, the term *dynamical core* refers to the numerical solution of the dry Navier-Stokes equations, while the term *physics* refers to the diabatic, mostly sub-grid scale, processes that have to be parameterized. In ICON, the following time steps have to be distinguished:

- Δt the basic time step specified via namelist variable `dtime`, 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 `ndyn_substeps` (namelist `nonhydrostatic_nml`, 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.

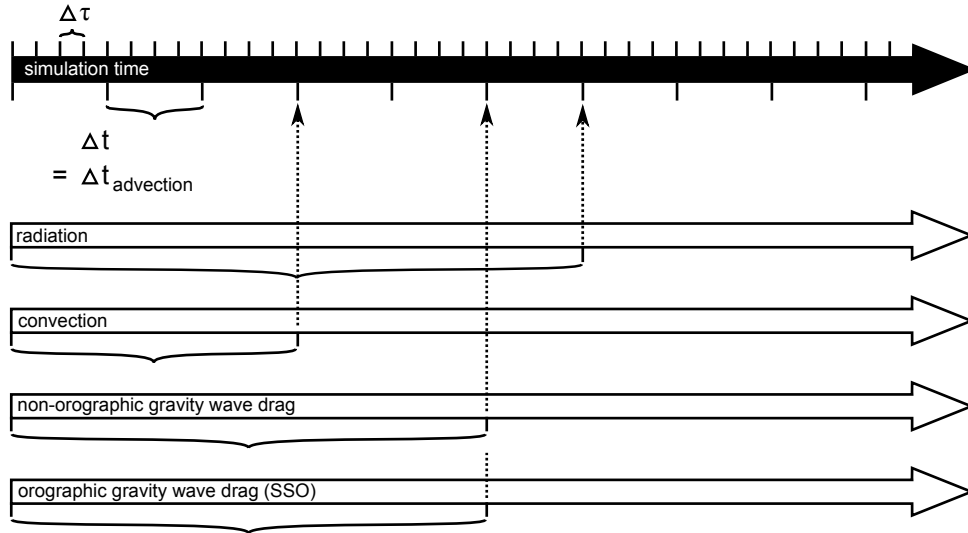


Figure 3.2.: ICON internal time stepping. Sub-cycling of dynamics with respect to transport, fast-physics, and slow-physics. Δt denotes the time step for transport and fast physics and $\Delta\tau$ denotes the short time step of the dynamical core. The time step for slow-physics can be chosen individually for each process. Details of the physics-dynamics coupling will be discussed in Chapter 3.7.

An illustration of the relationship between the time steps can be found in Figure 3.2. More details on the physics-dynamics coupling will be presented in Section 3.7.9.

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. 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

$$\Delta\tau = 1.8 \cdot 10^{-3} \overline{\Delta x} \frac{\text{s}}{\text{m}}, \tag{3.15}$$

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 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 8 if the Courant number for vertical 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 $\overline{\Delta x}$ 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 3.2.

3.5. Tracer Transport

Physically speaking, the tracer transport module predicts the large-scale re-distribution of tracers (e.g. water substances, aerosols, chemical species, ...) caused by air-motions. Mathematically (numerically) this is done by solving the partial mass continuity equation (3.6) for each tracer, with sources and sinks being neglected, i.e.

$$\frac{\partial \bar{\rho} \hat{q}_k}{\partial t} + \nabla \cdot (\bar{\rho} \hat{q}_k \hat{\mathbf{v}}) = 0 \quad (3.16)$$

The numerical solution of Eq. (3.16) 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 is partly based on work by Lauritzen et al. (2010, 2011), Harris and Lauritzen (2010), Skamarock and Menchaca (2010), Miura (2007) (for the horizontal) and Colella and Woodward (1984) (for the vertical).

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

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. Here and in the remainder of this Section, the overbar denotes volume averages rather than Reynolds averages.

3.5.1. Directional Splitting

By integrating the continuity equation (3.16) in space over a prismatic grid cell (the so-called control volume) and in time over the time step Δt , a solution to (3.16) can formally be written as

$$\bar{\rho q}_{i,k}^{n+1} = \bar{\rho q}_{i,k}^n + \Delta t [\mathcal{H}(q^n) + \mathcal{V}(q^n)] , \quad (3.17)$$

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

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 (3.17) by some approximation involving the two subproblems

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

will inevitably result in a residual error. This error is known as the *splitting error*. On default, the following approximation to (3.17) is used:

$$\bar{\rho q}_{i,k}^* = \bar{\rho q}_{i,k}^n + \Delta t \mathcal{V}(\bar{q}^n) \quad (3.18)$$

$$\bar{\rho q}_{i,k}^{n+1} = \bar{\rho q}_{i,k}^* + \Delta t \mathcal{H}(\bar{q}^*) \quad (3.19)$$

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 $[\mathcal{V}(\Delta t/2)][\mathcal{H}(\Delta t)][\mathcal{V}(\Delta t/2)]$ is also available as an option by choosing `lstrang = .FALSE.` (namelist `transport_nml`). Except for being more expensive (the vertical operator is called twice) no significant impact on the model results has been noted so far.

A shortcoming of the splitting (3.18), (3.19) is that it does not preserve an initially uniform tracer field (e.g. $q(x, y, z, t_0) = 1$) in a deformational non-divergent flow. 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 mass continuity equation (3.6) is simultaneously reintegrated with that for partial mass:

$$\begin{aligned}\overline{\rho q}_{i,k}^* &= \overline{\rho q}_{i,k}^n + \Delta t \mathcal{V}(\overline{q}^n) \\ \overline{\rho}_{i,k}^* &= \overline{\rho}_{i,k}^n + \Delta t \mathcal{V}(1) \\ \overline{q}_{i,k}^* &= \frac{\overline{\rho q}_{i,k}^*}{\overline{\rho}_{i,k}^*}\end{aligned}\quad (3.20)$$

$$\begin{aligned}\overline{\rho q}_{i,k}^{n+1} &= \overline{\rho q}_{i,k}^* + \Delta t \mathcal{H}(\overline{q}^*) \\ \overline{\rho}_{i,k}^{n+1} &= \overline{\rho}_{i,k}^* + \Delta t \mathcal{H}(1) \\ \overline{q}_{i,k}^{n+1} &= \frac{\overline{\rho q}_{i,k}^{n+1}}{\overline{\rho}_{i,k}^{n+1}}\end{aligned}\quad (3.21)$$

Changes in partial density solely due to mass convergence/divergence are corrected for in equations (3.20) and (3.21). The key point here is that the intermediate density $\overline{\rho}^*$ rather than $\overline{\rho}^{n+1}$ or $\overline{\rho}^n$ is used to recover the mass fraction \overline{q}^* in (3.20). For re-integration of the mass continuity equation, the mass fluxes that have already been computed in the dynamical core are re-used.

3.5.2. Horizontal Transport

A rigorous derivation of the horizontal transport operator $\mathcal{H}(q)$ 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 ρq_i^{n+1} for that cell on the basis of the old values ρq_i^n 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 we know the trajectories for all the air parcels which terminate at the walls of the Eulerian CV at the new time t^{n+1} . As an example, trajectories for the 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 starting points. By connecting the starting 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

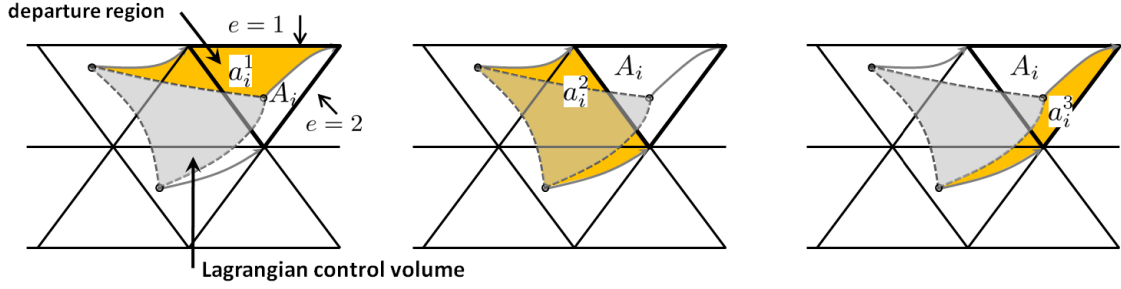


Figure 3.3.: Graphical illustration of the FFSL scheme. Black solid lines show the triangular grid, with thick solid lines highlighting the Eulerian control volume under consideration. Grey area shows the Lagrangian control volume and yellow areas show the flux areas (departure region) for each cell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

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 (3.16)

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

where ΔA_i and Δa_i denote the area of the Eulerian and Lagrangian CV, respectively (Lauritzen et al., 2011, see e.g.). $\overline{\rho q_i^n}$ 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 before, a more Eulerian rather than semi-Lagrangian viewpoint is taken in ICON. Here we keep track of the flux of mass passing the Eulerian cell walls rather than the mass in the Lagrangian CV. This is where the yellow areas in Figure 3.3 enter the game. We will refer to these as *flux areas*. Since the individual edges of the Lagrangian CV pass through the flux areas during $[t^n, t^{n+1}]$, it is exactly the mass inside in the flux areas that is swept across the Eulerian CV walls during one time step. Thus, starting from the mass ρq_i^n in the Eulerian CV and assuming that we know the shape as well as the tracer mass contained in the (yellow) flux areas, we can compute the updated value $\overline{\rho q_i^{n+1}}$.

Mathematically the scheme can be cast into the following form:

$$\overline{\rho q_i^{n+1}} = \overline{\rho q_i^n} - \frac{1}{\Delta A_i} \sum_{e=1}^{N_e} s_{ie} F_{ie} \quad , \text{with} \quad F_{ie} = \langle \overline{\rho_i^e} \rangle \iint_{a_i^e} q^n(x, y) da, \quad (3.22)$$

where F_{ie} defines the total mass crossing the e^{th} wall during Δt and a_i^e denotes the flux area for the e^{th} wall. $s_{ie}^e = \pm 1$ distinguishes inward and outward directed fluxes.

Note that this Eulerian viewpoint is fully equivalent to the semi-Lagrangian viewpoint. It can be shown (Lauritzen et al., 2011) that all areas involved in our quasi-Eulerian approach (Eulerian CV plus flux areas) sum up to the Lagrangian CV.

Basic Algorithm

The numerical algorithm which solves Eq. (3.22) for a single Eulerian CV proceeds in 4 major stages:

1. The flux area a_i^e for each cell wall is reconstructed by means of backward trajectories.
2. For each Eulerian CV the unknown tracer subgrid distribution $q(x, y, t_0)$ is estimated 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 total mass F_{ie} crossing the e^{th} wall is estimated by numerically evaluating the integral in Eq. (3.22). I.e. the estimated subgrid distribution $q(x, y, t_0)$ is integrated over the approximated departure region a_i^e by means of Gauss quadrature.
4. The sum on the r.h.s. of Eq. (3.22) is evaluated which leads to the solution $\bar{\rho}\bar{q}_i^{n+1}$.

3.5.3. 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.

The vertical transport scheme is based on the Piecewise Parabolic Method (PPM) developed by Colella and Woodward (1984). It is a finite volume scheme and thus inherently mass conserving. It makes use of a piecewise parabolic function for approximating the unknown subgrid distribution of a 1D scalar field $q(z)$. The function is forced to be continuous at cell interfaces. Its construction is based on the known cell averages \bar{q}_k . The PPM scheme bears some conceptual resemblance to the horizontal FFSL transport scheme. The basic concept is depicted in Figure 3.4.

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

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

ζ is a dimensionless coordinate which is 1 at the grid cell top and 0 at its bottom. Specific to the PPM scheme is the way how this parabola is constructed. The unknown coefficients a_i are derived from the 3 constraints,

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

which, said in words, state that the polynomial must be mass conserving, and that the polynomial equals q_u and q_l at its upper and lower end, respectively. q_u and q_l are appropriately reconstructed estimates at the upper and lower half levels and are shared between adjacent cells. By this, continuity of the reconstruction across cells is enforced. The half

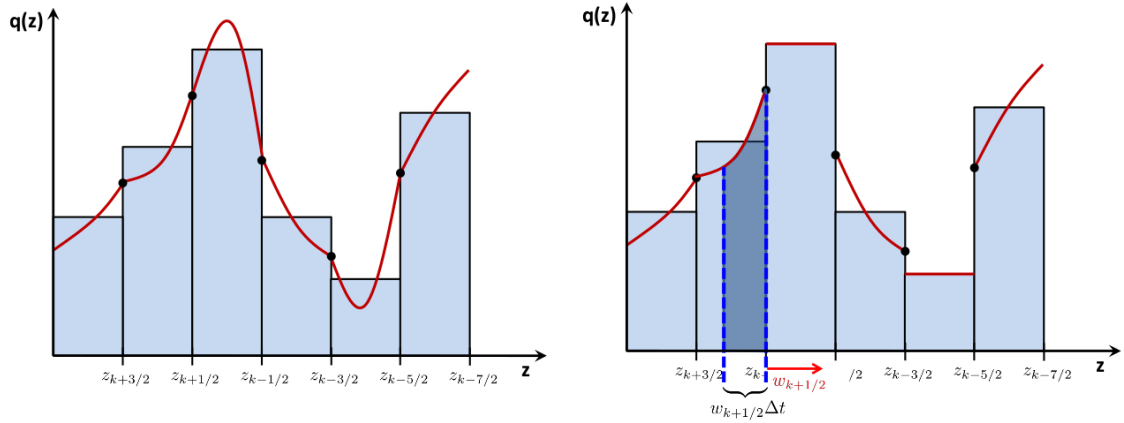


Figure 3.4.: The Piecewise Parabolic Method (PPM). Left: Unknown subgrid distribution $q(z)$ gets approximated by piecewise parabolic interpolants which are C^0 continuous at cell walls. Right: Polynomial reconstruction is filtered (optional) to render the scheme monotonous. Integration step: Sub-grid distribution is integrated over 'area' $w\Delta t$ (dark blue) in order to determine the mass which enters the k^{th} cell during Δt .

level estimate q_u 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 4 cells surrounding half level $z_{k-1/2}$ (likewise for q_l , see e.g. Zerroukat et al. (2002)). As noted in Lauritzen et al. (2011) the accuracy and stability of the scheme strongly depends on the accuracy with which the half level estimates are derived.

Step 2: As depicted in 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 with 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.

$$F_{k-1/2} = \frac{1}{\Delta t} \int_{z_{k-1/2}-w_{k-1/2}^{n+1/2}\Delta t}^{z_{k-1/2}} \rho(z)q(z)dz, \quad \text{for } w > 0 \quad (3.23)$$

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 simply given as $-w_{z_{k-1/2}}^{n+1/2}\Delta t$, where w is the vertical velocity provided by the dynamical core.

In its base version, the PPM scheme has a Courant number limitation $|Cr| \leq 1$. It can, however, be easily extended to larger Courant numbers by breaking down the computation

of the mass fluxes (3.23) into so-called integer and fractional fluxes (Lin and Rood, 1996). In its current form in ICON, the PPM scheme is stable up to $|Cr| = 5$.

3.5.4. Reduced Calling Frequency

Given that explicit time stepping is used, the continuity equation for air, the momentum and thermodynamic equation must obey the time-step restrictions imposed by the fastest waves in the system (i.e. sound waves). While the continuity equation for air is inherently coupled to the momentum equations, passive tracer transport equations can be solved in isolation given prescribed winds and air densities.

Continuity equations for passive tracers lack fast wave modes (sound and gravity waves) and, thus, have less restrictive time step limitations. Given the large number of passive tracers in state of the art climate and NWP models, significant computational cost savings can be obtained by sub-cycling the solution of the density, momentum and thermodynamic equation 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 has to be taken in order to maintain tracer-mass consistency.

In ICON, the number of times by which the integration of the continuity equation for air (and the entire dynamical core) is sub-cycled with respect to the tracer mass continuity equations is typically 5 (see Section 3.4). In order to preserve tracer-mass consistency, the time-averaged rather than the instantaneous mass flux is passed to the transport module. Thus, $\langle \bar{\rho}_i^e \rangle$ in Eq. (3.22) can be expressed in terms of the time-averaged horizontal mass flux $\langle F_{ie}^m \rangle$ as

$$\bar{\rho}_i^e = \langle F_{ie}^m \rangle \Delta t l_{ie}$$

with Δt denoting the time step for tracer transport and l_{ie} denoting the length of the e^{th} cell wall.

3.5.5. 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 has become obsolete, except for idealized cases. In real case runs, ICON takes care of initializing the correct number of tracers based on the selected physics packages.

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 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. Sub-cycling is only applied above a certain height defined by `hbot_qv_substep`. See Section 3.7.11.

FFSL and MIURA3 only 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 combine the improved stability of FFSL with the speed of MIURA3.

ivadv_tracer (namelist transport_nml, list of Integer values)

Comma separated list of integer values, specifying the type of 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
- 3** PPM

itype_hlimit (namelist transport_nml, list of Integer values)

Comma separated list of integer values, specifying the type of 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** monotonous 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 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-monotone reconstruction filter
- 2** monotone reconstruction filter
- 4** positive definite Flux Corrected Transport

Example Settings for a Standard NWP Run

Valid settings for a standard NWP run with one-moment microphysics (i.e. 5 prognostic water tracers) are depicted in Figure 3.5. Currently the mapping between the i^{th} entry in the above namelist parameters and the i^{th} entry in ICON's internal tracer list is hard coded. The order is as follows: q_v , q_c , q_i , q_r , q_s , q_g , i.e. the first entry relates to water vapour, the second one to cloud water, and so on. This can become unwieldy and error prone if more than a handful of tracers is used. The ART-package alleviate this problem by providing more elaborate way of configuring tracers which is based on XML files (see Section 10.4.3). Note, however that the configuration via XML files is restricted to ART-specific tracers, only.

PPM is the method of choice in vertical direction for all tracers. In horizontal directions MIURA is used for all tracers except vapour q_v . A somewhat more accurate (and more expensive) scheme is selected for q_v (hybrid MIURA3/FFSL). Note that sub-cycling is activated for q_v . **This is crucial for numerical stability!**

One might wonder why sub-cycling is only activated for q_v . The reason is that with standard NWP settings q_v is the only tracer for which transport is activated 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 18km) such that sub-cycling is not strictly required (see Section 3.7.11).

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

In terms of limiters, the rule of thumb is that at least a positive definite one 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 .

```
! 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

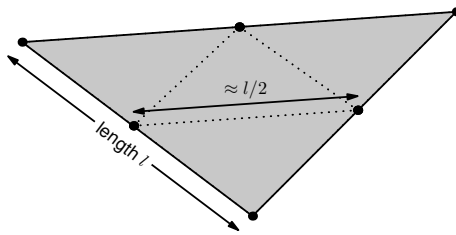
  itype_hlimit = 3, 4, 4, 4, 4 ! hor. limiter

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

Figure 3.5.: 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).

3.6. Variable Resolution Modeling

ICON has the capability for static mesh refinement in horizontal directions. This is realized through a *multi-grid approach* which means that one or more additional high resolution (child) domains can be overlaid on coarser (parent) domains. In the following we will make frequent use of the notion *parent* and *child*, in order to illustrate the relationship amongst multiple domains.



The grid spacing factor between the parent domain and the child domain is fixed to a factor of 2, i.e. each parent triangle is split into 4 child triangles. Correspondingly, in case of nested setups, the time step Δt is multiplied by a factor of 0.5 for each nesting level. Note that the time step Δt needs to be specified for the base (i.e. coarsest) domain only. The (hard-coded) adaption for nested regions is done automatically.



The multi-grid approach easily allows for switching domains on or off at runtime, as well as intertwining 1-way and 2-way nested domains. 2-way as opposed to 1-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 multi-grid approach closely resembles traditional two-way nesting and has to be distinguished from recent *uni-grid* approaches, where in special areas of interest more cells are added to an existing grid (*h-refinement*). Atmospheric models capable of static h-refinement are e.g. CAM-SE (Zarzycki et al., 2014) and MPAS-A (Skamarock et al., 2012). The basic multi-grid example shown in Figure 3.6 contains one global domain and one regional domain over Europe. It closely resembles the operational setup currently used at DWD.

3.6.1. Parent-Child Coupling

This section describes how the exchange of information between a parent and a child domain is realized, and which information (i.e. which fields) is interchanged.

As shown in Figure 3.7, a nested domain can conceptually be split into three areas: A boundary interpolation zone (red), a nudging zone (blue) and a feedback zone (light gray).

Prognostic computations are restricted to the latter two. In case of 2-way nesting, the nudging zone does not exist. Instead the feedback zone follows close upon the boundary zone. While the width of the boundary interpolation zone is fixed to 4 cell rows, the width of the nudging zone can be changed by means of the namelist switch `nudge_zone_width` (`interpol_nml`).

Once the model state on the parent domain \mathcal{M}_p has been updated from time step n to $n+1$, the states \mathcal{M}_p^n and \mathcal{M}_p^{n+1} are used to update the boundary zone on the child domain. By this, the necessary lateral boundary data (forcing) can be provided for integrating the model on the nested domain from state \mathcal{M}_c^n to \mathcal{M}_c^{n+1} .

In the feedback zone, the updated model state of the child domain 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 can benefit from the high-resolution results of the child domain.

In the nudging zone, which is only active for 1-way nesting, upscaled prognostic fields of the child domain are nudged towards the model state of the parent domain. A similar method is applied in limited area mode (LAM). It is further explained in Section 6.2.

Further details on the boundary update and the feedback mechanism are given in the following.



Figure 3.6.: Basic example of a multi-grid setup, consisting of a global ICON domain and a child domain over Europe with half grid spacing. Closely resembles configuration that is used operationally at DWD for deterministic forecasts with a horizontal grid spacing of 13 km globally and 6.5 km in the child domain.

Boundary Update: Parent → Child

The overall task of the boundary update mechanism is to provide the child domain with up-to-date lateral boundary conditions at each child time step. Boundary conditions are required for the following set of prognostic variables: $v_n, w, \rho, \theta_v, q_k$. In order to avoid that interpolated values of ρ enter the solution of the continuity equation in the dynamical core, an extended set of variables is used in the boundary update mechanism, namely: $v_n, w, \rho, \theta_v, q_k, \langle F_m \rangle$. Here, $\langle F_m \rangle = \langle \rho v_n \rangle$ denotes a temporal average of the mass flux over the dynamic substeps. Since the continuity equation is solved in flux form (see Eq. (3.5)), making use of $\langle F_m \rangle$ implies that interpolated values of ρ are not required for prognosing ρ on the child domain.

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 step 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

$$\frac{\partial \psi_p}{\partial t} = \frac{\psi_p^{n+1} - \psi_p^n}{\Delta t_p}$$

is diagnosed. Both, ψ_p^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

$$\begin{aligned} \psi_c^n &= \mathcal{I}_{p \rightarrow c}(\psi_p^n) \\ \frac{\partial \psi_c}{\partial t} &= \mathcal{I}_{p \rightarrow c}\left(\frac{\partial \psi_p}{\partial t}\right) \end{aligned}$$

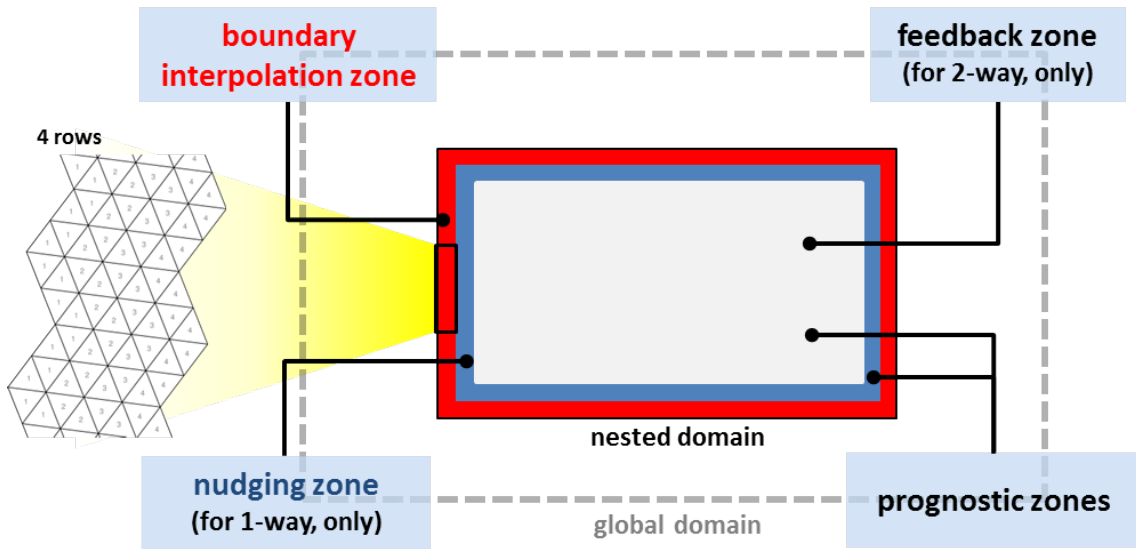


Figure 3.7.: General structure of a nested domain. Red: boundary interpolation zone consisting 4 cell rows. Blue: nudging zone, which is only active for 1-way nesting. Light-gray: feedback zone. Prognostic computations are performed in the feedback- and nudging zone.

Since the time step on the child domain Δt_c is only half of that on the parent domain, two integration steps are necessary in order to reach the model state \mathcal{M}_c^{n+1} . The interpolated tendency is used to update the lateral boundary data after the first physics step and after each dynamics substep. E.g. the lateral boundary conditions for the first and second integration step read ψ_c^n and $\psi_c^n + 0.5 \Delta t_p \partial \psi / \partial t|_c$, respectively.

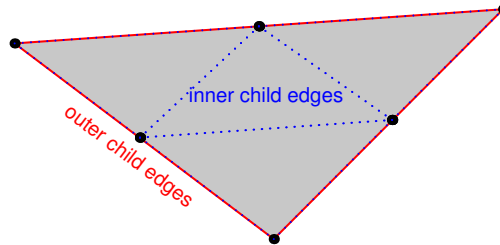
Note that in order to reduce interpolation errors above steep orography, for the thermodynamic variables ρ and θ_v perturbations from reference values, rather than the full values are interpolated to the child domain.

Regarding the interpolation operator $\mathcal{I}_{p \rightarrow c}$ we distinguish between cell based and edge-based variables. 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, [Narcovich and Ward \(1994\)](#)). The interpolated value at the child cell center is then calculated as

$$\psi_c = \psi_p + \nabla \psi_p \cdot \mathbf{d}(p, c),$$

with $\nabla \psi_p$ denoting horizontal gradient at the parent cell center, and $\mathbf{d}(p, c)$ the distance vector between the parent and child cell center. The same operator is applied to cell based tendencies.

Regarding the interpolation of edge based variables (i.e. $\partial v_n / \partial t$, $\partial F_n / \partial t$, $\langle F_m \rangle$), we distinguish between *outer child edges* that coincide with the edges of the parent cell, and *inner child edges*.



Edge-normal vector components ϕ at the inner child edges are reconstructed by means of a 5-point RBF reconstruction. The 5-point stencil comprises the edges of the two parent cells sharing the edge under consideration.

For the outer child edges a more elaborate reconstruction is applied, in order to assure that the mass flux across the parent edge is equal to the sum of the mass fluxes across the two child edges. To do so, we first reconstruct the vector quantity under consideration at the parent edge vertices using RBF, from which the gradient tangential to parent edge can be computed. The edge-normal vector component at the child edge is then computed as

$$\phi_c = \phi_p + \nabla_t \phi_p \cdot \mathbf{d}(p, c),$$

with $\nabla_t \phi_p$ denoting the gradient of the edge-normal vector component tangent to the parent edge, and $\mathbf{d}(p, c)$ the distance vector between the parent and child edge midpoints.

Since $\mathbf{d}(p, c_1) = -\mathbf{d}(p, c_2)$ holds on the ICON grid, with c_1, c_2 denoting the child cell's edge midpoints, the above mentioned mass flux consistency is ensured.

The interpolated mass fluxes valid for the current time step can be re-computed from the interpolated average mass flux $\langle F_m \rangle_c$, and the corresponding time tendency. These are prescribed at the outermost edges which separate the boundary zone from the prognostic zone. Since the continuity equation is solved in flux form (see Eq. (3.5)), this implies that interpolated values of ρ are not required for prognosing ρ on the child domain.

Feedback: Child \rightarrow Parent

If two-way nesting is selected (`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 every physics time step. In the following we will refer to this as *relaxation-type feedback*. It is applied to the prognostic variables v_n, w, θ_v, ρ as well as to the prognostic, non-sedimenting mass fractions q_v, q_c, q_i ². Let ψ denote any of the above variables. Conceptually, the method can be divided into three major steps:

1. **Upscaling:** The updated field ψ_c^{n+1} on the child domain is interpolated (upscaled) to the parent domain.
2. **Increment computation:** The difference between the solution on the parent domain ψ_p^{n+1} and the upscaled solution $\psi_{c \rightarrow p}^{n+1}$ 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 two.

By way of example, we will mathematically describe the feedback mechanism for the variables ρ and ρq_k . Other variables are handled in a very similar manner.

The feedback mechanism for ρ can be cast into the following form:

$$\rho_p^* = \rho_p^{n+1} + \frac{\Delta t_p}{\tau_{fb}} (\mathcal{I}_{c \rightarrow p}(\rho_c^{n+1}) - \rho_p^{n+1}) \quad (3.24)$$

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 capture synoptic-scale features. The relaxation time scale can be adjusted by means of the namelist variable `fbk_relax_timescale` (`gridref_nml`). Finally, $\mathcal{I}_{c \rightarrow p}(\rho_c^{n+1})$ denotes a horizontal interpolation operator which upscales the child domain solution to the parent domain.

²Note 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 8.4.

The interpolation operator is defined as

$$\mathcal{I}_{c \rightarrow p}(\rho_c^{n+1}) = \sum_{i=1}^4 \alpha_i \left(\rho_{c,i}^{n+1} - \Delta\rho_{corr} \right),$$

with α_i denoting the interpolation weight for the i th child cell. It basically performs a proper weighting of those 4 child cell center values that are connected to the parent cell under consideration. Both bilinear and area-weighted interpolation methods are available, with the former being the default choice. In order to account for differences in the vertical position of the child and parent cell circumcenters, the correction term $\Delta\rho_{corr}$ has been introduced. 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

$$\Delta\rho_{corr} = (1.05 - 0.005 \mathcal{I}_{c \rightarrow p}(\theta_v'^{n+1})) \Delta\rho_{ref,p},$$

with the difference in the reference density field

$$\Delta\rho_{ref,p} = \sum_{i=1}^4 (\alpha_i \rho_{ref,c,i}) - \rho_{ref,p},$$

and the upscaled perturbation value of θ_v denoted by

$$\mathcal{I}_{c \rightarrow p}(\theta_v'^{n+1}) = \sum_{i=1}^4 \alpha_i \left(\theta_{v,c,i}^{n+1} - \theta_{v,ref,i} \right).$$

The term $\Delta\rho_{ref,p}$, which is purely a function of the parent-child height difference can be regarded as a first order correction term. In order to minimize the remaining mass drift, the empirically determined factor $(1.05 - 0.005 \mathcal{I}_{c \rightarrow p}(\theta_v'^{n+1}))$ 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.

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

$$(\rho q_x)_p^* = (\rho q_x)_p^{n+1} + \frac{\Delta t_p}{\tau_{fb}} \left[\mathcal{I}_{c \rightarrow p}((\rho q_x)_c^{n+1}) - (\rho q_x)_p^{n+1} \right] \quad (3.25)$$

with

$$\mathcal{I}_{c \rightarrow p}((\rho q_x)_c^{n+1}) = \sum_{i=1}^4 \alpha_i \left(\rho_{c,i}^{n+1} - \Delta\rho_{corr} \right) q_{x,c,i}$$

Mass fractions are re-diagnosed thereafter:

$$q_{x,p} = \frac{(\rho q_x)_p^*}{\rho_p^*}$$

When summing Eq. (3.25) over all partial densities, Eq. (3.24) for the total density is recovered. Feedback of other prognostic variables is done in a very similar manner. Note, however, that for θ_v the upscaling is done for the perturbation from the reference state, in order to reduce numerical errors over steep mountains. In the case of v_n some numerical diffusion is added to the resulting feedback increment in order to damp small-scale noise.

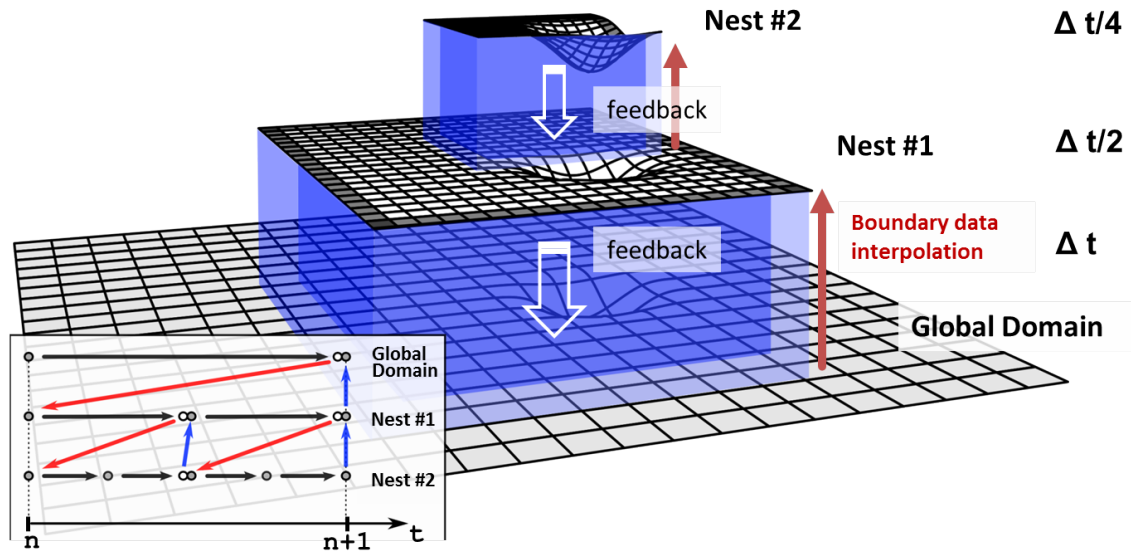


Figure 3.8.: Schematic of a global domain with two two-way nested domains nested into each other. The processing sequence for integrating the model from time step n to $n + 1$, is shown in the flowchart at the lower left. Red arrows indicate boundary data interpolation from parent to child, blue arrows indicate the feedback operation from child to parent and black arrows indicate the time integration on the respective domain.

3.6.2. Processing Sequence

So far, we concentrated on the information exchange between individual parent and child domains. Nothing has been said about the processing sequence if more than one nested domain, or even repeatedly nested domains are involved. Figure 3.8 provides a common example where a global domain is accompanied by two two-way nested domains nested into each other. The global domain is schematically depicted at the bottom, whereas the nested domains are vertically staggered on top of it. The two blueish regions show the boundary interpolation zone of the nests and the feedback zone. The integration time step on the global domain is Δt , whereas the time step is reduced by a factor of 2 when moving to the next higher grid level.

The processing sequence for integrating the model from time step n to $n + 1$ is shown in the flowchart at the lower left. The various domains are ordered top to bottom. The dots indicate the model state \mathcal{M}_x for the different domains, red and blue arrows indicate boundary data interpolation and feedback, respectively, and black arrows indicate time integration.

The basic processing sequence is as follows:

- First, a single time step with Δt is performed on the global domain which results in an updated model state indicated by the white circle.
- It is followed by boundary data interpolation to the first nested domain (red arrow). Then, the model is integrated on nested domain 1 over the time interval $\Delta t/2$.

- Since there exists a second nested domain within nest 1, lateral boundary fields based on the model state $\mathcal{M}_{c1}^{n+1/2}$ are interpolated to the second nested domain. Then, the model is integrated on the nested domain 2 over two times the time interval $\Delta t/4$, resulting in the model state $\mathcal{M}_{c2}^{n+1/2}$.
- Now, feedback is performed from nested domain 2 back to nested domain 1 (blue arrow), which results in an updated model state $\mathcal{M}_{c1}^{n+1/2}$ on nested domain 1 (gray circle). Now, the model is again moved forward in time on nested domain 1 to reach \mathcal{M}_{c1}^{n+1} .
- This is followed by a second lateral boundary data interpolation for the nested domain 2. Finally, nested domain 2 is integrated in time again, to reach its final state \mathcal{M}_{c2}^{n+1} . As a last step, feedback is performed from nested domain 2 to nested domain 1, followed by feedback from nested domain 1 to the global domain.

An alternative way of visualizing the processing sequence is shown in Figure 3.9. Again, the different grid levels are indicated by horizontal lines with the coarsest grid at the top and the finest one at the bottom. The model state is indicated by a white circle. Every black circle indicates that the model has been integrated in time over one domain-specific time step. Red and blue arrows again denote boundary interpolation and feedback. Note that in cases where only one domain per grid level exists (Figure 3.9a), the processing sequence resembles a W-cycle known from classic multi-grid algorithms for solving elliptic PDEs.

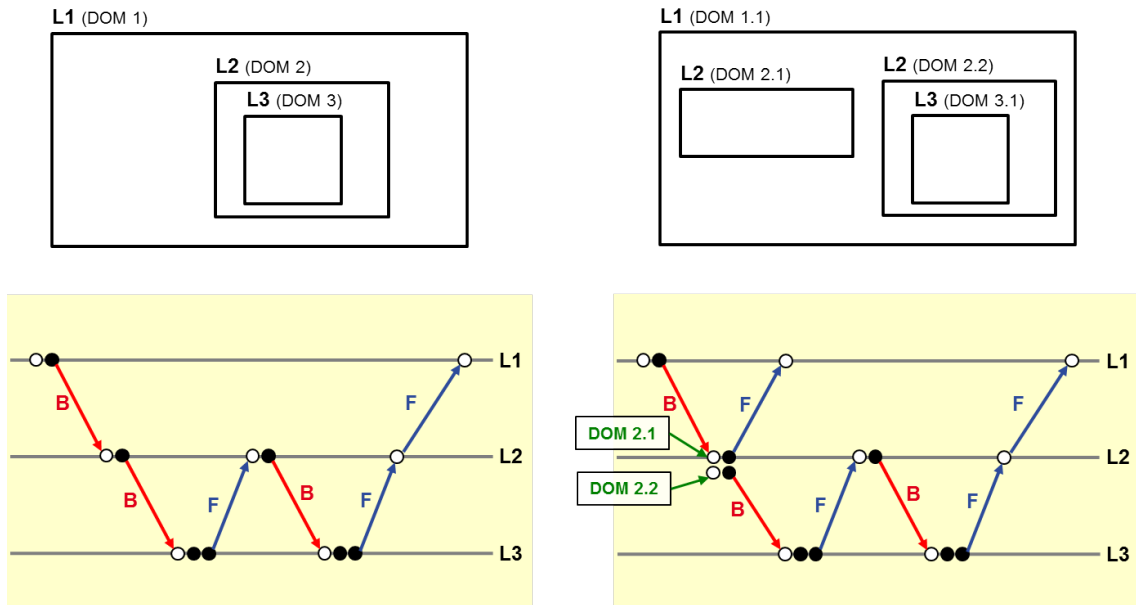


Figure 3.9.: Processing sequence (bottom row) for two generic multi-domain setups (top row). Horizontal lines represent the different grid levels and white circles denote the current model state. A filled black circle, instead, indicates a time integration step. Boundary interpolation and feedback are visualized through red and blue arrows, respectively. Setup (a) is identical to the one shown in Figure 3.8. Setup (b) differs from setup (a) by an additional nested domain on grid level 2.

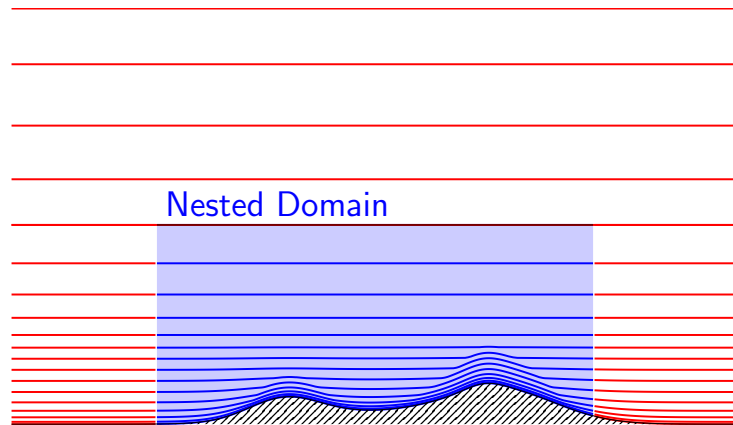


Figure 3.10.: Illustration of ICON’s vertical nesting. Note that even when choosing different numbers of vertical levels, vertical layers between the nested and the parent domain must match. Therefore, the nested domain may only have a lower top level height.

3.6.3. Vertical Nesting

The model top height can be chosen individually for each domain, with the constraint that the height of the parent domain must be larger or at least equal to the height of the child domain. Thereby it is e.g. possible to combine a global domain which extends into the mesosphere with one or several child domains that only extend into the lower stratosphere, see Figure 3.10.

However, it is not possible to choose different vertical level distributions for individual domains. The level distribution is always defined by the global domain. Lower model top heights in child domains are thus realized by simply neglecting a certain number of levels at the model top. Vertical *refinement* in the sense that vertical resolution is locally increased in a child domain is so far not possible.

In order to reduce the top height for individual 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 3.3 for details.

If vertical nesting is activated, boundary conditions must be provided for all prognostic variables at the uppermost half level of the nested domain. Several measures are necessary in order to avoid excessive reflections of vertically propagating sound and gravity waves from the domain’s model top. Further details, however, are beyond the scope of this tutorial.

3.7. 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 3.3 contains a summary of physical parameterizations available in ICON (NWP-mode). In what follows, an outline (executive summary) of the parameterization schemes is given.

3.7.1. Radiation

The calculation of radiation is performed with RRTM (Rapid Radiative Transfer Model, [Mlawer et al., 1997](#)). RRTM operates on 30 spectral bands with 16 bands in the long-wave spectrum and 14 bands in the shortwave spectrum. It utilizes the correlated-k method that strongly reduces computational costs with an accuracy comparable to more detailed line-by-line models.

3.7.2. Cloud Microphysics

Section author

A. Seifert, DWD Physical Processes Division

In the exercises for Chapter 8 (see p. 210) we will investigate ICON's physical parameterizations by means of a custom diagnostic quantity. We restrict ourselves to the cloud microphysics parameterization, where some additional background information will be of interest.

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.

Process	Scheme	Settings
Radiation	<input checked="" type="checkbox"/> 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
	PSRAD Pincus and Stevens (2013)	inwp_radiation=3
Non-orographic gravity wave drag	<input checked="" type="checkbox"/> Wave dissipation at critical level Orr et al. (2010)	inwp_gwd=1
Sub-grid scale orographic drag	<input checked="" type="checkbox"/> Lott and Miller scheme Lott and Miller (1997)	inwp_sso=1
Cloud cover	<input checked="" type="checkbox"/> Diagnostic PDF M. Köhler et al. (DWD)	inwp_cldcover=1
	All-or-nothing scheme (grid-scale clouds)	inwp_cldcover=5
Microphysics	<input checked="" type="checkbox"/> Single-moment scheme Doms et al. (2011), Seifert (2008)	inwp_gscp=1, 2
	Double-moment scheme Seifert and Beheng (2006)	inwp_gscp=4
Convection	<input checked="" type="checkbox"/> Mass-flux shallow and deep Tiedtke (1989), Bechtold et al. (2008)	inwp_convection=1
Turbulent transfer	<input checked="" type="checkbox"/> Prognostic TKE (COSMO) Raschendorfer (2001)	inwp_turb=1
	EDMF-DualIM (<u>E</u> ddy- <u>D</u> iffusivity/ <u>M</u> ass- <u>F</u> lux) Köhler et al. (2011), Neggers et al. (2009)	inwp_turb=3
	3D Smagorinsky diffusion (for LES)	inwp_turb=5
Land	<input checked="" type="checkbox"/> Tiled TERRA Schrodin and Heise (2002)	inwp_surface=1
	<input checked="" type="checkbox"/> Flake: Mironov (2008)	llake=.TRUE.
	<input checked="" type="checkbox"/> Sea-ice: Mironov et al. (2012a)	lseaiice=.TRUE.

Table 3.3.: 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 .

ICON does also provide a double-moment microphysics scheme (`inwp_gscp=4`), 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. Currently all choices regarding, e.g., cloud condensation and ice nuclei, particle geometries and fall speeds etc. have to be set in the code itself and can not be chosen via the ICON namelist.

3.7.3. 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.` in the namelist `nwp_phy_nml`).

The implemented scheme represents a spin-off 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 \(2017\)](#).

3.7.4. 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 vapour 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 diagnostic output variables q_{v_dia} , q_{c_dia} , q_{i_dia}). They are not equal to their prognostic grid-scale equivalents (standard output variables q_v , q_c , q_i), 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.

3.7.5. Turbulent Diffusion

Section author

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 $\theta_l = \theta - \frac{L_c}{c_{pd}} q_c$), 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 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_nm1`). 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

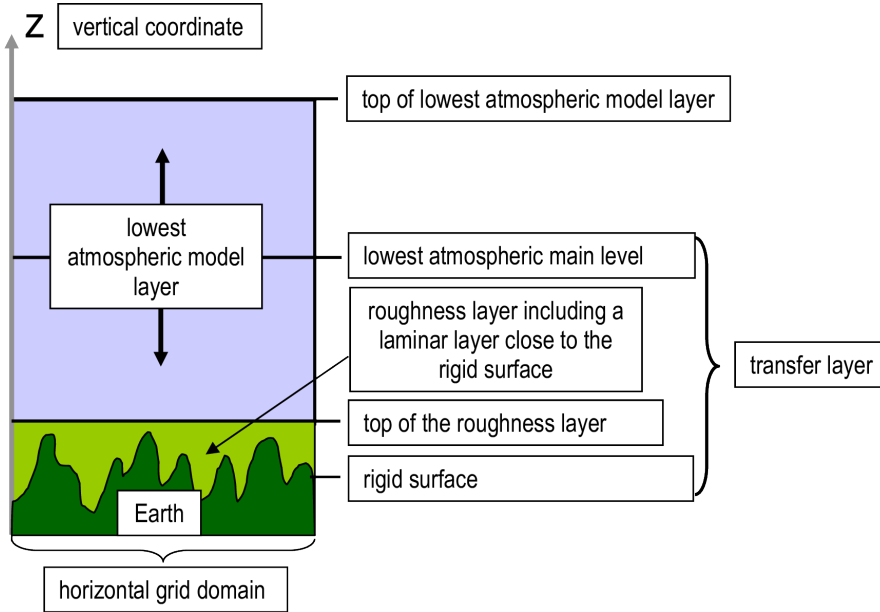


Figure 3.11.: Surface layer as described in the parameterization TURBTRAN of Raschendorfer (2001).

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. Although 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.

DualM EDMF for Turbulence and Shallow Convection.

The *Eddy Diffusivity Mass Flux* (EDMF) approach - operational at ECMWF (Köhler et al., 2011) - is based on the decomposition of the turbulent transports proposed by Siebesma and Cuijpers (1995) into eddy diffusivity and mass-flux components. This is based on the idea of unifying the eddy diffusivity and mass-flux concepts that are used in many NWP and climate models within one unified solver. When generalizing to multiple updrafts M_i one arrives at the following equation for the flux of a scalar quantity ϕ :

$$\overline{w'\phi'} = -K \frac{\partial \bar{\phi}}{\partial z} + \sum_i M_i (\bar{\phi}_i^u - \bar{\phi}).$$

Here, the mass flux is $M_i = a_i(\bar{w}_i^u - \bar{w})$ and K is the diffusion coefficient. The averaging operators $\bar{\phi}_i^u$ and $\bar{\phi}^e$ act on the updraft and environment fractions, respectively.

To arrive at this equation two assumptions are applied: (i) updraft fraction is small $a \ll 1$ and (ii) the flux within the environment $\overline{w'\phi'_e}$ can be approximated by K-diffusion.

The *DualM* framework by [Neggers et al. \(2009\)](#) postulates that two mass-fluxes are sufficient to treat the transition from a dry boundary layer to stratocumulus and shallow cumulus. In particular one dry mass-flux stops at cloud base, while the second moist mass-flux reaches to cloud top. The continuous area partitioning between the dry and moist updraft is a function of moist convective inhibition above the mixed layer top. Updraft initialization is a function of the updraft area fraction and is therefore consistent with the updraft definition. It is argued that the model complexity thus enhanced is sufficient to allow reproduction of various phenomena involved in the cloudsubcloud coupling, namely (i) dry countergradient transport within the mixed layer that is independent of the moist updraft, (ii) soft triggering of moist convective flux throughout the boundary layer, and (iii) a smooth response to smoothly varying forcings, including the reproduction of gradual transitions to and from shallow cumulus convection.

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 $\text{Ri} = \frac{N^2}{|S|}$ 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`).

3.7.6. 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-metre 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, 2012b). *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_thrhd` (`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. (2012b). 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.

3.7.7. 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. \(2012a\)](#), 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 3.7.6). 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. \(2012a\)](#), 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.

3.7.8. Land-Soil Model TERRA

Section author

J. Helmert, DWD Physical Processes Division

The soil-vegetation-atmosphere-transfer component TERRA (Schrodin and Heise, 2002, Heise et al., 2006) 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 ([Schulz et al., 2016](#)).
- 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
- 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.

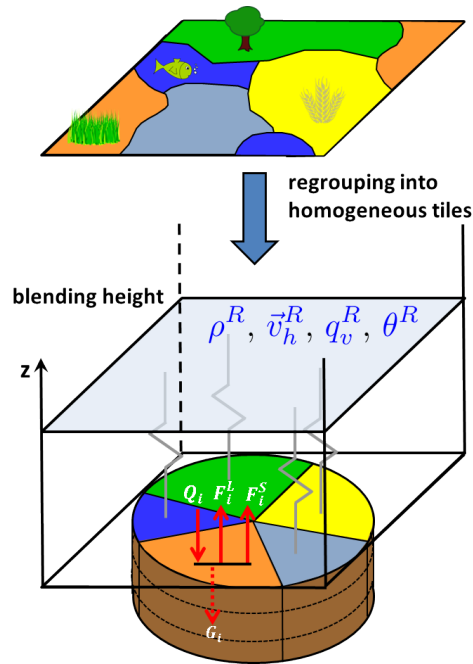


Figure 3.12.: Tile approach for a grid cell containing various surface types. Patches of the same surface type within a grid box are regrouped into homogeneous classes (tiles) for which the soil and surface parameterizations are run separately.

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.

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 setting `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.

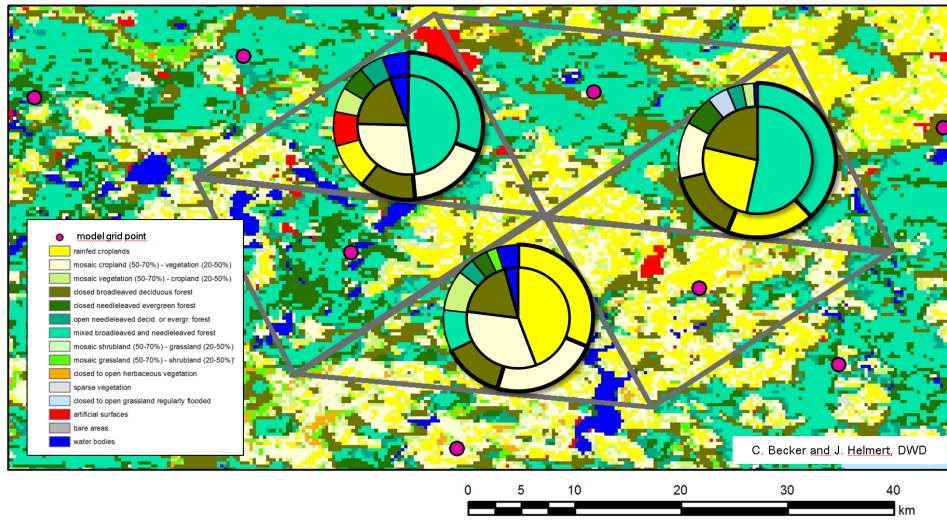


Figure 3.13.: Tile generation for $n_{\text{tiles}}=3$, for the case of a heterogeneous land surface. Outer circles show the fractional areas covered by the respective surface-type for a given grid cell. Inner circles show the selected tiles. Please note the re-scaling of fractional areas in the inner circle.

The process of tile generation in ICON works as follows:

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 n_{tiles} (typically about 3) dominating ones are represented by tiles, 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 $n_{\text{tiles}} > 1$, a tile coldstart becomes necessary. This can be done by setting `l_tile_init=.TRUE.` and `l_tile_coldstart=.TRUE.` in the namelist `initicon_nml`. Each tile is then initialized with the same cell averaged value. Note that `l_tile_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!

3.7.9. Details of ICON's Physics-Dynamics Coupling

For efficiency reasons, a distinction is made between so-called *fast-physics processes* (those 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. The relationship between the different time steps has already been explained in Section 3.4.

Fast-physics processes are calculated at every physics time step 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 saturation adjustment \rightarrow surface transfer scheme \rightarrow land-surface scheme \rightarrow boundary-layer / turbulent vertical diffusion scheme \rightarrow microphysics scheme, and again saturation adjustment in order to enter the slow-physics parameterizations with an adjusted state. The exceptions from the above-mentioned sequential splitting are the surface transfer scheme and the land-surface scheme, which 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 3.14.

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 convection, subgrid-scale cloud cover, radiation, non-orographic and orographic gravity wave drag are considered as slow processes. Typically, these processes are integrated with time steps longer than the (fast) physics time step. 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 $\partial q_x / \partial t$ with $x \in [v, c, i]$ are passed to the dynamical core and remain constant between two successive calls of the parameterization (Figure 3.14). Since ICON solves a prognostic equation for π rather than T , the temperature tendencies are converted into tendencies of the Exner function, beforehand. Rather than treating the moisture tendencies as a forcing term during tracer advection, they are treated in a time-split manner and added to the updated moisture variables thereafter.

3.7.10. 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

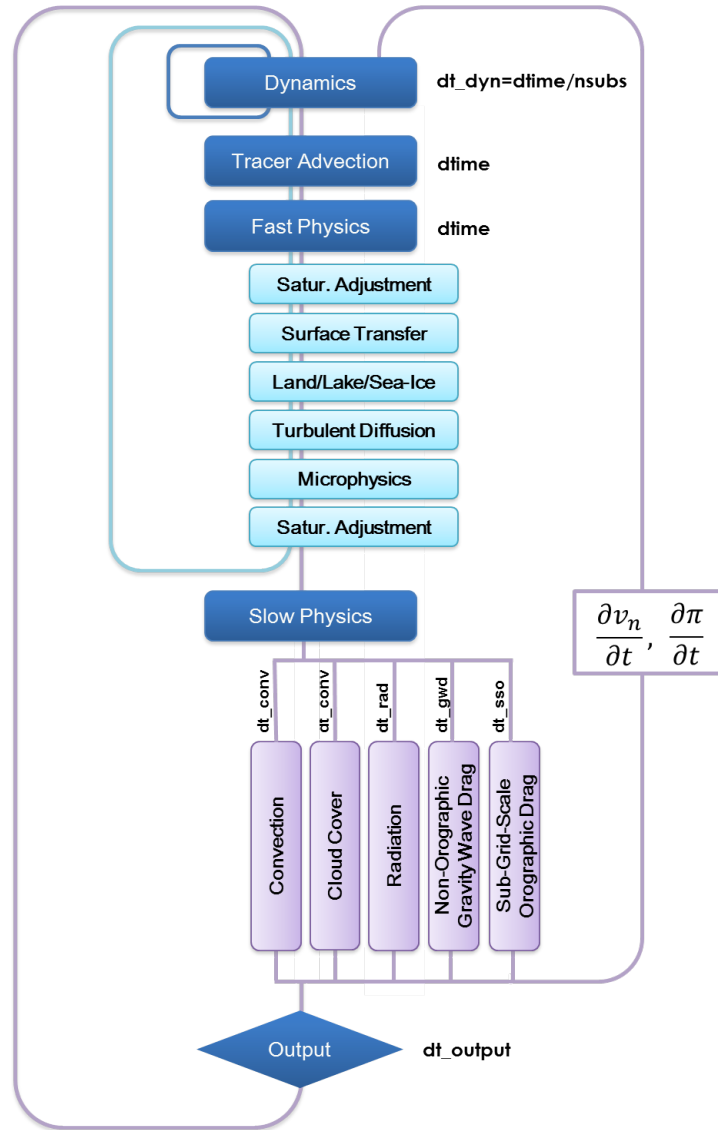


Figure 3.14.: Coupling of the dynamical core and the NWP physics package. Processes declared as fast (slow) are treated in a time-split (process-split) manner.

pressure derived directly from the prognostic model variables³. 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.

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

³Note 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.

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 .

3.7.11. Reduced Model Top for Moist Physics

A notable means for improving the efficiency of ICON is depicted in Figure 3.15. The switch

```
htop_moist_proc (namelist nonhydrostatic_nml, floating-point value)
```

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 3.4 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 3.5.5 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 (see e.g., Chapter 10).

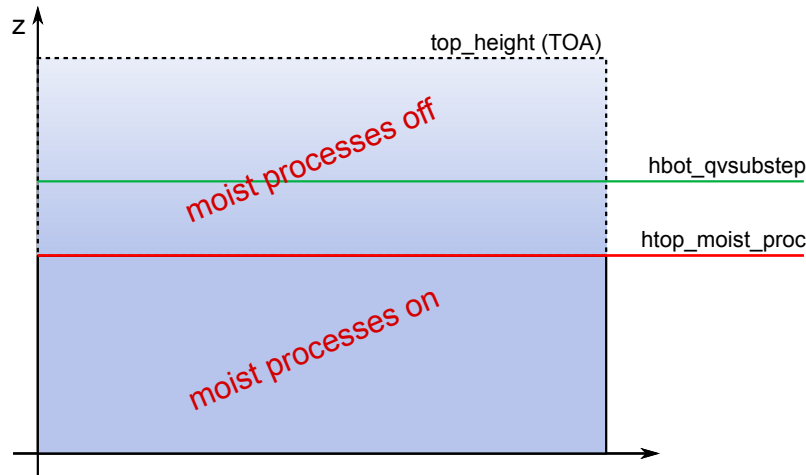


Figure 3.15.: Moist physics are switched off above `htop_moist_proc`, while tracer substepping is switched on above `hbot_qvsubstep`. (Remark: `hbot_qvsubstep` is allowed to be lower than `htop_moist_proc`)

3.8. 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 3.16:

- 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 long-wave 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 higher)

`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.

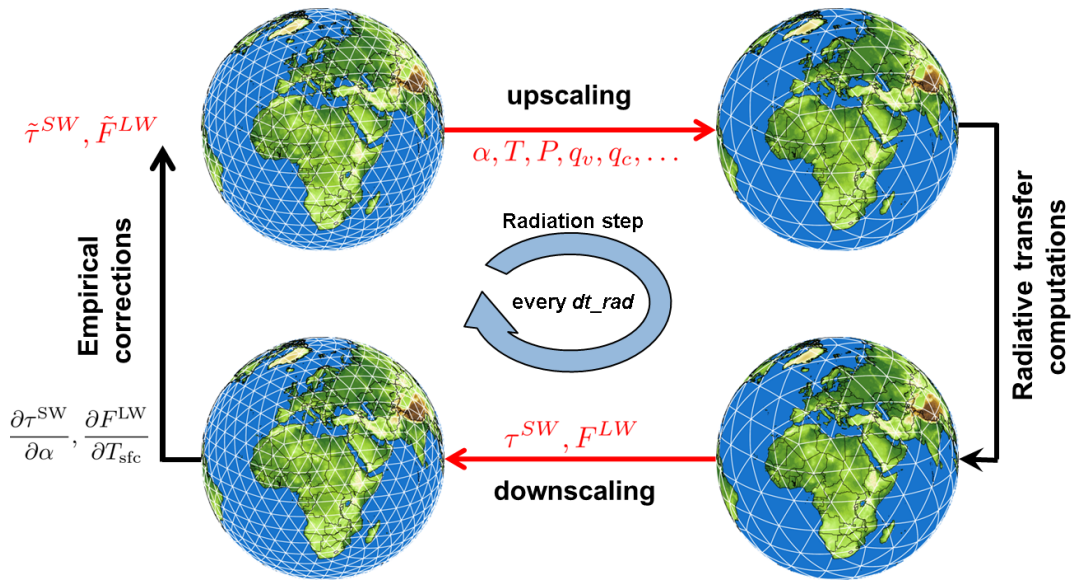


Figure 3.16.: Schematic showing how radiation is computed on a reduced (coarser) grid.

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 2.1.3. Internally, the coarse radiation grid is denoted by the domain index 0.

4. Running Idealized Test Cases

As opposed to real case runs, idealized test cases typically do not require any external parameter or analysis fields for initialization. Instead, initial conditions are computed within the ICON model itself, based on analytical functions. These are either evaluated point-wise at cell centers, edges, or vertices, or are integrated over the triangular control volume.

The ability to run idealized model setups serves as a simple possibility 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-implemented 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 below.

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

`icon/doc/Namelist_overview.pdf`

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 a set of namelist parameters described in Section 3.3.

4.2. Jablonowski-Williamson Baroclinic Wave Test

From the set of available idealized test cases we choose the Jablonowski-Williamson baroclinic wave test and walk through the procedure of configuring and running this test in ICON.

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. Thus, 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 instabilities become more and more apparent especially near the pentagon points (see Section 2.1), which are 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.

In Jablonowski et al. (2008) it is suggested to add a variety of passive tracers to the baroclinic wave test case, in order to investigate the general behavior of the advection algorithm. Questions that could be addressed are

- whether the advection scheme is monotone or positive-definite,
- how accurate or diffusive the advection scheme is,
- whether a constant tracer distribution is preserved (which checks for tracer-air mass consistency).

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.

4.2.1. Main Switches for the Idealized Test Case

This section explains several namelist groups and main switches that are necessary for setting up an idealized model run. Settings for the Jablonowski-Williamson test case are given in red.

Namelist `run_nml`:

`ltestcase = .TRUE.` (namelist `run_nml`, logical value)

This parameter must be set to `.TRUE.` for running idealized test cases.

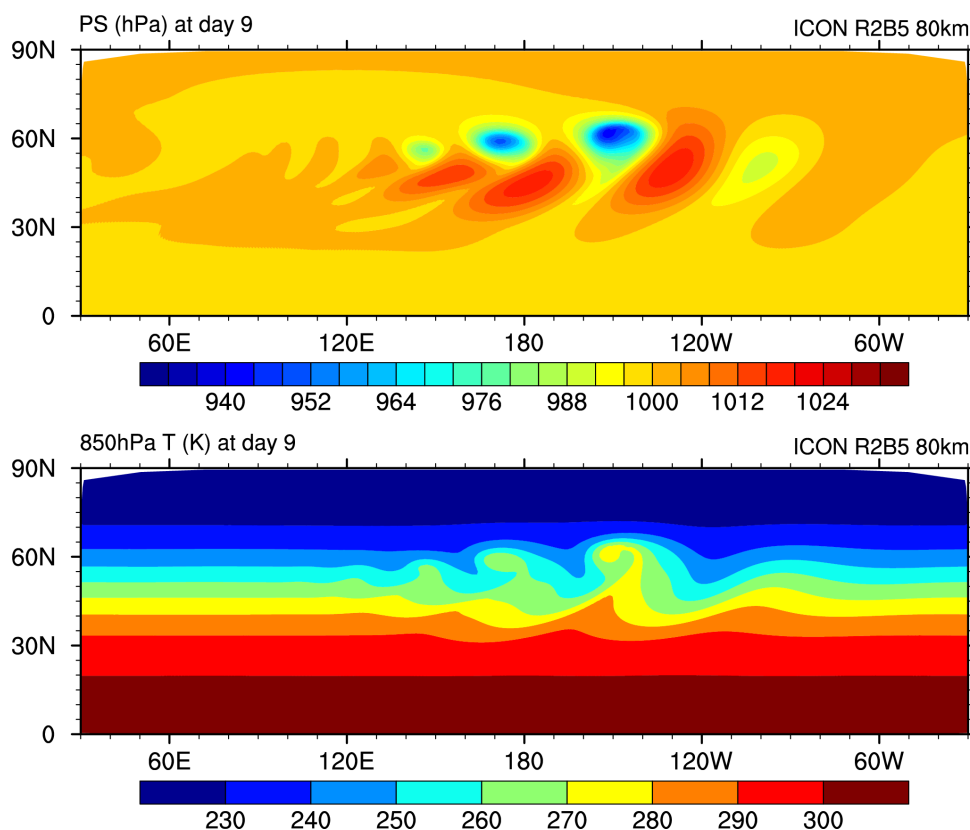


Figure 4.1.: Surface Pressure and 850 hPa Temperature at day 9 for the Jablonowski-Williamson test case on a global R2B5 grid.

`iforcing = 0` (**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 (see `nwp_phy_nml`) 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 3.3. In general, the setting of `iforcing` depends on the selected test case.

`ldynamics = .TRUE.` (**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 physics packages with prescribed dynamical forcing.

`ltransport = .FALSE./ .TRUE.` (**namelist run_nml, logical value**)

Main switch for the transport of passive tracers. If set to `.TRUE.`, transport is switched on and details of the transport schemes can be controlled via `transport_nml` (see Section 3.5.5 for additional help). If set to `.FALSE.`, transport of passive tracers is switched off completely.

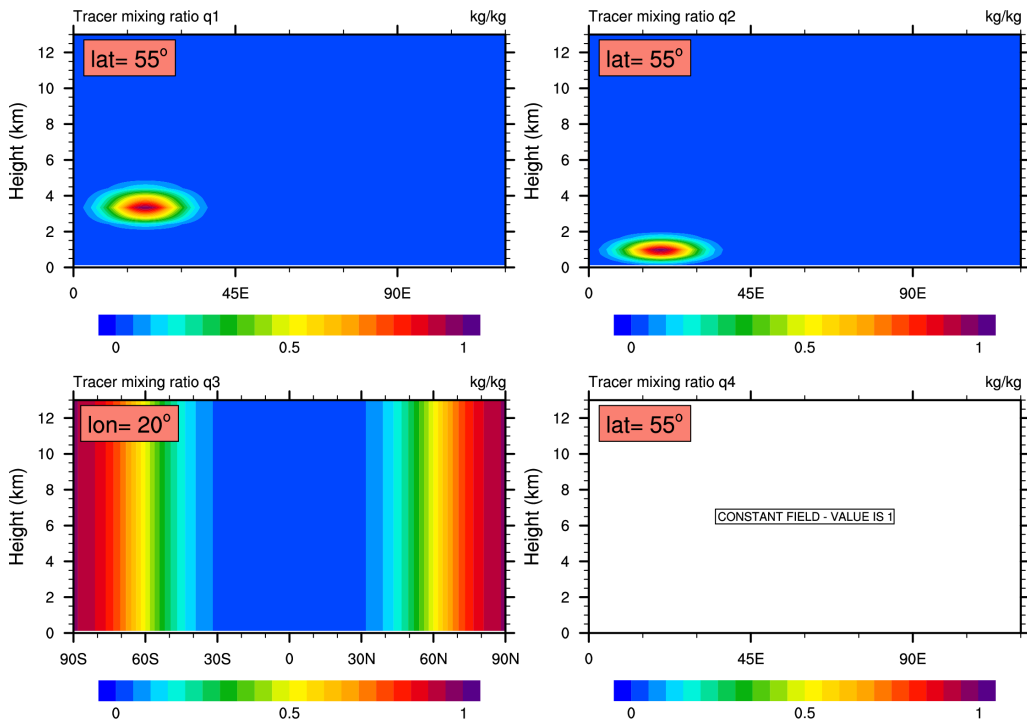


Figure 4.2.: Initial tracer distributions which are available for the Jablonowski-Williamson test case. Tracer $q3$ only depends on the latitudinal position, and tracer $q4$ is constant.

Namelist `nh_testcase_nml`:

`nh_test_name = 'jabw'` (**namelist** `nh_testcase_nml`, **string parameter**)

Main switch for selecting a test case. `nh_test_name='jabw'` selects the Jablonowski-Williamson baroclinic wave test case.

Namelist `extpar_nml`:

`itopo = 0` (**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 required for model initialization. Instead, all initial conditions are computed within the ICON model itself. Usually, `itopo` should be set to 0 for running idealized test cases.

4.2.2. Specifying the Computational Domain(s)

In the following we will explain how the Jablonowski-Williamson test case can be set up for a global domain only and, in a second step, for a global domain with nests.

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 example below).

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

Comma-separated list of integer values. For each domain, the grid ID of its parent domain must be specified.

Grid IDs start with 1 and correspond to the list given by the namelist parameter `dynamics_grid_filename`. The grid ID 0 indicates that a domain has no parent domain (i.e. the global domain). Therefore the parent grid file for domain i is given by `dynamics_grid_filename(dynamics_parent_grid_id(i))`.

Additional explanation and an example are given in Section 2.1.4 (“Step 3: Sub-domain Name and Parent Grid ID”).

Examples

Example 1: Settings for a global Jablonowski-Williamson test run without nest

```
dynamics_grid_filename = 'icon_grid_0014_R02B05_G.nc'
dynamics_parent_grid_id = 0
num_lev = 40
```

Example 2: For a global Jablonowski-Williamson test run including a single nest

```
dynamics_grid_filename =
'icon_grid_0014_R02B05_G.nc', 'icon_grid_0014_R02B05_N06_1.nc'
dynamics_parent_grid_id = 0,1
num_lev = 40,40
```

Example 3: Settings for a global Jablonowski-Williamson test run including two nests on the same nesting level (i.e. combination of Figure 4.3)

```
dynamics_grid_filename =
'icon_grid_0014_R02B05_G.nc', 'icon_grid_0014_R02B05_N06_1.nc',
'icon_grid_0014_R02B05_N06_2.nc'
dynamics_parent_grid_id = 0,1,1
num_lev = 40,40,40
```

Some additional information on the grid file naming convention can be found in Section 2.1.

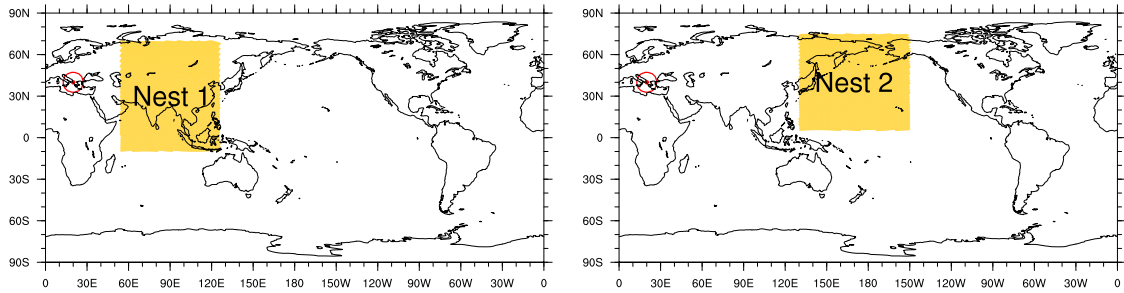


Figure 4.3.: Location of available nests for the baroclinic wave test case. The perturbation triggering the baroclinic wave is centered at (20°E, 40°N) (red circle).

4.2.3. Integration Time Step and Simulation Length

The integration time step and simulation length are defined as follows:

Namelist `run_nml`:

`dttime = 720` (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 will be given in Section 3.4.

`nsteps = 1200` (namelist `run_nml`, integer value)

Number of time steps.

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. For example, the run script that is used in Exercise D.4.1 contains three output namelists. The details of the model output specification are discussed in Section 7.1.

5. Running Real Data Test 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,
- 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.

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 Section 4.1, we will concentrate on their settings for real data runs. Settings appropriate for the exercises on this subject (see Ex. D.5.1) are highlighted in red.

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. This is done 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.



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`:

<code>time step</code>	<code>dttime</code>	<code>modelTimeStep</code>
------------------------	---------------------	----------------------------

Namelist `time_nml` (left) and `master_time_control_nml` (right):

<code>experiment start</code>	<code>ini_datetime_string</code>	<code>experimentStartDate</code>
<code>experiment stop</code>	<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, `dttime` must be specified in seconds.

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 3.5.5).

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 4.2.2 and the examples therein).

`dynamics_parent_grid_id` (namelist `grid_nml`, list of string parameters)

Comma-separated list of integer values. For each domain, the grid ID of its parent domain must be specified.

Grid IDs start with 1 and correspond to the list given by the namelist parameter `dynamics_grid_filename`. Additional explanation and an example are given in Section 2.1.4 (“Step 3: Sub-domain Name and Parent Grid ID”). If a domain has no parent domain this is indicated by the value 0 (e.g. the global domain).

`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(s) of the grid file(s) to be used for radiation must be specified here. See Section 3.8 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. See also Section 5.1.2 for additional keywords.

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`

- initialization mode `init_mode = 4` starts from interpolated COSMO-DE or ICON/IFS forecasts for limited area runs. Limited area simulations are a feature which will be addressed in Chapter 6.
- start from interpolated *initialized* ICON analysis with subsequent vertical remapping: `init_mode=7`

The most relevant modes are mode 1, 2, 5 and 7. They 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`.

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 2.6). Nevertheless, this mode is described for completeness. The process of obtaining the uninitialized DWD analysis for non-incremental update and its content is described in Section 2.2.1 (Option 2).

Model initialization is basically controlled by the following three 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.

In that case 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. See Table 2.2 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 `R α Byy` and the domain number `DOM ii` . 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:

<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>Rα</code> (single digit)
<code><nroot0></code>	grid root division <code>R$\alpha\alpha$</code> (two digits)
<code><jlev></code>	grid bisection level <code>Byy</code> (two digits)
<code><idom></code>	domain number (two digits).

5.1.3. Starting from Uninitialized DWD Analysis with IAU

As described in Section 2.2.1, 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.

The process of obtaining the uninitialized analysis for IAU and its content is described in Section 2.2.1 (Option 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 contains the analysis fields (mostly increments) produced by the assimilation process. See Table 2.1 for a 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.

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.

`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 (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 2.2.1 and depicted in Figure 5.1, 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.

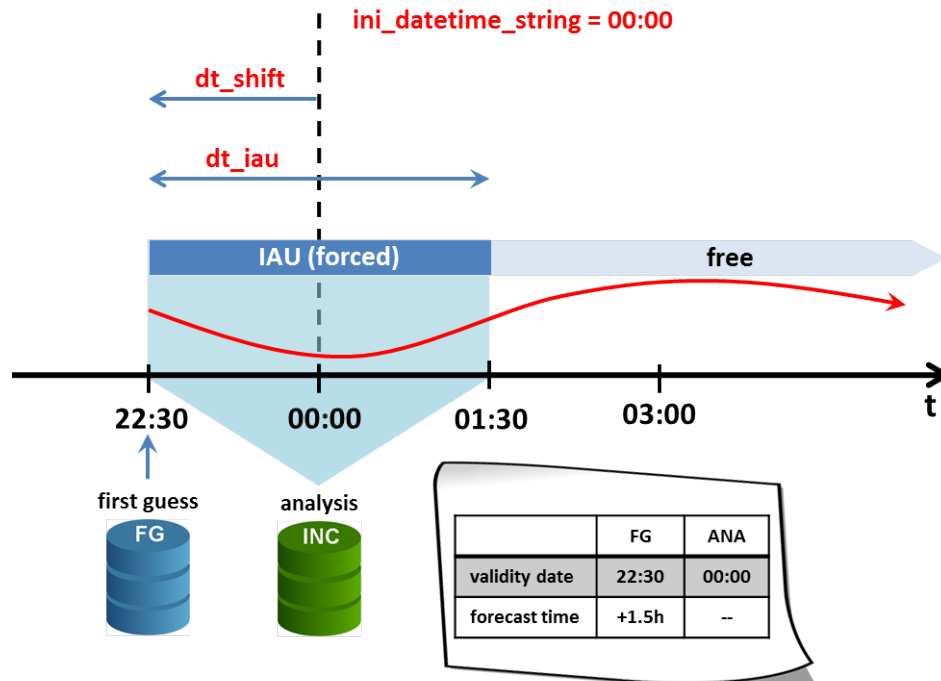


Figure 5.1.: Schematic illustrating typical settings for a global ICON forecast run starting from a DWD analysis **with IAU** at 00 UTC. IAU is performed over a 3 h time interval (`dt_iau`), with the model start being shifted ahead of the nominal start date by 1.5 h (`dt_shift`). Validity date of first guess and analysis is 22:30 UTC and 00 UTC, respectively.

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 2.6).

The process of obtaining the initialized analysis and its content is described in Section 2.2.1 (Option 3).

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:

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. 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.

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. Note that for this initialization mode only input data in NetCDF format are supported. 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`.

`start_time` (namelist `grid_nml`, list of real values)

Comma-separated list of integer 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 integer 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 preprocessing aspects regarding the regional mode have already been discussed in Section 2.3. Based on these prerequisites the exercises in this chapter (see Ex. D.6.1) will explain 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 3.6.1 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 and the limited area model may run on different computer sites. Often they also 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 3.8. This serves to simplify the discussion at this point.

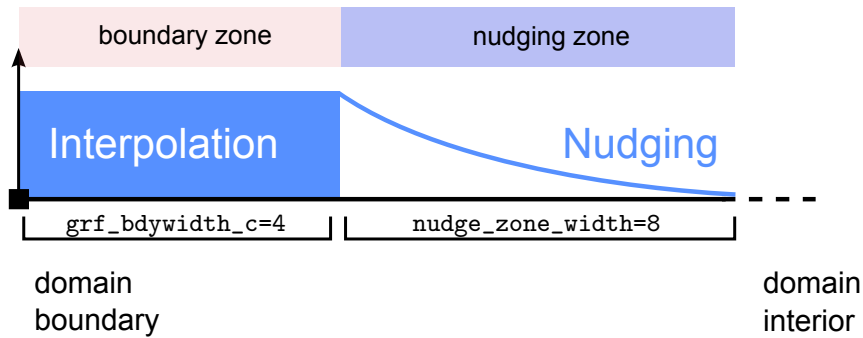


Figure 6.1.: Default structure of the boundary zone and accompanying nudging zone in limited-area mode. The nudging coefficient decays exponentially with increasing distance from the boundary zone.

- 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 3.6.3). 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 and 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 the externally specified boundary data. Figure 6.1 schematically depicts the partitioning of the near boundary region into a so-called *boundary zone* and an adjacent *nudging zone*. The sponge layer is equivalent to the nudging zone. In the boundary zone, which has a fixed width of 4 cell rows, externally supplied boundary data are simply prescribed.

The mathematical implementation of the sponge layer (nudging) 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 :

$$\left(\frac{\partial \psi}{\partial t}\right)_{nudge} = -\alpha_{nudge} (\psi - \psi_{bc}) ,$$

where ψ_{bc} is the externally specified value of the prognostic variable ψ , and α_{nudge} is a relaxation coefficient. α_{nudge} gradually decreases with increasing distance from the boundary. It is computed separately for cell- and edge-based variables due to their differing distance from the boundary and is of the form

$$\alpha_{nudge} = \begin{cases} A_0 \exp\left(-\frac{r-r_0}{\mu}\right), & \text{if } r - r_0 \leq L \\ 0, & \text{elsewhere} \end{cases},$$

with A_0 the maximum relaxation coefficient, L the width of the relaxation zone given in units of cell rows, μ the e-folding width given in 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 `grf_bdywidth_c+1`, 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 `interp_nml`. The nudge zone width should at least comprise 8 (better 10) cell rows in order to minimize boundary artifacts.



Important note: Currently, nudging towards the driving model is only possible along the lateral boundaries, but not along the model top. Vertical velocity and corresponding vertical mass fluxes are simply 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).

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.

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 = 4` initialize limited area run from **COSMO-DE data**.

`init_mode = 7` initialize limited area run 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.

Both 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 vertical interpolation requires that the field HHL (vertical half level heights) is contained in the initial data.

Specifics of `init_mode=4`

- 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). Note that the filename must be specified twice. The same filename must be provided for `dwdfg_filename(initicon_nml)` and `ifs2icon_filename(initicon_nml)`. Most likely this was programmed on a late Friday afternoon
- 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` (**namelist** `initicon_nml`, **logical value**)

By default, this namelist parameter is set to `.TRUE.`. If `.FALSE.`, a separate analysis file is not required. The filename of the first guess file is specified via the `dwdfg_filename` namelist option, see Section 5.1.2.

Note that in the recent ICON version `lread_ana=.FALSE.` is set automatically for `init_mode= 4`, if it has been forgotten by the user.

- The required input fields are depicted in Figure 6.2.

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.

Atmosphere										
U,	V,	W,	T,	P,	QV,	QC,	QI,	QR,	QS,	HHL
Soil/Surface										
SMI,	T_SO,	T_G,	T_ICE,	H_ICE,	QV_S,	W_I,	T_SNOW,	W_SNOW,	H_SNOW,	RHO_SNOW, FRESHSNW

Figure 6.2.: Required variable set when initializing ICON-LAM from COSMO-DE data (i.e. `init_mode=4`). Optional fields are marked in gray. The field HHL (vertical half level heights) is required for vertical interpolation. DWD GRIB2 shortNames are used for notation.

`lread_ana` (namelist `initicon_nml`, logical value)

By default, this namelist parameter is set to `.TRUE.`. If `.FALSE.`, a separate analysis file is not required. The filename of the first guess file is specified via the `dwdfg_filename` namelist option, see Section 5.1.2.

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 2.3. A valid option is to use DWD's initialized analysis product for initialization. See Section 2.2.1 for ways to obtain it.

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 6.3. 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 2.9 for its specific content.



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.

Read-in of boundary data is controlled by the following namelist parameters:

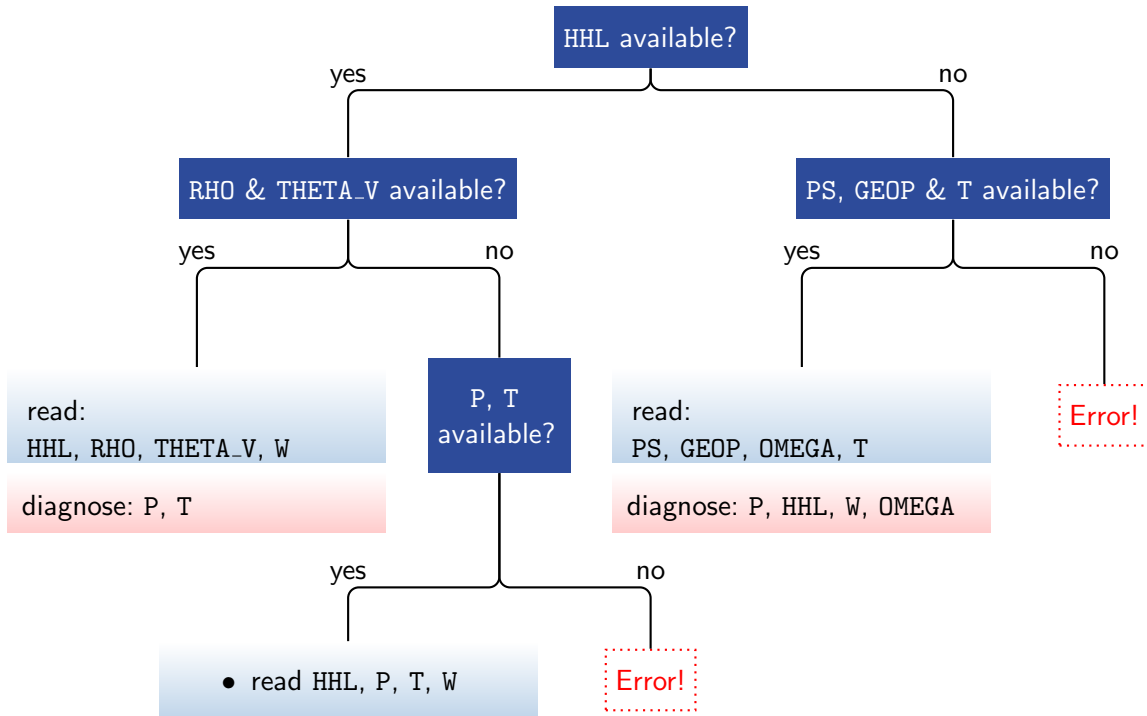


Figure 6.3.: Read-in of lateral boundary data. Based on this decision tree, ICON investigates the data file contents and diagnoses additional fields. Besides, the following fields are read from file: velocity fields U , V (or VN) and mixing ratios QV , QC , QI (optional: QR , QS).

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:

`dtime_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 preprocessing 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, keywords are used for `<nroot>` and `<jlev>`, which denote the grid's root subdivision and bisection level (see Section 2.1). The keywords `<y>`, `<m>`, `<d>`, and `<h>` are replaced by year, month, day, and hour. This naming scheme can be flexibly altered via the namelist parameter `latbc_filename` (namelist `limarea_nml`), see ICON's namelist documentation for details.

The absolute path to the boundary data can be specified with `latbc_path` (string parameter, `limarea_nml`).

Field names: `latbc_varnames_map_file` (namelist `limarea_nml`, string)

ICON supports NetCDF and GRIB2 as input format for boundary fields. Fields 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.

7. Parallelization and Output

In this chapter the possibilities offered by the namelist controlled model output are described.

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.

The chapter is concluded by 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.

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. 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 7.1.2.

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.

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 C.

Users can also specify the variable names in a different 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, floating point values, comma-sep.)

Comma separated list of model levels for which the variables and groups specified in the above mentioned variable list should be written to output. Level ordering does not matter.

dom (namelist output_nml, integer values, comma-sep.)

Related to setups with nests, i.e. multiple domains: 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 Section 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.

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), cf. 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, cf. 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 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 `num_io_procs` (`parallel.nml`) namelist parameter, see Section 7.3.1. 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_ml` (namelist output_nml, integer)

It may even be 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_ml` to set the number of output processes among which the output files should be divided.

The distribution of output load using `stream_partitions_ml` is illustrated in Fig. 7.1.

`pe_placement_ml` (namelist output_nml, integer array)

This array is related to the namelist parameters `num_io_procs` and `stream_partitions_ml` and allows for an even more fine-tuned distribution of the output workload. At most `stream_partitions_ml` 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.

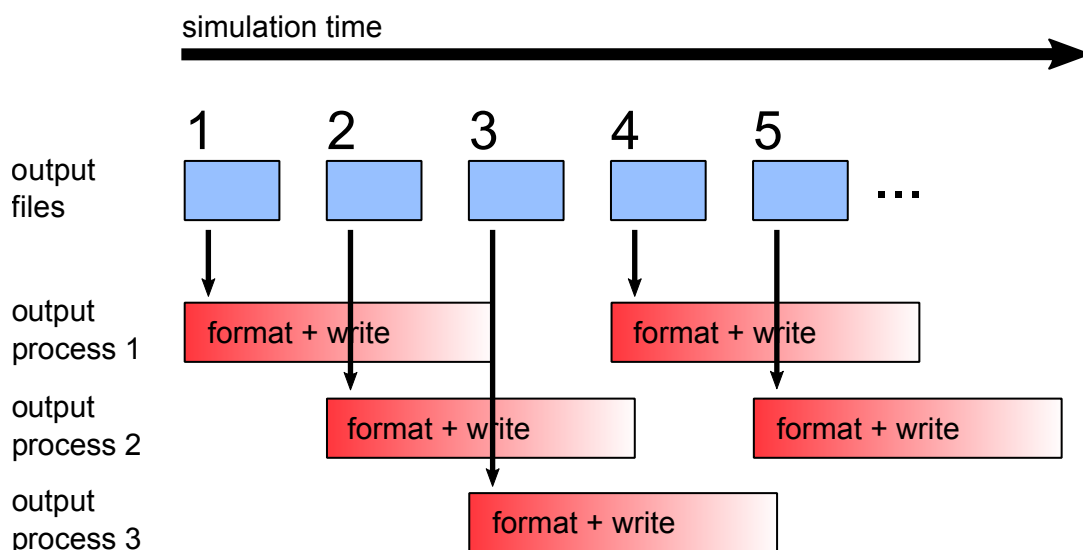


Figure 7.1.: Schematic illustrating the distribution of output load onto three output processes, using `num_io_procs=3` and `stream_partitions_ml=3`.

7.1.2. 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 (cf. Section 0.3). 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.

All of these parameters are set in the namelist `output_nml`. As it was already mentioned in Section 7.1, multiple instances of this namelist may be specified for a single model run, where each `output_nml` creates a separate output file.

The relevant namelist parameters for the interpolation of the output fields are:

hl_varlist / pl_varlist / il_varlist (character string lists)

Similar to the namelist parameter `ml_varlist`, 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 m, pressure levels in Pa and isentropes in K. Level ordering does not matter.

remap (namelist output_nml, integer value 0/1)

In combination with `reg_lat_def / reg_lon_def`:

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.

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:

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.

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 (cf. Section 7.3) 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 settings for `restart_write_mode` (character strings!) 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 (`num_restart_proc > 0`) write restart files while the simulation continues. 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. Parallelization and Performance Aspects

As mentioned in Section 1.2.1, ICON can use two different mechanisms for parallel execution:

- a) *OpenMP* – Multiple threads are run in a single process and share the memory of a single machine.
An implementation of OpenMP ships with your Fortran compiler. OpenMP-parallel execution therefore does not require the installation of additional libraries.
- b) *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.

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 OpenMP.

7.3.1. Settings for Parallel Execution

Several settings must be adjusted to control the parallel execution:

Namelist `parallel_nml`

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)

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: 118
#      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.

Batch queuing system

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*.

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.

When using the `qsub` command to submit a script file, the queuing system PBSPro allows for specification of options at the beginning of the file prefaced by the `#PBS` delimiter followed by PBS commands (see also the comments in Appendix A). For example, to run the executable in hybrid mode on 12 nodes with 4 OpenMP threads/processes, set

¹The notation “I/O” is justified by historical arguments. In the current version of ICON, these MPI processes exclusively operate as *output* servers.

```
#PBS -q xc_norm_h
#PBS -l select=12:ompthreads=4
#PBS -l place=scatter
#PBS -l walltime=01:00:00
#PBS -j oe
```

In more detail, the PBS keywords have the following meaning:

#PBS -q xc_norm_h	Job for the (Haswell) compute nodes of the XC 40.
#PBS -l select=\$NODES	To specify the number of compute nodes.
#PBS -l place=pack	If only one compute node is used.
#PBS -l place=scatter	If more than one compute node are used.
#PBS -l walltime=...	This directive specifies the maximum wall-clock time (real time) that a job should take.
#PBS -j oe	Put standard error and log to the same device.
#PBS -N	To give a special name to the job.

Application launch with aprun

Finally the user has to set the correct options for the application launcher, which is the `aprun` command on the Cray XC 40 platform. Here we have the following syntax:

```
aprun -n total number of MPI tasks
      -N number of MPI tasks/node
      -d number of threads/MPI task
      -j hyperthreading (enabled=2)
      -m memory/task
      command
```

Best Practice for Parallel Setups

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)

¹This implementation method is known as *loop tiling*, see also Section 8.4.

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 Intel-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.

7.3.2. Mixed Single/ Double Precision in ICON

To speed up code parts strongly limited by memory bandwidth, an 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, the preprocessor flags `-D__MIXED_PRECISION` and `-D__MIXED_PRECISION_2` need to be specified in the configuration settings used for generating the Makefile. The latter flag is used for physics tendencies.

Note that interpolation to a latitude-longitude grid is not supported for single-precision variables; if you desire to output physics tendency fields on a regular grid for diagnostic purposes, do not set `-D__MIXED_PRECISION_2`.

7.3.3. Bit-Reproducibility

Bit-reproducibility refers to the feature that running the same binary multiple times should ideally result in bitwise 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 Cray compiler, it is however possible to generate an ICON binary offering bit-reproducibility with only little performance loss. The ICON binary used in this workshop gives bit-reproducible results (will be checked in Exercise [D.5.7](#)).

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 a MPI parallelization bug.
- for checking the correctness of new code that is supposed not to change the results.

7.3.4. 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`).

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	118	...	272.564	272.637
L integrate_nh	247800	...	255.208	270.596
L nh_solve	247800	...	111.052	127.559
...				
L nh_hdiff	49678	...	4.618	6.894
L transport	49560	...	33.409	35.791
L adv_horiz	49560	...	22.849	24.663
L back_traj	148680	...	2.222	2.601
L adv_vert	49560	...	6.280	7.698
L prep_tracer	49560	...	0.113	1.588
L nesting	49560	...	0.000	0.001
L nesting.bdy_interp	49560	...	0.000	0.000
exch_data	2570040	...	18.931	73.496
L exch_data.wait	2570040	...	14.500	70.128
global_sum	50740	...	0.012	0.026
ordglb_sum	61596	...	0.441	5.243
wrt_output	1770	...	0.211	0.266
physics	49678	...	103.107	104.759
L nwp_radiation	10030	...	40.402	42.985
L radiation	220674	...	31.845	34.963
...				

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.



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.

8. 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 code. We exemplify this in Section 8.4 by implementing an own simple diagnostic.

8.1. 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 is split into several chunks of a much smaller length `nproma` (`parallel_nml`). We store the long vector in a 2D array, the first index counting the elements in a block (*line index*), the second index counting

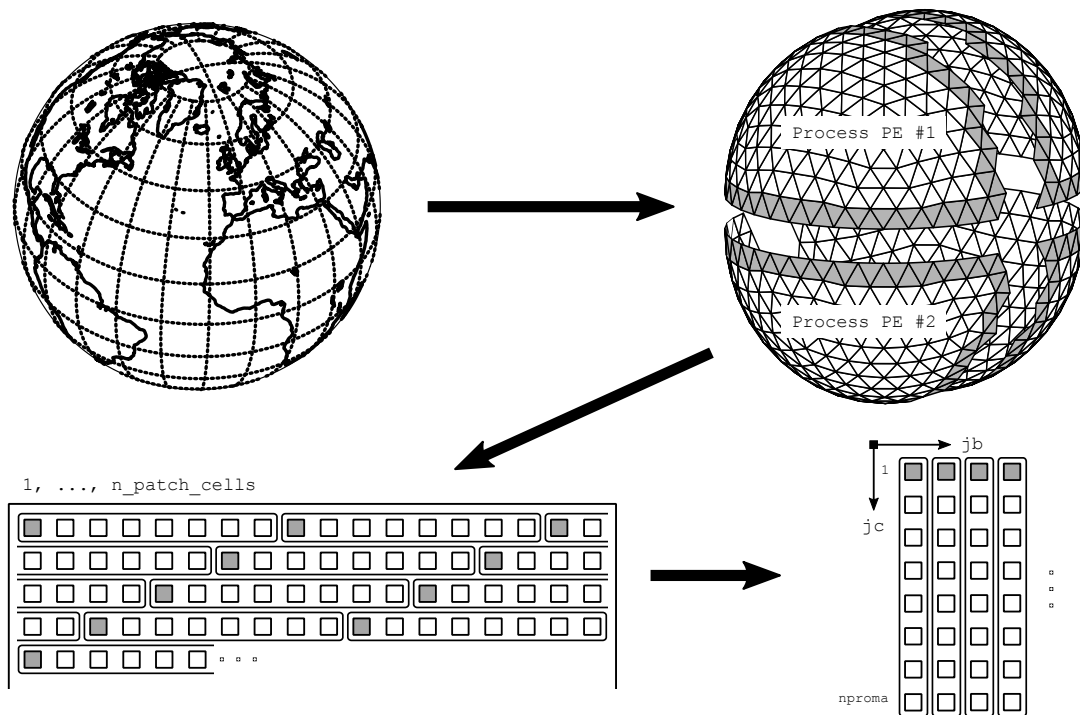


Figure 8.1.: Illustration of the 2D field representation. The original spherical domain is decomposed (light-gray: halo region). Afterwards, the long vector of grid cells is split into several chunks (blocking) of a much smaller length `nproma`.

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 8.1.

The 3D fields that were stored in 2D arrays, with the cell index as the first dimension and the second being the vertical coordinate, 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 reshape of the arrays.

The auxiliary functions `idx_no`, `blk_no` and `idx_1d` help to calculate the blocked indices from the 1D array index and vice versa (declared in the module `mo_parallel_config`).

Domain decomposition. Domain decomposition is, naturally, a prerequisite for scalability 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 processors. This means that we have only certain regions of a domain on each processor.

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

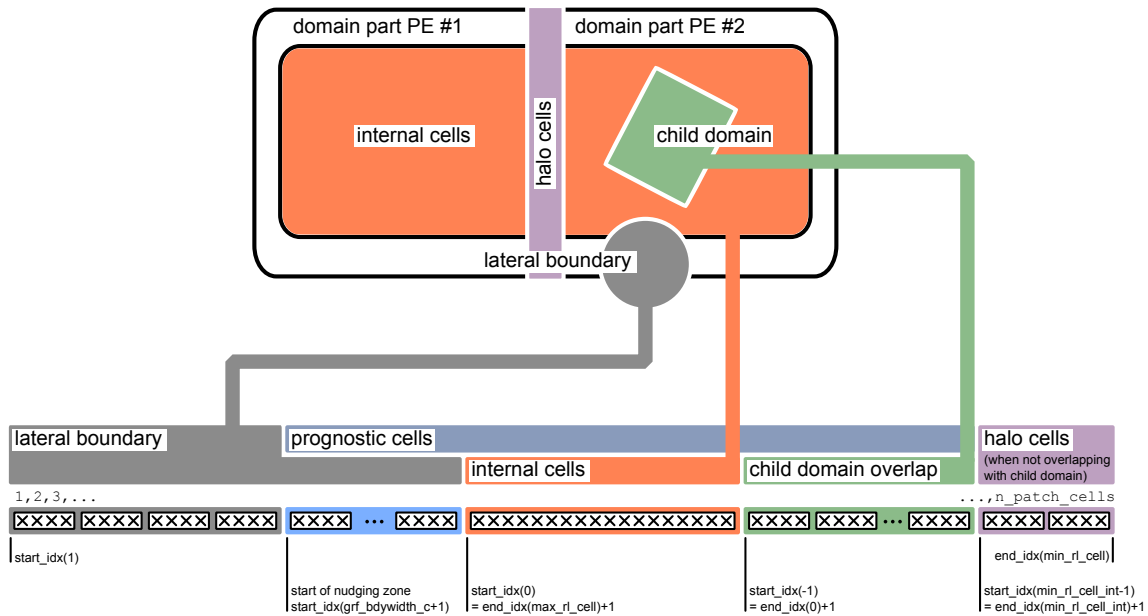


Figure 8.2.: Illustration of ICON’s index ordering using the example of grid cells. The upper part schematically shows the different parts of a computational domain, subdivided between two PEs: Each PE “owns” a subset of interior cells (orange area) and part of the lateral boundary (white). The (purple) halo region is shared between the PEs. Additionally, the sub-area of PE #2 covers a refined region (green child domain). The lower part of the illustration depicts the global index vector. It can be seen that the leftmost global indices (i. e. the smallest subscripts) correspond to the lateral boundary region, followed by the prognostic cells and, finally, the halo cells. Each portion is annotated by its start index, where the subscript corresponds to the `refin_cell_ctrl` value.

boundary of the partial domain which is overlaid with neighboring partial domains. 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 (which are also known as a *ghost-cell regions*) have to be updated by the neighboring partial domains. We will sketch this synchronization process in Section 8.2.4 below.



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 which extend beyond the halo region!

Index ordering. The index ordering is determined by the `refin_XXX_ctrl` index which counts the distance from the lateral boundary in units of cell/edge/vertex rows¹. To each

¹The `refin_c_ctrl` array already played a role for the preparation of lateral boundary input data, see Section 2.3.

value of `refin_c_ctrl`, say, corresponds a portion of the triangular cells: the cell rows at the lateral boundary, the nudging zone, the inner cells and those which overlap with a child domain, and the halo region. 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. The auxiliary function `get_indices_c` 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. Figure 8.2 visualizes the ordering of the grid portions in the global index vector.

8.2. Data Structures

8.2.1. Description of the Model Domain: `t_patch`

The `t_patch` data structure contains all information about the grid coordinates, 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 (`# domains`), 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 8.1 and non-existent indices 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
<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>	grid information, see below
<code>comm_pat_c/e/v</code>	halo communication patterns, see Section 8.2.4
	:

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
<code>neighbor_idx(:, :, :)</code>	line indices of triangles next to each cell
<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
<code>decomp_domain(:, :)</code>	domain decomposition flag, 0: owned (for cells)
	:

8.2.2. Date and Time Variables

When installing own processes 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 (<code>tc</code> means "time control")
<code>tc_exp_stopdate</code>	experiment stop
<code>tc_startdate</code>	start of single run (may differ from <code>tc_exp_startdate</code> in case of restart)
<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 library² which is precise up to milliseconds without round-off errors. The `mtime` library written in C resides in the directory `externals/mtime` 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
CALL datetimetostring(mtime_date, dstring)
WRITE (*,*) "2014-06-01T00:00:00 + 1 day = ", TRIM(dstring)
```

²The 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 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, `RefDate` denotes the event *reference date* (anchor date). 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)
```

8.2.3. 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 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`. 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)
```

Second, the data type `t_nh_diag` 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]
	:

8.2.4. Parallel Communication

To simplify the data exchange between neighboring domain portions, ICON contains synchronization routines `exchange_data`, defined in the module

```
src/parallel_infrastructure/mo_communication.f90.
```

These take the specific halo region as an argument (data type `t_comm_pattern`) and several exchange patterns are pre-defined for each domain (see 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.

As it has been noted in Section 7.3.1 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 everlasting chit-chat of the status log (screen) output: The `process_mpi_stdio_id` is always the 0 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 single line to screen would be

```
IF (my_process_is_stdio()) write (0,*) "Hello world!"
```

8.3. 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 8.3 we restrict ourselves to the most important subprograms.

These are listed together with a short description of their location and purpose, which, of course, changes gradually between released versions. We recommend to compare this to the flow chart of processes in the physics-dynamics coupling, see Fig. 3.14.

8.4. 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 with a view to the exercise Ex. D.8.1.

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 them step by step.

First, we need an appropriate **variable list** to which we can append our new variable. For the sake of simplicity, 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 `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 did not allocate the variable so far.

Each ICON variable must be accompanied by appropriate **meta-data**. In this example we need to initialize GRIB and NetCDF variable descriptors for a variable located in the cell circumcenters (mass points). As above we have omitted the necessary `USE` statements to keep this presentation as short as possible:

```
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 **dimensions** of a 2D field will be explained below. Here we take them as given:

```
shape2d_c = (/ nproma, nblks_c /)
```

Furthermore, 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 is reset every 6 hours.

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_HYBRID, &
             cf_desc, grib2_desc,             &
             action_list=actions(new_action(ACTION_RESET,interval)), &
             ldims=shape2d_c, lrestart=.FALSE. )
```

From now on the new field can be specified in the output namelists that were described in Section 7.1:

```
&output_nml
...
ml_varlist = 'newfield'
/
```

Looping over the grid points. Of course, the newly created field `'newfield'` still needs to be filled with values and the dimensions of the 2D field have not yet been declared. As explained 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 `nproma`.

Three-dimensional fields would have an additional dimension for the **column levels**:

```
shape3d_c = (/ nproma, nlev, nblks_c /)
```

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 points twice.

The auxiliary function `get_indices_c` 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(p_patch(domain), jb, i_startblk, i_endblk, is, ie, &
                   grf_bdywidth_c+1, min_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. A graphical interpretation of these constants is provided by Figure 8.2. The loop example therefore iterates over all prognostic cells (denoted by the blue area).



Loop exchange: Preprocessor flag `__LOOP_EXCHANGE`: This special preprocessor flag causes a loop interchange in many performance critical places of the ICON model code. 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.

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 once more take a look at Figure 3.14: The outer loop “Dynamics → Physics → Output” is contained in the core module `mo_nh_stepping` inside the `TIME_LOOP` iteration. Then, 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!



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.

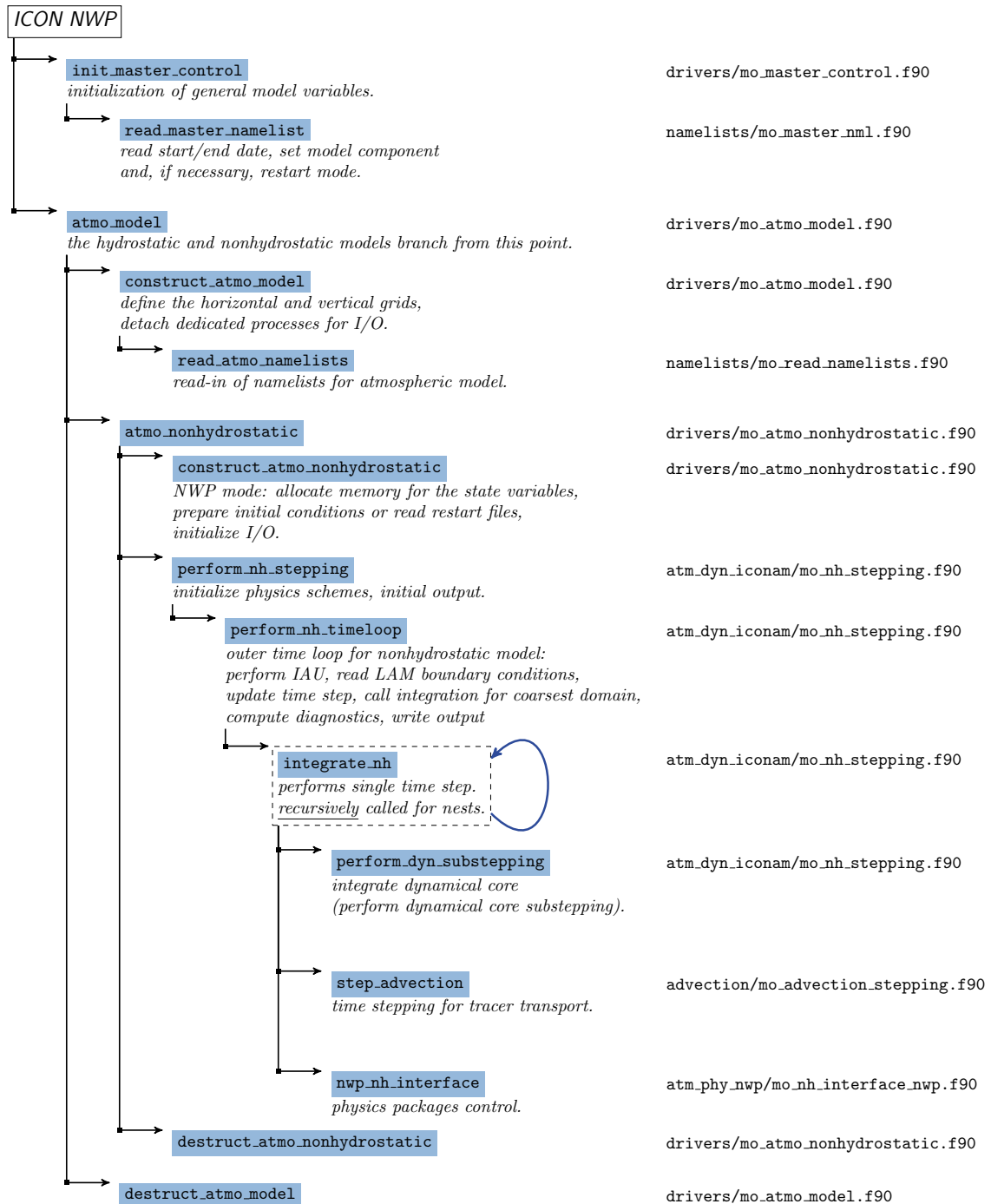


Figure 8.3.: Call tree of ICON's NWP component (note that this list has been restricted to the most important subroutines).

9. Post-Processing and Visualization

ICON offers the possibility to produce output either in NetCDF or GRIB2 format. Many visualization tools such as GrADS or Matlab 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 relatively new, many post-processing packages offer only limited support for GRIB data that has been stored on the triangular ICON grid.

For the visualization of *regular* grid data we will restrict ourselves in this course to a very simple program, `ncview`, which does not have a large functionality but is a very easy-to-use program for a quick view of NetCDF output files and therefore very useful for a first impression.

Model data that has been stored on the *triangular* ICON grid can be visualized with the NCL scripting language or the Generic Mapping Tools (GMT). Section 9.3 contains some examples how to visualize NetCDF data sets without the need of an additional regridding.

9.1. Retrieving Data Set Information

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.

9.1.1. `ncdump`

`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 -c data-file.nc
```

Dimension names and sizes, variable names, dependencies and values of dimensions will be displayed. To get only header information (same as `-c` option but without the values of dimensions) use

```
ncdump -h data-file.nc
```

To display the values of a specified variable which is contained in the NetCDF file, type

```
ncdump -v variable data-file.nc
```

To send data to a text file use

```
ncdump -b c data-file.nc > data-file.cdl
```

to produce 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 for example in *.txt-files just using:

```
ncdump -v variable data-file.nc > data-file.txt
```

For further information on working with `ncdump` see

[http://www.unidata.ucar.edu/software/netcdf/...
docs/netcdf_utilities_guide.html#ncdump_guide](http://www.unidata.ucar.edu/software/netcdf/...docs/netcdf_utilities_guide.html#ncdump_guide)

9.1.2. CDO – Climate Data Operators

The CDO (Climate Data Operators) are a collection of command-line operators to manipulate and analyze NetCDF and GRIB data. The CDO package is developed and maintained at MPI for Meteorology in Hamburg. Source code and documentation are available from

<https://code.mpimet.mpg.de/projects/cdo>

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.

9.2. Plotting Data Sets on Regular Grids

9.2.1. 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. 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 colormaps, invert data, etc.

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.

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!

9.2.2. Remapping to Regular Grids with CDO

Basically, two steps are necessary to process ICON data:

- In the first call of the CDO ("gencon"), coefficients 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 grids. Currently in operational use is the global grid #26 with 13 km horizontal grid spacing (http://icon-downloads.mpimet.mpg.de/grids/public/edzw/icon_grid_0026_R03B07_G.nc).

The structure of the lon-lat grid description file is explained in Appendix D of the CDO documentation (<https://code.mpimet.mpg.de/projects/cdo/embedded/index.html#x1-836000D>). A global regular grid, for example, would be defined as

```
gridtype = lonlat
xsize   = 1440
ysize   = 721
xfirst  = 0.0
xinc    = 0.25
yfirst  = -90.0
yinc    = 0.25
```

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 (which is a COSMO software) or the DWD ICON Tools (see below), which are internally used at DWD but lack official support.

9.3. Plotting Data Sets on the Triangular Grid

9.3.1. NCL – NCAR Command Language

The NCAR Command Language (NCL) is an interpreted language designed specifically for scientific data analysis and visualization. It allows convenient access to data in a variety of formats such as NetCDF and GRIB1/2, among others. NCL has many features common to modern programming languages, such as types, variables, operators, expressions, conditional statements, loops, and functions and procedures.

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.

NCL Quick-Start Example

The following example script creates a temperature contour plot with NCL (see Figure 9.1):

```

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
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 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 that is available online under

<http://www.ncl.ucar.edu>

Section 9.3.2 contains a step-by-step tutorial for another NCL example. For the exercises in this tutorial we refer to the prepared NCL scripts. These files are stored in the sub-directory `test_cases/casexx` together with the model run scripts.

9.3.2. NCL Step-by-step Tutorial

In the following we provide a detailed step-by-step tutorial for producing graphics from an ICON data set. We will use NCL's batch mode, i.e. instead of typing each command in interactive mode, we will create a file `visualization_tutorial.ncl` where a sequence of commands can be stored and executed with

```
ncl visualization_tutorial.ncl
```

Please note that this tutorial script requires NCL version 6.2.0 or higher.

We begin by loading some NCL "libraries" which provide high-level plotting functions

```
; load libraries
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
```

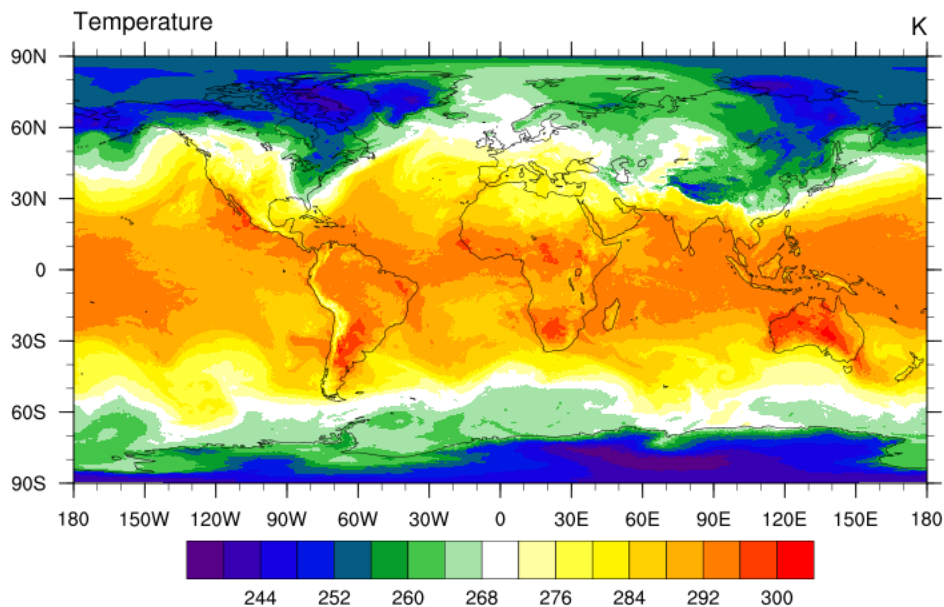


Figure 9.1.: ICON temperature field on a specific model level produced with the above NCL script.

These lines also contain a comment. Comment lines in NCL are preceded by the semicolon character ';':

Step 1: Reading Grid Coordinates From File

Then, as the ICON model uses an unstructured grid topology, we open and read such a topology file, stored in NetCDF format, by the following commands:

```
gridfile = addfile( "gridfile.nc", "r" )
print(gridfile)
```

The `print` command lists all variables that have been found in the NetCDF file as textual output. For the ICON grid, the vertex positions of the grid triangles are of special interest. They are stored as longitude/latitude positions in the `vlon`, `vlat` (this is explained in more detail in Section 2.1.1, page 19). For NCL we convert from steradians to degrees:

```
rad2deg = 45./atan(1.)
vlon      = gridfile->vlon * rad2deg
vlat      = gridfile->vlat * rad2deg
```

Additionally, we load the vertex indices for each triangle edge of the icosahedral mesh.

```
edge_vertices = gridfile->edge_vertices
```

The indices are stored in the grid file data set `edge_vertices` and reference the corresponding vertices from `vlon`, `vlat`,

$$\text{edge } \#i : \quad (\text{vlon}[q_1], \text{vlat}[q_1]) - (\text{vlon}[q_2], \text{vlat}[q_2])$$

where

$$q_{1/2} := \text{edge_vertices}[1/2, i] - 1$$

Note that by subtracting 1 we take the 0-based array indexing of NCL into account.

Furthermore, it is convenient to store the size of the edges array, i.e. the number of grid edges, in a local variable `nedges`.

```
size_edge_vertices = dimsizes(edge_vertices)
nedges             = size_edge_vertices(1)
```

Step 2: Creating a Plot of the Triangular Grid

Producing graphics with NCL requires the creation of a so-called *workstation*, i.e. a description of the output device. In this example, this “device” will be a PostScript file `plot.ps`, but we could also define a different output format, e.g. “png” instead.

```
wks = gsn_open_wks("ps", "plot")
```

Then the map settings have to be defined and we collect these specifications in a data structure named `config1`. First of all, we disable the immediate drawing of the map image, since the ICON icosahedral grid plot will consist of two parts: the underlying map and the grid lines. We do so by setting `gsnFrame` and `gsnDraw` to `False`.

We then define an orthographic projection centered over Europe. It is important that grid lines are true geodesic lines, otherwise the illustration of the ICON grid would contain graphical artifacts, therefore we set the parameter `mpGreatCircleLinesOn`.

```
config1                = True
config1@gsnMaximize    = True
config1@gsnFrame       = False
config1@gsnDraw        = False

config1@mpProjection   = "Orthographic"
config1@mpGreatCircleLinesOn = True
config1@mpCenterLonF   = 10
config1@mpCenterLatF   = 50
config1@pmTickMarkDisplayMode = "Always"
```

(Note that “pm” in `pmTickMarkDisplayMode` is the correct spelling!)

Having completed the setup of the `config1` data structure, we can create an empty map by the following command:

```
map = gsn_csm_map(wks,config1)
```

Now, the edges of the ICON grid must be added to the plot. As described before, we convert the indirectly addressed `edge_vertices` into an explicit list of geometric segments with dimensions `[nedges × 2]`:

```
ecx = new((/nedges,2/),double)
ecy = new((/nedges,2/),double)

ecx(:,0) = vlon(edge_vertices(0,:)-1)
ecx(:,1) = vlon(edge_vertices(1,:)-1)
ecy(:,0) = vlat(edge_vertices(0,:)-1)
ecy(:,1) = vlat(edge_vertices(1,:)-1)
```

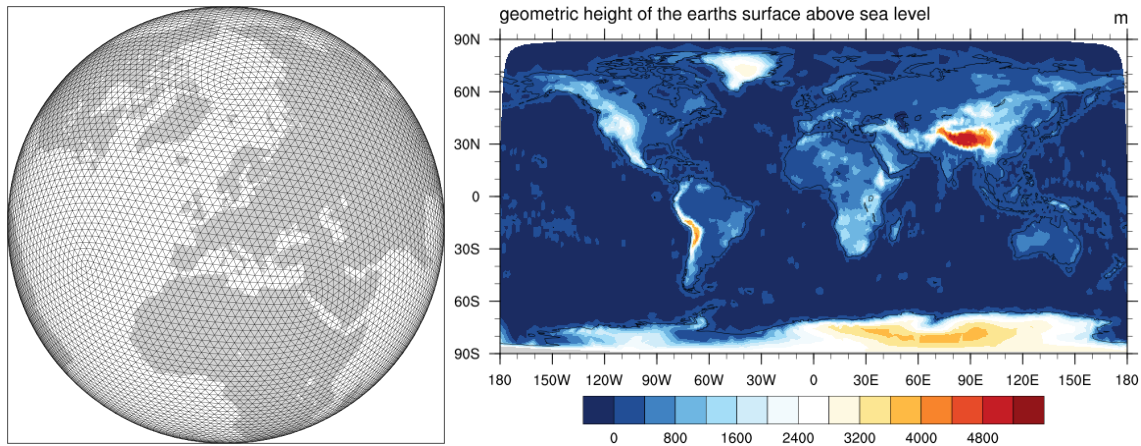


Figure 9.2.: The two plots generated by the NCL example script in Section 9.3.2.

There exists an NCL high-level command for plotting lines, `gsn_add_polyline`. Since this function expects one-dimensional lists for its interface, we use the auxiliary function `ndtooned` for reshaping the array of lines,

```
lines_cfg          = True
lines_cfg@gsSegments = ispan(0,nedges * 2,2)
poly = gsn_add_polyline(wks,map,ndtooned(ecx),ndtooned(ecy),lines_cfg)
```

The whole plotting process is now triggered by the command

```
draw(map)
frame(wks)
```

The first page of the resulting PostScript file `plot.ps` will contain an illustration similar to Fig. 9.2 (left part).

Step 3: Loading a Data Set from a Second File

In order to visualize unstructured data sets that have been produced by the ICON model they have to be stored in NetCDF format. As a second file we open such a NetCDF data set `datafile.nc` in read-only mode and investigate its data set `topography_c`:

```
datafile = addfile("datafile.nc", "r" )
topo = datafile->topography_c
printVarSummary(topo)
```

The final step of this exercise is the creation of a contour plot from the data contained in `datafile`. As it has been stated by the previous call to `printVarSummary`, the data sites for the field `topography_c` are the triangle circumcenters, located at `clon`, `clat`.

```
clon = gridfile->clon * rad2deg
clat = gridfile->clat * rad2deg
```

For a basic contour plot, a cylindrical equidistant projection with automatic adjustment of contour levels will do. It is important to specify the two additional arguments `sfXArray` and `sfYArray`.

```
config2                = True
config2@mpProjection   = "CylindricalEquidistant"
config2@cnFillOn       = True
config2@cnLinesOn      = False
config2@sfXArray       = clon
config2@sfYArray       = clat
```

Afterwards, we generate the plot (page 2 in our PostScript file) with a call to `gsn_csm_contour_map`.

```
map = gsn_csm_contour_map(wks,topo,config2)
```

Note that this time it is not necessary to launch additional calls to `draw` and `frame`, since the default options in `config2` are set to immediate drawing mode.

You may wonder why the plot has a rather smooth appearance without any indication of the icosahedral triangular mesh. What happened is that NCL generated its own Delaunay triangulation building upon the cell center coordinates provided via `clon`, `clat`. Thus, we are unable to locate and investigate individual ICON grid cells. In order to visualize individual cells, we need to additionally load the vertex coordinates of each triangle into NCL. This information is also available from the grid file and is stored in the fields `clon_vertices`, `clat_vertices`.

```
clon_vertices          = gridfile->clon_vertices * rad2deg
clat_vertices          = gridfile->clat_vertices * rad2deg
config2@sfXCellBounds  = clon_vertices
config2@sfYCellBounds  = clon_vertices
config2@cnFillMode     = "CellFill"
```

By choosing the `CellFill` mode, it is ensured that every grid cell is filled with a single color.

Afterwards we generate the plot once more with a call to `gsn_csm_contour_map`.

```
map = gsn_csm_contour_map(wks,topo,config2)
```

Do you see the difference?

9.3.3. Visualization with R

Section author

J. Förstner, 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 Foundations 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.

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.

R Quick-Start Example

The following example script creates a temperature contour plot with R (see Figure 9.3):

```
# ... on LCE, as a prerequisite issue the following module commands
# on the command line:
# module unload gribr_api ; module load eccodes/2.6.0
# module load oracle/12.1.0.2
# module load R/3.4.1 ; module load eccodes/R-gribr-2.6.0
```

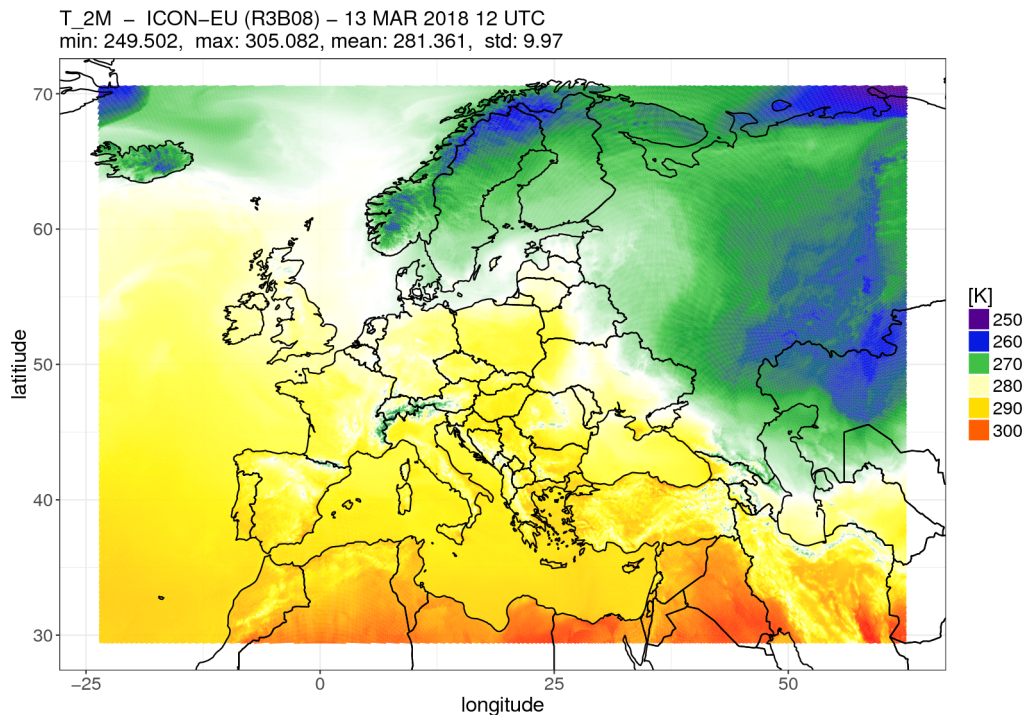


Figure 9.3.: ICON 2 m temperature field produced with the given example script for R.

```
# load necessary libraries
library(grib)
library(RNetCDF)
library(data.table)
library(ggplot2)

# function to convert coordinates in degrees
rad2deg <- function(rad) {(rad * 180) / (pi)}

# landscape mode
pdf(paper="a4r", width=11.692, height=8.267)

# open the icon grid file
ncHandle <- open.nc("./icon_grid_0027_R03B08_N02.nc")

# get the longitudes and latitudes of the triangle vertices of a cell
# (in degrees)
vlon <- rad2deg(var.get.nc(ncHandle,"clon_vertices"))
vlat <- rad2deg(var.get.nc(ncHandle,"clat_vertices"))

# close the icon grid file
close.nc(ncHandle)

# open the grib file
gribHandle <- grib_open("./T_2M.R3B08_ICON-EU.grb")
```

```

# select the grib record based on a given list of keys
gribRecord <- grib_select(gribHandle, list(shortName = "T_2M"))

# close the grib file
grib_close(gribHandle)

# colors and levels
colors <- c("#570088", "#3700b3", "#1500e2", "#0116e2",
            "#045c83", "#079d2c", "#44c549", "#a5e3a8", "#ffffff",
            "#ffffa2", "#ffff3e", "#fff400", "#ffd100",
            "#ffb000", "#ff7d00", "#ff3c00", "#ff0000")

# specify the domain, i.e. the longitudinal and latitudinal range to plot
xrange <- c(-23.55, 62.55)
yrange <- c(29.45, 70.55)

# create a data table with the 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))

# 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]

# finally plot the data using ggplot2 with geom_polygon

# first create the plot object
p <- ggplot() +
  geom_polygon(data=DT, aes(x=lon, y=lat, group=id, fill=var)) +
  scale_fill_gradientn(colors=colors, guide="legend") +
  borders(colour = "black", xlim=xrange, ylim=yrange) +
  coord_cartesian(xlim=xrange,ylim=yrange) +
  theme_bw() +
  labs(x="longitude", y="latitiude", 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("T_2M - ICON-EU (R3B08) - 13 MAR 2018 12 UTC\n",
                "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
p

```

9.3.4. GMT – Generic Mapping Tools

GMT is an open source collection of command-line tools for manipulating geographic and Cartesian data sets and producing PostScript illustrations ranging from simple x-y plots via contour maps to 3D perspective views. GMT supports various map projections and transformations and facilitates the inclusion of coastlines, rivers, and political boundaries. GMT is developed and maintained at the University of Hawaii, and it is supported by the National Science Foundation.

To install GMT on your local platform, see the GMT website:

<http://gmt.soest.hawaii.edu>

Since GMT is comparatively fast, it is especially suited for visualizing high resolution ICON data on the native (triangular) grid. It is capable of visualizing individual grid cells and may thus serve as a helpful debugging tool. So far, GMT is not capable of reading ICON NetCDF or GRIB2-output offhand. However, CDO can be used to convert your data to a format readable by GMT.

From your NetCDF output, you should first select your field of interest and pick a single level at a particular point in time:

```
cdo -f nc selname,VNAME -seltimestep,ITIME -sellevidx,ILEV \  
    ICON_OUTPUT.nc ICON_SELECTED.nc
```

Now this file must be processed further using the `outputbounds` command from CDO, which finally leads to an ASCII file readable by GMT.

```
cdo -outputbounds ICON_SELECTED.nc > ICON_SELECTED.gmt
```

The output looks as follows:

```
# Generated by CDO version 1.6.3  
#  
# Operator = outputbounds  
# Mode      = horizontal  
#  
# File   = NWP_DOM01_ML_0006_temp.nc  
# Date   = 2012-01-04  
# Time   = 00:00:00  
# Name   = temp  
# Code   = 0  
# Level  = 80  
#  
> -Z254.71  
  -155.179  90  
  36  89.5695  
  108  89.5695  
  -155.179  90  
> -Z255.276  
  36  89.5695  
  36  89.1351
```



```

72  89.2658
36  89.5695
...

```

For each triangle, it contains the corresponding data value (indicated by `-Z`) and vertex coordinates.

As a starting point, a very basic GMT script is added below. It visualizes the content of `test.gmt` on a cylindrical equidistant projection including coastlines and a colorbar. An example plot based on this script is given in Figure 9.4.

```

#!/bin/bash

# Input filename
INAME="test.gmt"

# Output filename
ONAME="test.ps"

# generate color palette table (min/max/int)
makecpt -Cpolar -T"235"/"305"/"5" > colors.cpt

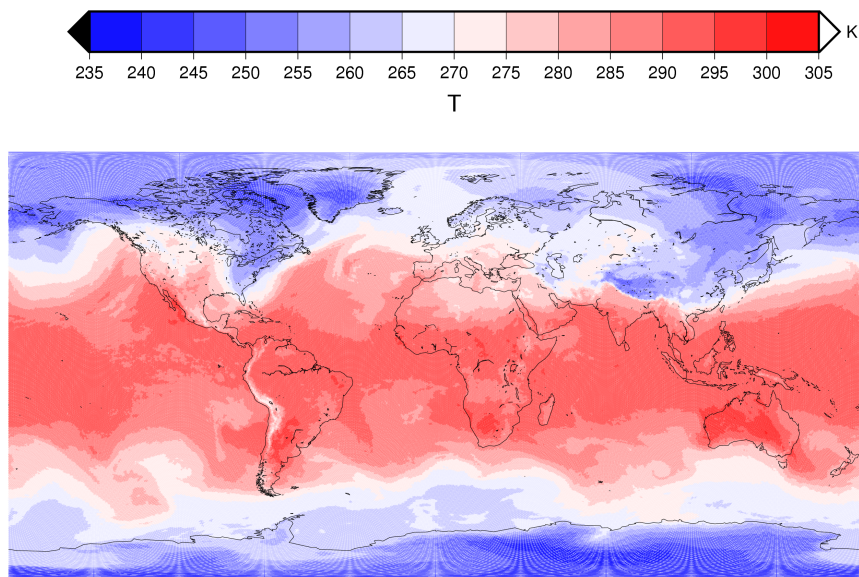
# draw triangle and take fill color from colors.cpt
psxy ${INAME} \
-Rd -Jq0/1:190000000 -Ccolors.cpt -X3.2 -Y4. -K > ${ONAME}

# visualize coastlines
pscoast -Rd -Jq0/1:190000000 -Dc -W0.25p,black -K -0 >> $ONAME

# plot colorbar
psscale -D11c/14c/18c/1.0ch -Ccolors.cpt -E -B:"T":/:K: -U -0>> $ONAME

```

Note: In order to get filled polygons, the `-L` option must be added to `psxy`. The purpose of `-L` is to force closed polygons, which is a prerequisite for polygon filling. However, in the latest release of GMT (5.1.2) adding this option results in very large output files whose rendering is extremely slow. Thus, the `-L` option was omitted here so that only triangle edges are drawn and colored.



GMT 2014 Jun 14 15:42:47

Figure 9.4.: ICON temperature field on a specific model level produced with the above GMT script.

10. Running ICON-ART

In this lesson you will learn how to run the package for **A**erosols and **R**eactive **T**race **G**ases, ICON-ART (Rieger et al., 2015, Schröter et al., 2018).

10.1. General Remarks

ICON-ART is an extension of the ICON model that was developed at the Institute of Meteorology and Climate Research (IMK) at the Karlsruhe Institute of Technology (KIT). It allows the online calculation of reactive trace substances and their interaction with the atmosphere. The interfaces to the ART code are part of the official ICON code.

In order to obtain the ART code, the institution that wants to use ICON-ART has to sign an additional license agreement with Karlsruhe Institute of Technology (KIT). Further information can be found on the following website:

<http://icon-art.imk-tro.kit.edu>

After you have signed the license agreement, you will be provided with a compressed file with the recent source code of ART which is called `ART-v<X>.<YY>.tar.gz`. `<X>` and `<YY>` indicate the version number.

10.2. ART Directory Structure

The ART directory contains several subdirectories. The purposes of those subdirectories are explained in the following.

The Directory `aerosol_dynamics`

ICON-ART solves the diffusion equation of aerosol. For this purpose, the following processes have to be considered: Advection¹, turbulent diffusion¹, changes due to subgrid-scale convective transport¹, sedimentation, washout, coagulation, condensation from the gas-phase, radioactive decay, and emissions². With a few exceptions (marked by ¹ and ²), the modules calculating the tendencies due to these processes are stored within the `aerosol_dynamics` folder. Additionally, routines calculating diagnostic properties that are needed as input for the aerosol process parameterizations are stored within this folder.

The exceptions are the following:

- ¹: The tendencies due to these processes are calculated within the ICON code. This is part of the tracer framework of ICON.
- ²: The emission routines are an important source for atmospheric aerosol. ART offers the option to easily plug in new emission schemes. In order to keep clarity within the folders, emissions routines get their own folder `emissions` (see below).

The Directory `chemistry`

ICON-ART solves the diffusion equation of gaseous tracers. Besides advection, turbulent diffusion and subgrid-scale convective transport which are treated by the ICON tracer-framework, this includes also chemical reactions. The `chemistry` directory contains the routines to calculate chemical reaction rates of gaseous species.

The Directory `emissions`

Within the `emissions` directory, emission routines for aerosol and gaseous species are stored.

The Directory `externals`

Within the `externals` directory, code from external libraries is stored (i.e. `cloudj`, `meci-con`).

The Directory `io`

The `io` directory contains input and output routines.

The Directory `chem_init`

The `chem_init` directory contains routines needed for an initialization of ICON-ART tracers with Mozart results.

The Directory `phy_interact`

Modules within the `phy_interact` directory treat the direct interaction of aerosol particles and trace gases with physical parameterizations of ICON. Examples are the interaction of aerosols with clouds (i.e. the two-moment microphysics) and radiation.

The Directory `runctrl_examples`

The `runctrl_examples` directory contains the following subdirectories

emiss_ctrl storage of example emission files for volcanic emissions, radioactive release

init_ctrl location of coordinate file for MOZART initialisation and initialisation table for LINOZ (linearized ozone) algorithm

photo_ctrl location of CloudJ Input files (Cross sections, Q-Yields)

run_scripts runscripts for testsuite and training course testcases

xml_ctrl storage of basic .xml files for tracer registration and system files (.dtd)

The Directory `shared`

The `shared` directory contains a collection of routines that do not fit into other categories. This applies mostly to initialization and infrastructure routines.

The Directory `tools`

The `tools` directory contains helpful tools for ART developers (e.g. a generalized clipping routine).

10.3. Installation

In this section, a brief description of how to compile ICON-ART is given. The user has to do the same steps as compiling ICON with a few additions. The reader is referred to Section 1.2.2 in order to compile ICON successfully. First, the `art.tar.gz` file has to be uncompressed. You will obtain a directory, which should be copied inside the ICON source directory `$ICON-DIR/src/`. In the following, we refer to this directory `$ICON-DIR/src/art` as `$ARTDIR`.

If you have compiled ICON without ART before, you have to do clean up first:

```
make distclean
```

In order to compile ICON-ART, an additional flag has to be set at the configuration command:

```
./configure --with-fortran=cray --with-art
```

By setting `--with-art` a compiler flag `-D__ICON_ART` is set. This flag tells the preprocessor to compile the code inside the ART interfaces and hence connect the ICON code with the ART code. As soon as the configuration is finished, you can start to compile the ICON-ART code:

```
./build_command
```

10.4. Configuration of an ART Job

10.4.1. Recommended ICON Namelist Settings

It is necessary for the user to choose the ICON settings carefully to obtain a stable ICON-ART simulation with scientifically reasonable results. Hence, the user should pay special attention to the namelist parameters listed in table 10.1.

Table 10.1.: Recommended ICON namelist settings for ART tracers.

Parameter	Value	Namelist	Description
<code>dttime</code>	-	<code>run_nml</code>	If facing stability problems, it is recommended to use a shorter time step as recommended by operational setups (e.g. $\frac{3}{5} \cdot dttime_{oper}$).
<code>ndyn_substeps</code>	-	<code>run_nml</code>	There is no need to call the dynamics more often than in operational setup. Adjust it according to your <code>dttime</code> choice.

10.4.2. ART Namelists

ICON-ART has an own namelist to modify the setup of ART simulations at runtime. The main switch for ART, `lart`, is located inside `run_nml`. The namelist for the other ART switches is called `art_nml`.

A naming convention is used in order to represent the type of data. An `INTEGER` namelist parameter starts with `iard_`, a `REAL` namelist parameter start with `rart_`, a `LOGICAL` namelist parameter starts with `lart_`, and a `CHARACTER` namelist parameter starts with `cart_`.

The ICON-ART namelist is located in the module `src/namelists/mo_art_nml.f90`. General namelist parameters are listed and explained within Table 10.2. Namelist parameters for ART input are listed within Table 10.3. Namelist parameters related to atmospheric chemistry are listed within Table 10.4. Namelist parameters related to aerosol physics are listed within Table 10.5.

Table 10.2.: General namelist parameters to control the ART routines. These switches are located inside `art_nml`. The only exception is the `lart` switch which is located in the `run_nml`.

Namelist Parameter	Default	Description
<code>lart</code>	<code>.FALSE.</code>	Main switch which enables the ART modules. Located in the namelist <code>run_nml</code> .
<code>lart_chem</code>	<code>.FALSE.</code>	Enables chemistry. The chemical mechanism and the according species are set via <code>lart_chem_mechanism</code> .
<code>lart_pntSrc</code>	<code>.FALSE.</code>	Enables point sources for passive tracer. The sources are controlled via <code>cart_pntSrc.xml</code> . See also Section 10.4.5.
<code>lart_aerosol</code>	<code>.FALSE.</code>	Main switch for the treatment of atmospheric aerosol.
<code>lart_passive</code>	<code>.FALSE.</code>	Main switch for the treatment of passive tracer.
<code>lart_diag_out</code>	<code>.FALSE.</code>	If this switch is set to <code>.TRUE.</code> , diagnostic output fields are available. Set it to <code>.FALSE.</code> when facing memory problems.

Table 10.3.: Namelist parameters to control ART input. These switches are located inside `art_nml`. For details regarding the tracer and modes initialization with XML files, see Section 10.4.3.

<code>cart_chemistry.xml</code>	''	Path and file name to the XML file for specifying chemical tracer. See also Section 10.4.3.
<code>cart_aerosol.xml</code>	''	Path and file name to the XML file for specifying aerosol tracer. See also Section 10.4.3.
<code>cart_passive.xml</code>	''	Path and file name to the XML file for specifying passive tracer. See also Section 10.4.3.
<code>cart_modes.xml</code>	''	Path and file name to the XML file for specifying aerosol modes. See also Section 10.4.4.
<code>cart_pntSrc.xml</code>	''	Path and file name to the XML file for specifying point source emissions. See also Section 10.4.5.

Table 10.4.: Namelist parameters related to atmospheric chemistry. These switches are located inside `art_nml`.

Namelist Parameter	Default	Description
<code>iaart_chem_mechanism</code>	0	Sets the chemical mechanism and takes care of the allocation of the according species. Possible values: 0: Stratosph. short-lived Bromocarbons. 1: Simplified OH chemistry, takes photolysis rates into account 2: Full gas phase chemistry

Table 10.5.: Namelist parameters related to aerosol physics. These switches are located in `art_nml`.

Namelist Parameter	Default	Description
<code>iaart_seasalt</code>	0	Treatment of sea salt aerosol. Possible values: 0: No treatment. 1: As specified in Lundgren et al. (2013) .
<code>iaart_volcano</code>	0	Treatment of volcanic ash particles. Possible values: 0: No treatment. 1: 1-moment treatment. As described in Rieger et al. (2015) . 2: 2-moment treatment.
<code>cart_volcano_file</code>	''	Path and filename of the input file for the geographical positions and the types of volcanoes.

10.4.3. Tracer Definition with XML Files

The definition of tracers in ICON-ART is done with the use of three XML files. A distinction between aerosol, chemical and passive tracers is made. Aerosol tracers are defined by an XML file specified with the namelist switch `cart_aerosol_xml` containing all liquid and solid particles participating in aerosol dynamics. Chemical tracers are defined by an XML file via `cart_chemistry_xml` and cover gaseous substances participating in chemical reactions. Passive comprises all gaseous, liquid and solid tracers that are only participating in transport processes. These are specified with the XML file `cart_passive_xml`. With the following example XML file, two passive tracers called `trPASS1` and `trPASS2` are defined:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tracers SYSTEM "tracers.dtd">

<tracers>
  <passive id="trPASS1">
    <transport type="char">stdaero</transport>
    <unit type="char">kg kg-1</unit>
  </passive>
  <passive id="trPASS2">
    <transport type="char">stdaero</transport>
    <unit type="char">kg kg-1</unit>
  </passive>
</tracers>
```

Additionally, each tracer gets two different meta data. Firstly, transport defines a template of horizontal and vertical advection schemes and flux limiters, in this example the template `stdaero` is used. A description of available transport templates is given below. Secondly, a unit is specified, which will also be added as meta data to any output of the tracer. Note, that the type of meta data has to be specified. In this example, both meta data are of type character ("char"). For other meta data you could also choose integer ("int") or real ("real").

Passive Tracers

For passive tracers, the only required meta data is unit (char). You can find an example for the passive tracer XML file at the `runctrl_examples/xml_ctrl` folder named `tracers_passive.xml`.

Chemical Tracers

For chemical tracers, the only required meta data is unit (char). Usually, `mol_weight` (molecular weight, real) is also needed, although it is technically not required. You can find an example for the chemical tracer XML file at the `runctrl_examples/xml_ctrl` folder named `tracers_chemtracer.xml`.

Aerosol Tracers

For aerosol tracers, there is a list of necessary meta data specifications: the meta data unit (char), moment (int), mode (char), sol (solubility, real), rho (density, real) and mol_weight (molecular weight, real) are required. You can find an example for the aerosol tracer XML file at the `runctrl_examples/xml_ctrl` folder named `tracers_aerosol.xml`.

Available Transport Templates

Currently, there are three different transport templates available: **off**, **stdaero** and **stdchem**. These templates avoid the necessity to add a tracer advection scheme and flux limiter for each single tracer in the namelist. Hence, the values of the namelist parameters `ihadv_tracer`, `ivadv_tracer`, `itype_hlimit` and `itype_vlimit` are overwritten by the template. Specific information concerning the advection schemes mentioned in the following can be found in Section 3.5.

Specifying **off**, all advective transport for this tracer is turned off (i.e. `ihadv_tracer` and `ivadv_tracer` are set to 0).

The transport template **stdaero** uses a combination of Miura and Miura with subcycling for the horizontal advection (i.e. `ihadv_tracer` = 22), 3rd order piecewise parabolic method handling CFL >1 for vertical advection (i.e. `ivadv_tracer` = 3) in combination with monotonous flux limiters (i.e. `itype_hlimit` = 4 and `itype_vlimit` = 4). This means that the conservation of linear correlations is guaranteed which is important for modal aerosol with prognostic mass and number and diagnostic diameter. By this, the diameter of aerosol does not change due to transport.

The transport template **stdchem** uses the same advection schemes as `stdaero` (i.e. `ihadv_tracer` = 22 and `ivadv_tracer` = 3). However, the considerably faster positive definite flux limiters are used (i.e. `itype_hlimit` = 3 and `itype_vlimit` = 2). By this, the mass is still conserved. However, the conservation of linear correlations is traded for a faster computation of the advection.

As you might have noticed in the previous section, the choice of a transport template is not required at the tracer definition. If no transport template is chosen, **stdaero is used as default template**.

10.4.4. Modes Definition with XML Files

Similar to the previously described tracer definition with XML files, aerosol modes are also defined with XML files. The according namelist parameter specifying the XML file is called `cart_modes_xml`. An example file `modes.xml` is provided in the `runctrl_examples/xml_ctrl` folder. The definition of a mode is done as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE modes SYSTEM "modes.dtd">
```

```

<modes>
  <aerosol id="seasa">
    <kind type="char">2mom</kind>
    <d_gn type="real">0.200E-6</d_gn>
    <d_gm type="real">0.690E-6</d_gm>
    <sigma_g type="real">1.900E+0</sigma_g>
    <rho type="real">2.200E+3</rho>
  </aerosol>
</modes>

```

In this example, a mode called `seasa` is defined with 2 prognostic moments (`2mom`). The initial number and mass diameters (`d_gn` and `d_gm`) as well as the geometric standard deviation (`sigma_g`) and density (`rho`) are specified. For an aerosol tracer, that shall be associated to this mode, the meta data mode has to be set to `seasa` at the tracer definition (see previous section). In general, all available modes are listed in the example file `modes.xml`. Hence, it is highly recommended to adapt this file according to your simulation setup.

10.4.5. Point Source Module: `pntSrc`

ICON-ART provides a module which adds emissions from point sources to existing tracers. The namelist switches associated to this module are `lart_passive`, `lart_pntSrc` and `cart_pntSrc.xml` (see Section 10.4.2).

The prerequisite is that you have added a passive tracer via an XML file using the namelist parameter `cart_passive.xml`. Starting from this point, point sources can be added using an XML file specified via `cart_pntSrc.xml`. Additionally, you have to set `lart_passive` and `lart_pntSrc` to `.TRUE..` Inside the XML file specified via `cart_pntSrc.xml`, you can add point sources following the subsequent example:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tracers SYSTEM "pntSrc.dtd">

<sources>
  <pntSrc id="RNDFACTORY">
    <substance type="char">testtr</substance>
    <lon type="real">8.00</lon>
    <lat type="real">48.00</lat>
    <source_strength type="real">1.0</source_strength>
    <height type="real">10.</height>
    <unit type="char">kg s-1</unit>
  </pntSrc>
</sources>

```

The options you can specify here have the following meaning:

- **pntSrc id:** The name of the point source. This information is actually not used in the ICON-ART code and serves only for a better readability of the XML file. Hence, also multiple point sources with the same id are technically allowed.
- **substance:** This is the name of the substance, the point source emission is added to. Here you have to specify the very same name of the tracer that you have specified in the `cart_passive_xml` file.
- **lon:** Geographical longitude of the point source in degrees north.
- **lat:** Geographical latitude of the point source in degrees east.
- **source_strength:** The source strength of the point source in the unit specified below.
- **height:** The height of the point source in meters above ground.
- **unit:** The unit of the source strength. Note, that currently every unit different from kg s-1 will lead to a model abort, as no unit conversion is implemented so far.

Via the XML file you can also specify multiple point sources. By this, you can either add point sources to different tracers or specifying multiple source for a single tracer with for example differing source strengths.

10.4.6. Volcanic Ash Control

If volcanic eruptions should be considered the switch `iart_volcano` has to be set. For the 1-moment description of volcanic ash, i.e. six monodisperse size bins for the number concentrations, the integer value is 1. For the 2-moment description where 3 lognormal modes are used, the switch has to be set to 2.

Further input is necessary to define the appropriate volcano(s):

- `volcano_list_whatyouwant.txt` has to contain information for the initialization of each volcano. The path including filename has to be set in the `namelist (cart_volcano_file)`. The line(s) for the volcano(s) of interest can be copied from the file `of2009-1133_table3_EDITED_HISTORICAL.txt` in the `runctrl_examples/emiss_ctrl` folder. In Figure 10.1 an example is given. The first column contains an ID of the volcano which is not read but needed due to format specifications. The next column contains the name of the volcano. It is followed by information on its location. There the correct longitude and latitude must be present. Afterwards information on the volcano type is given.

```
1702-02= 17 02 -02- Eyjafjocell Iceland-S Historical 63.63 N X 19.62 W 1666 Stratovolcano D3 S0
```

Figure 10.1.: Example for the content of `volcano_list_name.txt`.

With this setup ICON-ART performs a simulation with the standard parameters for the respective volcano type.

10.5. Output

In principle, output of ICON-ART variables works the same way as for ICON variables. The following five quantities of the output have to be specified:

- The time interval between two model outputs.
- The name of the output file.
- The name of the variable(s) and/or variable group(s).
- The type of vertical output grid.
- The type of horizontal output grid.

In general, **the output of all (prognostic) tracers defined in the different XML files (passive, chemistry, aerosol) is possible**. Additionally, several diagnostic output variables have been added in ICON-ART. These are listed in table 10.6.

There is an option to obtain all variables belonging to a certain group without having to specifying all of them. The output variables that are associated to this group will be written. Available output groups are: ART_AERO_VOLC¹, ART_AERO_RADIO², ART_AERO_DUST³, ART_AERO_SEAS⁴, ART_CHEMTRACER⁵ and ART_PASSIVE⁶. As the names indicate, these groups contain variables associated to ¹volcanic ash aerosol, ²radioactive particles, ³mineral dust aerosol, ⁴sea salt aerosol, ⁵chemical tracer and ⁶ passive tracer.

Table 10.6.: Selected list of available diagnostic output fields for aerosol.

Variable	Associated namelist switch	Description	Groups
seasa_diam seasb_diam seasc_diam	iart_seasalt = 1 lart_diag_out = .true.	Median diameter of sea salt mode A, B, C respectively	ART_AERO _SEAS
asha_diam ashb_diam ashc_diam	iart_volcano = 2 lart_diag_out = .true.	Median diameter of volcanic ash mode A, B, C respectively	ART_AERO _VOLC
tau_volc_340nm tau_volc_380nm tau_volc_440nm tau_volc_500nm tau_volc_550nm tau_volc_675nm tau_volc_870nm tau_volc_1020nm tau_volc_1064nm	iart_volcano = 2 lart_diag_out = .true.	Volcanic ash optical depth at the wavelength indicated by the name	ART_AERO _VOLC
ash_total_mc	iart_volcano = 2 lart_diag_out = .true.	Total concentration of volcanic ash in column	ART_AERO _VOLC

11. ICON's Data Assimilation System

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).

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 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.

11.1.1. Variational Data Assimilation

The basic 3D-VAR step minimizes the functional

$$\mu(x) := \|x - x^{(b)}\|_{B^{-1}}^2 + \|y - H(x)\|_{R^{-1}}^2, \quad (11.1)$$

ICON Basic Cycling Environment

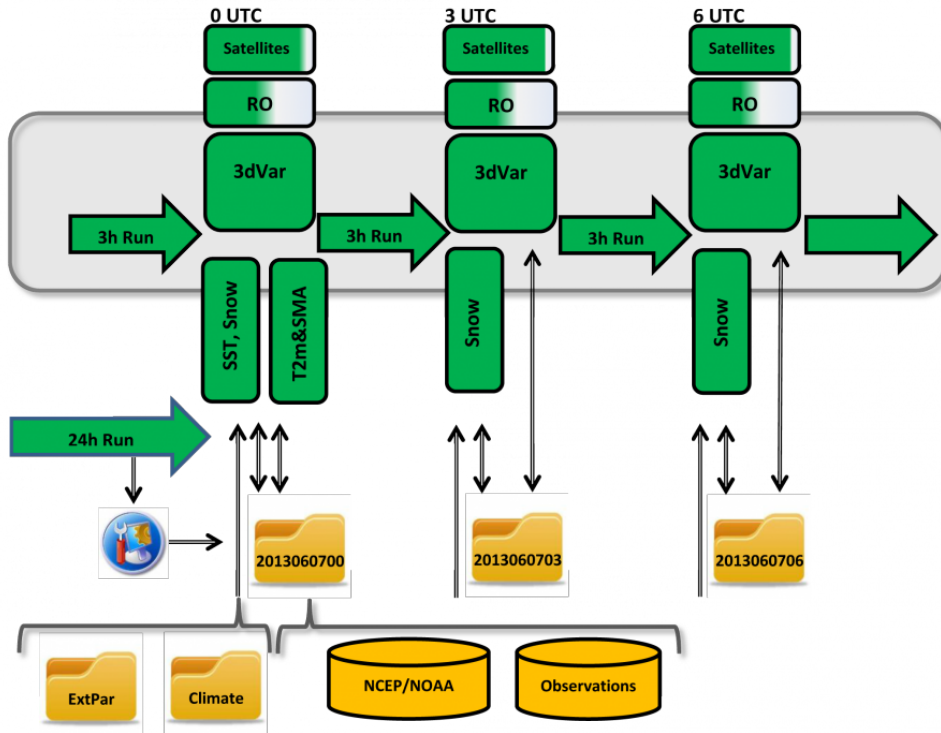


Figure 11.1.: Basic ICON cycling environment using 3DVar. Observations are merged with a background field taken from a 3 h forecast (first guess) of the ICON model. *Courtesy of R. Potthast, DWD.*

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

$$(R + HBH^T)z = y - H(x_k^{(b)})$$

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 ran: 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. The analysis fields necessary to initialize the nest are interpolated from the underlying global grid. 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 time of the whole system is again limited to +30 h lead time at 03/09/15/21 UTC. At 00/12 UTC the system is iterated up to 120 h. The primary purpose of the ensemble forecasts is to estimate the forecast uncertainty, which arises due to uncertainties in initial conditions and 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 C 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 C 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 C 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 C 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, 2 m-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.

A. The Computer System at DWD

Available Platforms at DWD

The NWP exercises this week will be mainly carried out on the Cray XC 40 supercomputer system at DWD. This system consists of several compute nodes with corresponding service nodes. Some of the service nodes are the so-called *login*-nodes (with names `xce00.dwd.de` and `xce01.dwd.de`), on which you can use training accounts.

- `xce00`, `xce01`:
These are the login nodes, which run a SUSE Linux Enterprise (SLES) Linux. The nodes are used for compiling and linking, preparation of data, basic editing work and visualization of meteorological fields. They are not used for running parallel programs, but jobs can be submitted to the Cray XC 40 compute nodes.
- Cray XC 40:
The Cray XC 40 has 432 compute nodes, where each node is equipped with 2 Intel Haswell processors with 12 cores and a second node partition with 864 Intel Broadwell nodes. Each Haswell node therefore has 24 computational cores (Broadwell: 36). These nodes cannot be accessed interactively, but only by batch jobs. Such jobs can use up to 62 GByte of main memory per node, which is about 2.5 GByte per core (Broadwell: 1.7 GByte). For normal test jobs it should be enough to use 10-15 nodes (depending on the chosen grid resolution).
- `lce00`, `lce01`:
Visualization tools (CDO, NCL, `ncview`) are only available on the pre-/post-processing cluster `lce`. This cluster comprises 66 Haswell batch nodes, but we will only use the login nodes `lce00`, `lce01` during the course.

There is a common filesystem across all nodes and every user has three different main directories:

- `/e/uhome/username` (`$HOME`)
Directory for storing source code and scripts to run the model. This is a Panasas file system suitable for many small files.
- `/e/uwork/username` (`$WORK`)
Directory for storing larger amounts of data.
- `/e/uscratch/username` (`$SCRATCH`)
Directory for storing very large amounts of data. For the `$WORK` and the `$SCRATCH` filesystem there is a common quota for every user of 2.5 TByte.

The Batch System for the Cray XC 40

Jobs for the Cray XC 40 system have to be submitted with the batch system PBS. These batch jobs may either be launched from the Linux cluster `lce` or from the XC 40 login nodes `xce00/01`. Together with the source code of the programs we provide some run scripts in which all necessary batch-commands are set.

Here are the most important commands for working with the PBS:

<code>qsub <i>job_name</i></code>	to submit a batch job to PBS. This is done in the run scripts.
<code>qstat</code>	to query the status of all batch jobs on the XC 40. You can see whether jobs are Q (queued) or R (running). You have to submit jobs to the queue <code>xc_norm_h</code> .
<code>qstat -u <i>user</i></code>	to query the status of all your batch jobs on the machine.
<code>qdel <i>job_nr@machine</i></code>	to cancel your job(s) from the batch queue of a machine. The <i>job_nr</i> is given by <code>qstat -w</code> .
<code>qcat -f <i>job_nr@machine</i></code>	To follow stdout and stderr interactively.

In your run scripts, execution begins in your home directory, regardless of what directory your script resides in or where you submitted the job from. You can use the `cd` command to change to a different directory. The environment variable `$PBS_O_WORKDIR` makes it easy to return to the directory from which you submitted the job:

```
cd $PBS_O_WORKDIR
```

B. Troubleshooting

When you work with the ICON software package, you can have a lot of trouble. Some of the problems are due to imperfect (or even missing) documentation, others are caused by program bugs. We apologize right now for any inconvenience this may cause. But there are also troubles resulting from compiler bugs or hardware problems.

In the following, we want to describe some problems that can occur during the single phases you are going through when installing and running the model. If possible, we also want to give some hints for the solution.

Troubleshooting: Compiling and Linking

These are the most common difficulties when compiling and linking the software:

- *Failing compilation process due to syntax errors.*
The ICON code requires compiler support for a rather recent version of the Fortran programming language. A Fortran 2003 compliant compiler is necessary. 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 preprocessor. Please note that due to the complex structure of the source code there may occur (rare) cases of compiler bugs. There may be even compiler bugs that manifest themselves in *internal compiler errors* or run-time failures. In particular, the Cray Fortran compiler versions $8.1.9 < version < 8.3.0$ are known to fail for the ICON model code.
- *Failing configuration or linking process due to missing libraries.*
The ICON model code requires an installation of the NetCDF library and the GRIB-API library. Furthermore, binaries for distributed-memory parallel runs require the MPI library. If problems occur during the linking process, check the prerequisites that have been outlined in Section 1.2.1. If the configuration still fails, though all three libraries are available, 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.
- *No pre-defined configuration for your platform is available.*
Inside the `config` directory, different machine dependent configurations are stored within the configuration files. To add a specific compiler or change your compiler flags, you have to enter the `config/mh-OS` according to your operating system `OS`.

Troubleshooting in NWP Mode

What to Do If the DWD ICON Tools Fail?

- *IFS2ICON*: Check the data files retrieved from the ECMWF MARS database. All fields that are defined in your `iconremap` namelist settings must be contained in this input data file. For GRIB1 input data, provide the correct field parameter with the namelist parameter `code`.
- Test, if the computational resources, i.e. memory and the number of processors, are sufficient to run the model. Otherwise change the resource settings in your batch queue script. If the ICON tools are run sequentially or in shared-memory mode (with OpenMP, but without MPI), consider running `iconremap` in distributed memory mode with MPI. The ICON Tools themselves contain example run scripts with resource setups and the ICON Tools documentation provides information on OpenMP environment variables.
- If the cause of the error still does not become clear, you may increase model output verbosity by setting the command-line options `-v`, `-vv`, `-vvv` etc.
- Stop right there, and don't move. Speak to the bear in a low, calm voice, and slowly raise your arms up above your head. Clearly, you should try to leave now. Do it slowly and go back from whence you came. Don't cross the path of the bear (or any cubs, if present).

What to Do If the ICON Model Run Fails?

- *The model aborts due to missing or incorrect input files.*
 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, cf. 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 global grid file. These values have to match the corresponding attributes of the external parameters and initial data file.
- *The model aborts due to incorrect namelist settings.*
 Of course, namelist settings may lead to a model crash. This may be the case if the settings are incorrect with regards to content, but oftentimes a namelist setup can also be syntactically wrong, e.g. when a different data type is expected by the model. This happens especially if shell script variables are inserted into the namelists by the batch queueing script. Therefore, check the namelist settings for obvious technical errors:
 First of all, the run scripts in this tutorial create a file `NAMELIST.NWP` which contains all user-defined namelist parameters, together with the substituted shell script variables. Furthermore, during each ICON run, the file `nml.atmo.log` is created automatically, which contains all namelist parameters, including the default settings that have not been touched by the user settings. Please note that there is a small

caveat: Defaults defined in the ICON code after the namelist read-in are not monitored.

Finally, your namelist settings may have been specified w.r.t. a former version of ICON. Then, there may be the case that certain parameters have been declared deprecated. Take a look at the namelist documentation `doc/Namelist_overview.pdf` which comes with your version of the ICON model code. This document contains a section on incompatible changes, and may provide some useful hints.

- *The model aborts due to lack of resources.*
Please check, if the computational resources, i.e. memory and the number of processors, are sufficient to run the model. Otherwise change the resource settings in your batch queue script. If the ICON model is run sequentially or in shared-memory mode (with OpenMP, but without MPI), consider running the model in distributed memory mode with MPI.
- If the cause of the error still does not become clear, you may increase the model output verbosity, see the namelist parameter `msg_level` in the namelist `run_nml`.

There are surely many more reasons for problems and errors, whose discussion goes beyond the scope of this tutorial. It really gets troublesome when a program aborts and writes core files. Then you will need some computer tools (like a debugger) to investigate the problem. If you cannot figure out what the reason for your problem is we can try to give some support.

C. Table of ICON 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 7.1 for details.

By "ICON-internal" variable names, we denote 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.

Therefore, if you are interested in the model output of a certain variable and if this variable is not listed in the table below, you may search for the corresponding call to `add_var/add_ref` in the source code instead.

Variable name	Description
<code>acdnc</code>	cloud droplet number concentration
<code>adrag_u_grid</code>	zonal resolved surface stress mean since model start
<code>adrag_v_grid</code>	meridional resolved surface stress mean since model start
<code>alb_dif</code>	Shortwave albedo for diffuse radiation
<code>albdif</code>	Shortwave albedo for diffuse radiation
<code>albni_dif</code>	Near IR albedo for diffuse radiation
<code>albnirdif</code>	Near IR albedo for diffuse radiation
<code>albnirdir</code>	Near IR albedo for direct radiation
<code>alb_si</code>	sea ice albedo (diffuse)
<code>albu_v_dif</code>	UV visible albedo for diffuse radiation
<code>albvisdif</code>	UV visible albedo for diffuse radiation
<code>albvisdir</code>	UV visible albedo for direct radiation
<code>alhfl_bs</code>	latent heat flux from bare mean since model start
<code>alhfl_pl</code>	latent heat flux from plantmean since model start
<code>alhfl_s</code>	surface latent heat flux mean since model start
<code>aqhfl_s</code>	surface moisture flux mean since model start
<code>ashfl_s</code>	surface sensible heat flux mean since model start
<code>asob_s</code>	Surface net solar radiation mean since model start
<code>asob_t</code>	TOA net solar radiation mean since model start
<code>asodifd_s</code>	Surface down solar diff. rad. mean since model start
<code>asodifu_s</code>	Surface up solar diff. rad. mean since model start
<code>asodird_s</code>	Surface down solar direct rad.mean since model start
<code>asod_t</code>	Top down solar radiation mean since model start
<code>asou_t</code>	Top up solar radiation mean since model start
<code>astr_u_sso</code>	zonal sso surface stress mean since model start
<code>astr_v_sso</code>	meridional sso surface stress mean since model start

Continued on next page

¹The Table C.1 in this Appendix is based on the revision state e5ae09fe (2017-02-06).

Table C.1 – Continued from previous page

Variable name	Description
aswflx_par_sfc	Downward PAR flux mean since model start
athb_s	surface net thermal radiation mean since model start
athb_t	TOA net thermal radiation mean since model start
athd_s	Surface down thermal radiation mean since model start
athu_s	Surface up thermal radiation mean since model start
aumfl_s	u-momentum flux flux at sumean since model start
avg_qc	tci_specific_cloud_water_content (diagnostic)_avg
avg_qi	tci_specific_cloud_ice_content (diagnostic)_avg
avg_qv	column integrated water vapour (diagnostic)_avg
avmfl_s	v-momentum flux flux at sumean since model start
aw	area weights for regular lat-lon grid
cape	conv avail pot energy
cape_ml	cape of mean surface layer parcel
cin_ml	convective inhibition of mean surface layer parcel
clc	cloud cover
clch	high_level_clouds
clcl	low_level_clouds
clcm	mid_level_clouds
clct_avg	total cloud cover time avg
clct_mod	modified total cloud cover for media
clct	total cloud cover
cldepth	modified cloud depth for media
cloud_num	cloud droplet number concentration
con_gust	convective contribution to wind gust
con_prec_rate_avg	convective precip rate, time average
cosmu0	Cosine of solar zenith angle
c_t_lk	shape factor (temp. profile in lake thermocline)
ddt_exner_phy	Exner pressure physical tendency
ddt_pres_sfc	surface pressure tendency
ddt_qc_conv	convective tendency of specific cloud water
ddt_qc_gscp	microphysics tendency of specific cloud water
ddt_qc_turb	turbulence tendency of specific cloud water
ddt_qi_conv	convective tendency of specific cloud ice
ddt_qi_gscp	microphysics tendency of specific cloud ice
ddt_qi_turb	turbulence tendency of specific cloud ice
ddt_qr_gscp	microphysics tendency of rain
ddt_qs_gscp	microphysics tendency of snow
ddt_qv_conv	convective tendency of specific humidity
ddt_qv_gscp	microphysics tendency of specific humidity
ddt_qv_turb	turbulence tendency of specific humidity
ddt_temp_drag	sso + gwdrag temperature tendency
ddt_temp_dyn	dynamical temperature tendency
ddt_temp_gscp	microphysical temperature tendency
ddt_temp_pconv	convective temperature tendency
ddt_temp_radlw	long wave radiative temperature tendency
ddt_temp_radsw	short wave radiative temperature tendency
ddt_temp_turb	turbulence temperature tendency

Continued on next page

Table C.1 – *Continued from previous page*

Variable name	Description
ddt_tke_hsh	TKE tendency horizontal shear production
ddt_tke_pconv	TKE tendency due to sub-grid scale convection
ddt_tke	tendency of turbulent velocity scale
ddt_u_gwd	GWD tendency of zonal wind
ddt_u_pconv	convective tendency of zonal wind
ddt_u_sso	sso tendency of zonal wind
ddt_u_turb	turbulence tendency of zonal wind
ddt_v_gwd	GWD tendency of meridional wind
ddt_vn_phy	normal wind physical tendency
ddt_v_pconv	convective tendency of meridional wind
ddt_v_sso	sso tendency of meridional wind
ddt_v_turb	turbulence tendency of meridional wind
depth_lk	lake depth
dgeopot_mc	geopotential at cell center
div	Divergence
div_ic	divergence at half levels
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	meridional resolved surface stress
dtheta_v_ic_abc	potential temperature at child upper boundary
dv_n_ie_int	normal velocity at parent interface level
dv_n_ie_abc	normal velocity at child upper boundary
dwdx	Zonal gradient of vertical wind
dwdy	Meridional gradient of vertical wind
dw_abc	vertical velocity at child upper boundary
dyn_gust	dynamical gust
eai	(evaporative) Earth area index
emis_rad	longwave surface emissivity
exner_dyn_incr	Exner dynamics increment
exner	Exner pressure
exner_incr	Exner increment from DA
exner_pr	Exner perturbation pressure
exner_ref_mc	Reference atmosphere field Exner
fetch_lk	wind fetch over lake
fis	Geopotential (s)
for_d	Fraction of deciduous forest
freshsnow	weighted indicator for age of snow in top of snow layer
fr_glac	Fraction glacier
fr_lake	fraction lake
fr_land	Fraction land
fr_seaice	fraction of sea ice
gamso_lk	attenuation coefficient of lake water with respect to sol. rad.
geopot_agl	geopotential above groundlevel at cell center
geopot_agl_ifc	geopotential above groundlevel at cell center
geopot	geopotential at full level cell centre
grf_tend_mflx	normal mass flux tendency (grid refinement)

Continued on next page

Table C.1 – Continued from previous page

Variable name	Description
grf_tend_rho	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)
gsp_prec_rate_avg	gridscale precip rate, time average
gust10	gust at 10 m
gz0	roughness length
h_b1_lk	thickness of the upper layer of the sediments
hbas_con	height_of_convective_cloud_base
hdef_ic	Deformation
h_ice	sea/lake-ice depth
h_ml_lk	mixed-layer thickness
hmo3	height of O3 maximum (Pa)
h_snow_lk	depth of snow on lake ice
h_snow_si	depth of snow on sea ice
h_snow	weighted snow depth
htop_con	height_of_convective_cloud_top
htop_dc	height_of_top_of_dry_convection
hzerocl	height_of_0_deg_C_level
k400	level index corresponding to the HAG of the 400hPa 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
ktype	type of convection
lai	Leaf Area Index
lc_class_t.xx	tile point land cover class
lhfl_bs	latent heat flux from bare soil
lhfl_pl	latent heat flux from plants
lhfl_s	surface latent heat flux
lwflxall	longwave net flux
mask_mtnpoints_g	Mask field for mountain points
mask_mtnpoints	Mask field for mountain points
mass_fl_e	horizontal mass flux at edges
mass_fl_e_sv	storage field for horizontal mass flux at edges
ndvi_max	NDVI yearly maximum
ndviratio	(monthly) proportion of actual value/maximum NDVI (at init time)
o3	ozone mixing ratio
omega	vertical velocity
omega_z	vertical vorticity
pat_len	length scale of sub-grid scale roughness elements
plcov	Plant covering degree in the vegetation phase
pres_ifc	pressure at half level
pres_msl	mean sea level pressure
pres	Pressure
pres_sfc	surface pressure

Continued on next page

Table C.1 – *Continued from previous page*

Variable name	Description
pv	Ertel Potential Vorticity
qc	specific_cloud_water_content
qc	specific_cloud_water_content
qhfl_s	surface moisture flux
qi	specific_cloud_ice_content
qi	specific_cloud_ice_content
qr	rain_mixing_ratio
qr	rain_mixing_ratio
qs	snow_mixing_ratio
qs	snow_mixing_ratio
qv_2m	specific water vapor content in 2m
qv_incr	specific humidity increment from DA
qv	Specific humidity
qv	Specific humidity
qv_s	specific humidity at the surface
rain_con	convective rain
rain_con_rate_3d	3d convective rain rate
rain_con_rate	convective rain rate
rain_gsp	gridscale rain
rain_gsp_rate	gridscale rain rate
rain_upd	rain in updrafts
rcld	standard deviation of the saturation deficit
rh_2m_land	relative humidity in 2m over land fraction
rh_2m	relative humidity in 2m
rho	density
rho_ic	density at half level
rho_incr	density increment from DA
rho_ref_mc	Reference atmosphere field density
rho_ref_me	Reference atmosphere field density
rho_snow	weighted snow density
rh	relative humidity
rootdp	root depth of vegetation
rsmin	Minimal stomata resistance
rstom	stomatal resistance
runoff_g	weighted soil water runoff; sum over forecast
runoff_s	weighted surface water runoff; sum over forecast
sai	surface area index
shfl_s	surface sensible heat flux
slope_angle	Slpe angle
slope_azimuth	Slpe azimuth
smi	soil moisture index
snow_con	convective snow
snow_con_rate_3d	3d convective snow rate
snow_con_rate	convective snow rate
snowfrac_lc	snow-cover fraction
snowfrac_lc.t.xx	tile-based snow-cover fraction
snowfrac	snow-cover fraction

Continued on next page

Table C.1 – *Continued from previous page*

Variable name	Description
snowfrac_t_xx	local tile-based snow-cover fraction
snow_gsp	gridscale snow
snow_gsp_rate	gridscale snow rate
snowlmt	Height of snow fall limit above MSL
sob_s	shortwave net flux at surface
sob_t	shortwave net flux at TOA
sodifd_s	shortwave diffuse downward flux at surface
sod_t	downward shortwave flux at TOA
soiltyp	soil type
sou_s	shortwave upward flux at surface
sou_t	shortwave upward flux at TOA
sp_10m	wind speed in 10m
sso_gamma	Anisotropy of sub-gridscale orography
sso_sigma	Slope of sub-gridscale orography
sso_stdh_raw	Standard deviation of sub-grid scale orography
sso_stdh	Standard deviation of sub-grid scale orography
sso_theta	Angle of sub-gridscale orography
str_u_sso	zonal sso surface stress
str_v_sso	meridional sso surface stress
swflx_par_sfc	downward photosynthetically active flux at surface
t_2m_land	temperature in 2m over land fraction
t_2m	temperature in 2m
tai	transpiration area index
t_b1_lk	temperature at the bottom of the upper layer of the sediments
t_bot_lk	temperature at the water-bottom sediment interface
t_bs_lk	clim. temp. at bottom of thermally active layer of sediments
tch	turbulent transfer coefficients for heat
t_c1	CRU near surface temperature climatology
tcm	turbulent transfer coefficients for momentum
td_2m	dew-point in 2m
td_2m_land	dew-point in 2m over land fraction
temp_ifc	temperature at half level
temp	Temperature
tempv	Virtual temperature
tfh	factor of laminar transfer of scalars
tfm	factor of laminar transfer of momentum
tfv	laminar reduction factor for evaporation
t_g	weighted surface temperature
thb_s	longwave net flux at surface
theta_ref_ic	Reference atmosphere field theta
theta_ref_mc	Reference atmosphere field theta
theta_ref_me	Reference atmosphere field theta
theta_v_ic	virtual potential temperature at half levels
theta_v	virtual potential temperature
thu_s	longwave upward flux at surface
t_ice	sea/lake-ice temperature
tke	turbulent kinetic energy
tkred_sfc	reduction factor for minimum diffusion coefficients

Continued on next page

Table C.1 – *Continued from previous page*

Variable name	Description
tkr	turbulent reference surface diffusion coefficient
tkvh	turbulent diffusion coefficients for heat
tkvm	turbulent diffusion coefficients for momentum
tmax_2m	Max 2m temperature
tmin_2m	Min 2m temperature
t_mnw_lk	mean temperature of the water column
topography_c	geometric height of the Earth's surface above sea level
tot_prec_rate_avg	total precip rate, time average
tot_prec	total precip
tot_qc_dia	total_specific_cloud_water_content_(diagnostic)
tot_qi_dia	total_specific_cloud_ice_content_(diagnostic)
tot_qv_dia	total_specific_humidity_(diagnostic)
tqc_dia	total column integrated cloud water (diagnostic)
tqc	total_column_integrated_cloud_water
tqi_dia	total column integrated cloud ice (diagnostic)
tqi	total_column_integrated_cloud_ice
tqr	total_column_integrated_rain
tqs	total_column_integrated_snow
tqv_dia	total column integrated water vapour (diagnostic)
tqv	total_column_integrated_water_vapour
tracer_vi_avg01	tracer_vi_avg
tracer_vi_avg02	tracer_vi_avg
tracer_vi_avg03	tracer_vi_avg
trsolall	shortwave net tranmissivity
t_seasfc	sea surface temperature
tsfc_ref	Reference surface temperature
tsfctrad	surface temperature at trad
t_snow_lk	temperature of snow on lake ice
t_snow_si	temperature of snow on sea ice
t_snow	weighted temperature of the snow-surface
t_so	weighted soil temperature (main level)
t_s	weighted temperature of ground surface
tvh	turbulent transfer velocity for heat
tvm	turbulent transfer velocity for momentum
t_wml_lk	mixed-layer temperature
u_10m	zonal wind in 10m
umfl_s	u-momentum flux at the surface
u	Zonal wind
v_10m	meridional wind in 10m
vio3	vertically integrated ozone amount
v	Meridional wind
vmfl_s	v-momentum flux at the surface
vn_ie	normal wind at half level
vn	velocity normal to edge
vor	Vorticity
vt	tangential-component of wind
vwind_expl_wgt	Explicit weight in vertical wind solver

Continued on next page

Table C.1 – Continued from previous page

Variable name	Description
<code>wwind_impl_wgt</code>	Implicit weight in vertical wind solver
<code>w_concorr_c</code>	contravariant vertical correction
<code>w_i_t_xx</code>	weighted water content of interception water
<code>w_i</code>	weighted water content of interception water
<code>w_snow_t_xx</code>	water equivalent of snow
<code>w_snow</code>	weighted water equivalent of snow
<code>w_so_ice</code>	ice content
<code>w_so_ice_t_xx</code>	ice content
<code>w_so</code>	total water content (ice + liquid water)
<code>w_so_t_xx</code>	total water content (ice + liquid water)
<code>w</code>	Vertical velocity
<code>ww</code>	significant_weather
<code>z_ifc</code>	geometric height at half level center
<code>z_mc</code>	geometric height at full level center

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).

Group ADDITIONAL_PRECIP_VARS

`cape`, `clct`, `clct_mod`, `con_prec_rate_avg`, `gsp_prec_rate_avg`, `tqc_dia`, `tqi_dia`, `tdq_dia`

Group ATMO_DERIVED_VARS

`div`, `omega_z`, `vor`

Group ATMO_ML_VARS

`pres`, `qc`, `qg`, `qi`, `qr`, `qs`, `qv`, `temp`, `tke`, `u`, `v`, `w`

Group ATMO_PL_VARS

`qc`, `qg`, `qi`, `qr`, `qs`, `qv`, `temp`, `tke`, `u`, `v`, `w`

Group ATMO_ZL_VARS

`pres`, `qc`, `qg`, `qi`, `qr`, `qs`, `qv`, `temp`, `tke`, `u`, `v`, `w`

Group CLOUD_DIAG

`clc`, `tot_qc_dia`, `tot_qi_dia`, `tot_qv_dia`

Group DWD_FG_ATM_VARS

`pres`, `pres_sfc`, `qc`, `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_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_snow_t_*, t_so_t_*, w_i_t_*, w_snow_t_*, w_so_ice_t_*, w_so_t_*

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_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, qi, qr, qs, qv, rho, temp, theta_v, u, v, vn, w, z_ifc

Group MODE_COMBINED_IN

fr_seaice, freshsnow, h_ice, h_snow, qv_s, rho_snow, t_g, t_ice, 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, qi, qr, qs, qv_s, rho, rho_snow, t_bot_lk, t_g,
t_mnw_lk, 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, qv, temp, u, v

Group MODE_IAU_ANA_IN

fr_seaice, freshsnow, h_snow, pres, 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, qi, qr, qs, qv, qv_s, rho,
rho_snow, snowfrac_lc, 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_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, 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 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, tvm, u_10m, v_10m

Group PHYS_TENDENCIES

ddt_temp_drag, ddt_temp_radlw, ddt_temp_rads, ddt_temp_turb, ddt_u_gwd,
ddt_u_sso, ddt_u_turb, ddt_v_gwd, ddt_v_sso, ddt_v_turb

Group PRECIP_VARS

graupel_gsp, rain_con, rain_gsp, snow_con, snow_gsp, tot_prec

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, sob_s,
sob_s_t_*, sob_t, sod_t, sodifd_s, sou_s, sou_t, swflx_par_sfc, thb_s, thb_s_t_*,
thu_s

Group SNOW_VARS

rho_snow, t_snow

D. Exercises

List of Exercises

Installation of the ICON Model Package	192
Exercise 1.1	192
Necessary Input Data	192
Exercise 2.1: Grid Generator and ExtPar Web Interface	192
Exercise 2.2: Retrieving DWD Analysis Data	193
Exercise 2.3: Retrieving IFS Data	194
Exercise 2.4: Preparation of Limited Area Runs	194
Running Idealized Test Cases	196
Exercise 4.1: Jablonowski-Williamson Test Case	196
Exercise 4.2: Nested Sub-domain	197
Exercise 4.3: Tracer Advection	198
Real Data Test Cases	199
Exercise 5.1: Starting a Global ICON Forecast from DWD Analysis	200
Exercise 5.2: Time-stepping	201
Exercise 5.3: Run Time Performance	202
Exercise 5.4: Parallelization	202
Exercise 5.5: Writing Output for Driving a Limited Area Simulation	203
Exercise 5.6: A Deeper Look into Spin-Up Effects	205
Exercise 5.7: Checking for Bit-Reproducibility	206
Limited Area Mode	206
Exercise 6.1: Limited Area Run	207
Exercise 6.2: Modifying the Temporal Resolution	209
Programming ICON	210
Exercise 8.1: Implementing New Diagnostics	210
Running ICON-ART	212
Exercise 10.1: Point Sources in ICON-ART LAM	212
Exercise 10.2: Very Short-lived Substances	212
Exercise 10.3: Volcano Eruption	213
Exercise 10.4: Sea Salt Aerosol	214

D.1. Installation of the ICON Model Package

For practical work during these exercises, you need information about the use of the computer systems and from where you can access the necessary files and data.

Please take a look at Appendix A to get information on how to use DWD's supercomputer and run test jobs.



EX 1.1

Log into the Cray XC 40 login node "xce" and **install the ICON model** in the \$WORK directory of your Cray XC 40 user account. For that you have to do the following:

- The necessary files for the ICON tutorial can be found in the sub-directory

`/e/uwork/trng024/packages`

Copy the tar-file `icon_tutorial.tar.gz` into your \$WORK directory.

- Change into your \$WORK directory and extract the compressed tar file to yield the directory structure depicted in Figure 1.1 containing the ICON sources and test data.
- Follow the instructions in Section 1.2.2 to configure and compile the ICON model on the Cray XC 40 platform.

Note:

Note that you can also compile the routines in parallel by using the GNU-make with the command `gmake -j np`

Install also the **DWD ICON Tools**:

- Change into the sub-directory `dwd_icon_tools` and build the DWD ICON Tools according to Section 1.3.2.

D.2. Necessary Input Data

Preparation of Global Runs

In this exercise, you will deal with the necessary preparatory steps for performing global real case ICON runs.



EX 2.1

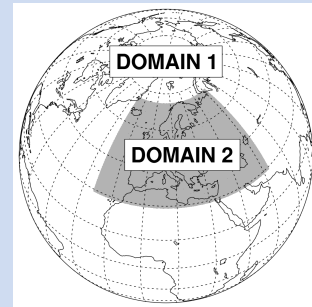
Grid generator and ExtPar web interface: Generate the necessary grids and external parameter files using the ICON web service (see Section 2.1.4).

- Open the DWD grid generator and ExtPar web interface

<https://webservice.dwd.de/cgi-bin/spp1167/webservice.cgi>

in your web browser.

- Select a sequence of three grids: Domain #0 – reduced radiation grid, domain #1 – global grid, domain #2 – refined region over Europe.
Make sure that you set the “parent grid ID” appropriately.
- For the reduced radiation grid, select a “synthetic grid” with R3B05 resolution. Select global refinement for the domain #1.
- Specify the R3B07 grid, domain #2, as a regular-shaped refinement region over Europe, see the illustration on the right or Fig. D.1:
Center lon/lat: 19.5 / 50.0 deg, half width lon/lat: 44.5 / 21.5 deg.
- Submit the web form – but do not forget to provide your e-mail address! Wait for the job to finish, this may take some time. Afterwards, pick the generated data according to the download information that you have received via e-mail.
- Copy and rename the grid files s. t. the file names match the nomenclature `iconR<nroot>B<jlev>_DOM<idom>.nc`, see Section 5.1.2.
Rename the ExtPar data to `extpar_DOM<idom>.nc`.
- Take a look at the grid data and the external parameters using the `ncdump` utility, see Section 9.1.1 for details. Find out whether the external parameter fields are defined at the grid vertices, edge midpoints or cells.



Retrieving initialized DWD analysis data for global forecast runs:

- Request native analysis data from DWD’s meteorological data management system SKY for the date 2017-06-01T00:00:00 at a horizontal grid spacing of 13 km using the `pamore` script, see Section 2.2.1:

```
~/for0exp/bin/pamore -d date -hstart 0 -hstop 0 -lt m \  
-model iglo -iglo_startdata_0
```

- Take a look at the data with the `grib_ls` command-line tool.
- Export the environment variable `GRIB_DEFINITION_PATH` with the setting `GRIB_DEFINITION_PATH=$GRIB_SAMPLES_PATH/./definitions` and run `grib_ls` once more. Are there any differences wrt. the displayed short names? See Section 1.1.2 for an explanation.

Remapping of initial data:

- Create a copy of the run script

```
dwd.icon_tools/icontools/xce_remap_inidata
```



EX 2.2

which performs the task of remapping the variables of an uninitialized analysis product (Table 2.2).

- Adapt the set of variables which is to be remapped for initialized analysis products, according to Table 2.3, i. e. replace the fields VN, THETA_V, DEN by the fields T, U, V, P.
- Run the script. Afterwards repeat the remapping procedure for the 13 km nest (R3B07) over Europe.
- Copy and rename the remapped analysis files s. t. the file names match the nomenclature `dwdANA_R<nroot>B<jlev>_DOM<idom>.grb`, see Section 5.1.2.



EX 2.3

Retrieving IFS data:

Prepare the initial data for ICON to start from an IFS analysis.

- *Download* – This step has to be executed on the ECMWF site:
Start the `mars4icon` script and download an IFS analysis file for the date 2017-01-12T00:00:00, see Section 2.2.2.
- Interpolate the data horizontally from the lat/lon grid onto the triangular ICON grid using `iconremap`. The run script

```
dwd.icon_tools/example/runscripts/xce_ifs2icon.run
```

will do this job.

Fill in the name of the IFS file by setting `in_grid_filename` and `in_filename`. Further help on `iconremap` can be found in Section 2.2.3.

- Submit the `iconremap` job to the Cray XC 40.
- List the fields contained in the output file using `cdo` and/or `ncdump`. Compare this to Table 2.4. Besides, find out which field(s) are not defined on cell circumcenters.

Preparation of Limited Area Runs

Note:

The following exercise requires a number of data files which are produced as model output in the real data exercise, Ex. D.5.5.



EX 2.4

Local grid file and its external parameters:

- In the directory `case_lam/input` you find an archive file which is the result of the web-based grid generator described in Section 2.1.4.

Uncompress this file and visualize its content with the NCL file `case_lam/plot_grid.ncl`.

See Figure D.3 for a reference (without highlighted boundary).

- Investigate the grid file with the `ncdump` utility (see Section 9.1.1): How many triangle cells are contained in the local grid? – Answer: cells

Remapping of initial data:

The directory “`lam_forcing`” generated by the real case run contains a file with the prefix `init`, which will serve as initial conditions for the limited area run. Remap the data onto the local grid.

The sub-directory `case_lam` contains a script `xce_remap_inidata` which will do the remapping.

- Insert the path to the ICON Tools binaries, and
- adapt the path to the input data file (directory “`lam_forcing`” generated by real case run).

Inspect the local (TARGET) grid file and the grid which was used in Ex. D.5.5 for creating the initial data (SOURCE).

- Identify the grid spacing of both grids in ICON’s `RxBY` nomenclature. (You could make use of `ncdump`.)

SOURCE grid	remap data	TARGET grid
<input style="width: 100px;" type="text"/>	→	<input style="width: 100px;" type="text"/>

- If the source grid has a horizontal grid spacing of ≈ 13 km, what is the grid spacing of your local (TARGET) grid in km, given the `RxBY` expressions identified above (see Eq. 2.1)?

– Answer: TARGET grid res. km

Run the script.

- Check the result: The remapping script should have created a file with prefix `init` in `case_lam/output`.

Remapping of boundary data:

- The sub-directory `case_lam` contains a copy of the DWD ICON Tools script `xce_limarea`, see Section 2.3. Open this script and
 - insert the path to the ICON Tools binaries,
 - adapt the path to input data files (directory “`lam_forcing`” generated by real case run).
- Run the script.
- Check the result: Visualize the boundary data with the NCL script `case_lam/plot_boundary_data.ncl`.

- Investigate the files: How many cells are contained in the boundary grid?

– Answer: cells

- When looking at the set of boundary data variables, do you have any idea for further improvement in terms of storage space?

– Answer:

D.4. Running Idealized Test Cases

In this exercise you will learn how to set up idealized runs in ICON with and without nested domains.

Job submission to the Cray XC 40 can be performed on the Linux cluster `lce`. **Note, however, that visualization tools (CDO, NCL, ncview) are *only* available on the `lce`!**

Running the Jablonowski-Williamson Test Case



EX 4.1

Preparations:

Retrieve the necessary grid file from the ICON download server (see Section 2.1.2):

- Open the download page for the pre-defined ICON grids <http://icon-downloads.mpimet.mpg.de> in your web browser.
- Pick the R2B05 grid no. 14 from the list and right-click on the hyperlink for the grid file, then choose "copy link location".
- Open a terminal window, login into the Linux cluster `lce`, and change into the sub-directory `test_cases/case_idealized/input`. Download the grid file into this sub-directory by typing

```
wget link_location -e http-proxy=ofsquid.dwd.de:8080.
```

Run the Jablonowski-Williamson (JW) test case without nested domains:

- Change into the run script directory `test_cases/case_idealized`. The run script is named `run_ICON_R02B05_JW`.
- Fill in the name of the ICON model binary (including the path) and several missing namelist parameters. I.e. set `ltestcase`, `ldynamics`, `iforcing`, `nh_test_name` and `itopo`. See Section 1.1.1 for the location of your ICON model binary and Section 4.2.1 for additional help on the namelist parameters.
- Submit the job to the Cray XC 40, using the PBS command `qsub`.

- Check the job status via `qstat` and `qcat` (see Section A).
- Go to the output directory of `case_idealized`. You will find three NetCDF output files with (a) model level output on the native (triangular) grid, (b) pressure level output on the native grid, (c) model level output interpolated onto a regular lat-lon grid.
- Have a closer look to the different output files and their internal structure to find out which one is which. We suggest to use the tools `cdo` and `ncdump` as described in Section 9.1.
What output interval (in h) has been used? – Answer: h
- Visualize the output with one of the tools described in Section 9. An NCL script named `JW_plot.ncl` is available in `test_cases/case_idealized`.
- Compare the output of the NCL script with the reference `JABW_DOM01.ps` given in sub-directory `reference`.

Repeat the run of the previous exercise with a nested sub-domain.

- Go to the run script directory.
The run script for a nested grid experiment is termed `run_ICON_R02B05N6_JW`.
- Fill in the name of your ICON model binary and missing namelist parameters. I.e. extend
 - `dynamics_grid_filename`,
 - `num_lev (run_nml)` and
 - `dynamics_parent_grid_id (grid_nml)`,

see Section 4.2.2 for additional help. The same output fields as for the global domain will be generated for the regional domain(s).

The grid files of the nested domains have already been prepared for you. They are stored in the sub-directory `test_cases/case_idealized/input`. The location of the prepared nests is depicted in Figure 4.3.

Feel free to add one or both nests to your global domain.

- Submit the job to the Cray XC 40.
- After the job has finished, visualize the output on the global domain as well as on nest(s) using one of the tools described in Section 9. When using NCL (`JW_plot.ncl`), you will have to adapt `workdir` and `domain_nr`.
- Compare the results on the global domain with those of the previous exercise. Does the global domain output differ? If so, do you have an explanation for this?

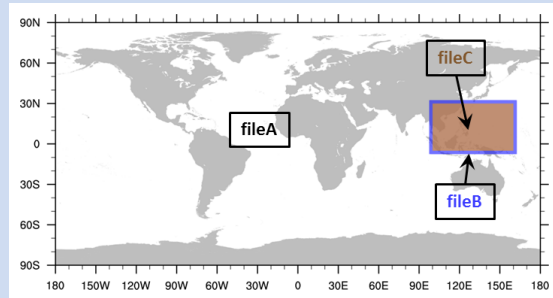
– Answer:



EX 4.2

Additional question: setting the parent grid ID.

The figure to the right depicts an alternative nest configuration, in which two domains are nested into each other. How does the correct setting for `dynamics_parent_grid_id` look like, given that the grid files are ordered as follows



`dynamics_grid_filename = 'fileA','fileB','fileC'`

– Answer: `dynamics_parent_grid_id =`

Optional Exercise: Tracer Advection



EX 4.3

The JW test case provides a set of four pre-defined idealized tracer fields (see Figure 4.2). Here, you will learn how to enable and control the transport of passive tracers in idealized tests. See Section 3.5.5 for details on the configuration of the tracer transport for standard NWP runs.

- Enable tracer advection:
 - Go to the directory `test_cases/case_idealized` and open the run script `run_ICON_R02B05N6_JW`.
 - Enable the transport module by setting the main switch `ltransport` to `.TRUE..`
 - Enable (uncomment) the namelist `transport_nml`, which specifies details of the applied transport scheme.
 - Select one or more tracers from the set of pre-defined tracer distributions (see Figure 4.2). A specific distribution can be selected by adding the respective tracer number (1,2,3, or 4) to `tracer_inidist_list` (namelist `nh_testcase_nml`, comma-separated list of integer values).
- Extend the output namelist
 - Add the selected tracer fields to the list of output fields in the namelists `output_nml` (namelist parameters `m1_varlist` and `p1_varlist`).

For idealized test cases, tracer fields are named `qx`, where `x` is equal to the suffix specified via the namelist variable `tracer_names` (namelist `transport_nml`, comma-separated list of string parameters). The n^{th} entry in `tracer_names` specifies the suffix for the n^{th} entry in `tracer_inidist_list`.

If nothing is specified, tracer fields are named qx , where x is a number indicating the position of the tracer within the ICON-internal 4D tracer container.

- Visualization: Enable `lplot_transport` in your NCL script. You can also have a quick look using `ncview`.
- Have a look at tracer `q4`. Initially it is constant ($= 1$) everywhere. Ideally, an initially constant tracer should stay constant for all times, no matter how complicated the flow. Does ICON preserve a constant tracer with/without nest?
- Avoid nonphysical negative values for tracer `q1`.
 - Every transport scheme that is more than first order accurate generates spurious over- and undershoots in the advected quantity. E.g. have a look at tracer `q1`. You will notice that spurious negative values emerge after some time. In NWP runs, such negative values can lead to numerical instabilities and must be avoided.
 - Switch on appropriate filter/limiter such that the transport scheme for `q1` becomes at least positive definite (i.e. $q1(t) \geq 0 \forall t$). See Section 3.5.5 for additional help. Repeat the run to confirm that your settings are valid.

D.5. Running Real Data Test Cases

In this exercise you will learn how to start ICON from DWD analysis data and how to perform a multi-day forecast with 26 km grid spacing globally and 13 km over Europe. Another practical aim of this exercise is to create raw data for driving a limited area ICON run. Further use of this data will be made in Ex. D.6.1.

The configuration and compilation of the ICON code has to be done on the Cray XC 40 login node `xce`. Job submission to the Cray XC 40 can be performed on either the login node `xce` or the Linux cluster `lce`. **Note, however, that visualization tools (CDO, NCL, ncview) are *only* available on the Linux cluster `lce`!**

Input Data

The exercises in this section require a number of grid files, external parameters and input data. This data is already available in the directory `case_realdata/input`. Their creation process is explained in Ex. D.2.1 and D.2.2 (see p. 192).

On default, ICON expects the input data to be located in the experiment directory (termed `$EXPDIR` in the run scripts). The run script creates symbolic links in the experiment directory, which point to the input files.

Note:

Note that the names originating from the database requests (see Section 2.2.1) differ from the file names above. The input files have to be renamed in this case in order to match the default filename structure expected by the model.

Starting a Global ICON Forecast from DWD Analysis

In this exercise you will learn how to run ICON in real data mode.



EX 5.1

Open the ICON run script `case_realdata/run_ICON_R03B06_dwdini` and prepare the script for running a global **72 hour forecast** on an R3B06 grid with 26 km horizontal grid spacing without nest. The start date is

`2017-01-12T00:00:00` , i.e. January 12, 2017.

Basic settings

- Fill in the missing namelist parameters `ini_datetime_string`, `end_datetime_string`, `ltestcase`, `ldynamics`, `ltransport`, `iforcing`, and `itopo` for real data runs (see Section 5.1.1).
- For better runtime performance, switch on *asynchronous* output by setting the number of dedicated I/O processors (`num_io_procs`) to a value larger than 0 (e. g. 1). See Section 7.3.1 for more details on the asynchronous output module.

Specifying the input data

- The grid file(s) to be used are already specified in the run script (see `dynamics_grid_filename`, `radiation_grid_filename`).
- Note that – funnily enough – the initialized analysis file is provided via the namelist parameter `dwdfg_filename`, see Section 5.1.4.

Due to the appropriate choice of the file names in the experiment directory, ICON is able to locate the analysis data sets automatically without specifying the namelist variable `dwdfg_filename`. However, it is given in the run script for reference, using the keyword nomenclature mentioned in Section 5.1.2.

Have a look at these settings and try to understand how these keywords work.

Which of the following filenames would be accepted by the ICON model?

- `dwdANA_R3B06.grb`
- `dwdANA_R2B6_DOM01.grb`
- `dwdANA_R9B02_DOM02.grb`
- `dwdana.R2B06_DOM01.grb`

- Specify the name of the external parameter file (`extpar_filename`). Instead of specifying the full name, try to make use of the keyword *idom*.

Settings for the Initialized Analysis Data

- Choose the appropriate initialization mode `init_mode` for starting the model from DWD initial analysis data (see Section 5.1.4).
- Surface tiles are activated in the given run-script, as can be deduced from the parameter `ntiles` in the namelist `lnd_nml`. Since the initial data contain aggregated (cell averaged) surface fields, a tile coldstart must be performed by setting `ltile_coldstart` appropriately, i.e. each tile must be initialized with the same cell averaged value (see Section 3.7.8).

Running the model and inspecting the output

- Submit the job to the Cray XC 40.
- After the job has finished, inspect the model output:
 - Take a look at the output files in `case_realdata/output`. You should find two files named `NWP_...`. Use `cdo sinfov` to identify the
 - time interval between two outputs – Answer: h
 - total time interval for which output is written – Answer: h
 - type of vertical output grid (ML, PL, HL) – Answer:
 - one file contains output on the native ICON grid, the other one output on a regular lat/lon grid. Use `cdo sinfov` to identify which file is which.
 - Answer: *native*
 - Answer: *lat/lon*
 - Visualize the 2 m temperature, integrated water vapour, gusts at 10 m, and total precipitation using the `ncview` utility. If you like, you can look into other fields as well.

Time-stepping

This exercise focuses on aspects of the ICON time-stepping scheme, explained in Section 3.4.

- Compute the *dynamics* time step $\Delta\tau$ from the specification of the physics time step Δt (`dttime`) and the number of dynamics substeps `ndyn_substeps`.

– Answer: $\Delta\tau =$ s

- Take a look at Equation (3.15) and calculate an estimate for the maximum dynamics time step which is allowed for the horizontal grid spacing at hand.

– Answer: $\Delta\tau_{max} =$ s

Now compare this to the time step used: Did we make a reasonable choice?



EX 5.2

Parallelization and Run Time Performance

In this exercise you will learn how to specify the details of the parallel execution. We will use the ICON timer module for basic performance measuring.



EX 5.3

Performance assessment using the timer output:

- Open your run script from the previous Exercise D.5.1 and enable the ICON routines for performance logging (timers). To do so, follow the instructions in Section 7.3.4.
- Repeat the model run.
- At the end of the model run, a log file is created. It can be found in your base directory `case_realdata`. Scroll to the end of this file. You should find wall clock timer output comparable to that listed in Section 7.3.4. Try to identify
 - the total run time – Answer: s
 - the time needed by the radiation module – Answer: s



EX 5.4

Changing the number of MPI tasks: In case your computational resources are sufficient, one possibility to speed up your model run is to increase the number of MPI tasks.

- Create a copy of your run script `run_ICON_R03B06_dwdini` named `run_ICON_R03B06_dwdini_fast`. In order to avoid overwriting your old results, replace the output directory name (`EXPDIR`) by `exp02_dwdini_fast`.
- Double the total number of MPI tasks compared to your previous job and re-submit. In more detail, do the following:
 - Your old script (`run_ICON_R03B06_dwdini`) ran the executable in hybrid mode on 25 nodes using 12 MPI tasks/node and 4 OpenMP threads/MPI task with hyper-threading enabled.
 - Your new script (`run_ICON_R03B06_dwdini_fast`) should run the executable in hybrid mode on 50 nodes using 12 MPI tasks/node and 4 OpenMP threads/MPI task with hyper-threading enabled.

You need to adjust both the PBS settings and the `aprun` command. See Section 7.3.4 for additional help.

- Compute the speedup that you gained from doubling the number of MPI tasks.
 - Compare the timer output of the dynamical core, `nh_solve`, and the transport module, `transport`, with the timer output of your previous run.

<code>nh_solve</code>	<code>nh_solve</code>	<code>transport</code>	<code>transport</code>
25 nodes	50 nodes	25 nodes	50 nodes
<input type="text"/> s	<input type="text"/> s	<input type="text"/> s	<input type="text"/> s

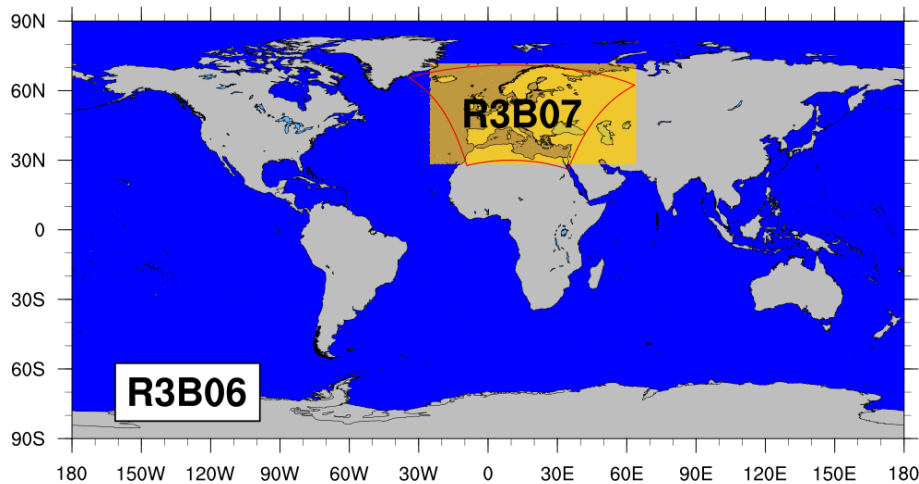


Figure D.1.: Computational grid with a two-way nested region over Europe (yellow shading). The outline of the COSMO-EU domain (formerly used operationally by DWD) is shown in red for comparison.

What do you think is a more sensible measure of the effective cost: `total min` or `total max`?

– Answer:

– Which speedup did you achieve and what would you expect from “theory”?

– Answer: Speedup achieved = $\frac{T_{25\text{nodes}}}{T_{50\text{nodes}}} =$

– Answer: Speedup expected =

Writing Output for Driving a Limited Area Simulation

In this exercise you will learn about some of the output capabilities of ICON. We will set up a new output namelist and generate a data set which enables us to drive a limited area version of ICON as described in Chapter 6.

In order to achieve a somewhat higher spatial resolution at acceptable costs, we will switch on a two-way nested region over Europe with a horizontal grid spacing of 13 km (see Figure D.1):

- Create a copy of your run script `run_ICON_R03B06_dwdini` named `run_ICON_R03B06N7_dwdini`. Alternatively, you may adopt the settings of the reference version `reference/case2/run_ICON_R03B06_dwdini`.
- In order to avoid overwriting your old results, replace the output directory name (`EXPDIR`) by `exp02_dwdini_lamforcing`.



EX 5.5

- Activate the nest, by extending the list of horizontal grids to be used (`dynamics_grid_filename`) and the parent grid IDs `dynamics_parent_grid_id` (`grid_nml`).
- In order to save some computational resources, the nested region should have a reduced model top height and comprise only the lowermost 60 vertical levels of the global domain (instead of 90 levels). Please extend `num_lev` (`run_nml`) accordingly.

Adding a new output namelist:

- The run script contains two commented-out output namelists. One is meant for writing forcing (boundary) data, the other for writing initial data for driving a limited area run. Activate the namelists and fill in the missing parameters. See Section 7.1 for additional details regarding output namelists. Forcing data should be written
 - in GRIB2 format
 - for the EU-nest only
 - 2-hourly from the start until 48 hours forecast time
 - with one output step per file
 - on the native (triangular) grid
 - into the sub-directory “`lam_forcing`” of your output directory `exp02_dwdini_lamforcing`, using the filename prefix “`forcing`”. If the sub-directory does not exist, please create.
 - containing model level output for `U, V, W, THETA_V, DEN, QV, QC, QI, QR, QS, HHL`.
- Submit the job to the Cray XC 40.

Check the correctness of your output files:

- You should find **25 files** in your output directory “`lam_forcing`”. Apply the command `cdo sinfov data-file.grb > data-file.sinfov` to the last file. Compare with the reference output in Table D.1 to see whether your output namelist is correct.

Optional: A Deeper Look into Spin-Up Effects

Depending on the data set used to initialize the model, spin up effects may become visible during the first few hours of a forecast. This is especially true if third party (i.e. non-native) analysis data sets are used. Here we will have a look into the spin up behavior when ICON is started from native vs. non-native analysis. As an example for a popular non-native analysis, we will choose data from the IFS.

Table D.1.: Reference output for Ex. D.5.5. Structure and content of file forcing_DOM02_ML_20170114T000000Z.grb

```

File format : GRIB2
-1 : Institut Source  Ttype    Levels Num   Points Num Dtype : Parameter name
 1 : DWD      unknown instant    60   1   172740  1  P16  : U
 2 : DWD      unknown instant    60   1   172740  1  P16  : V
 3 : DWD      unknown instant    61   2   172740  1  P16  : W
 4 : DWD      unknown instant    60   1   172740  1  P16  : THETA_V
 5 : DWD      unknown instant    60   1   172740  1  P16  : DEN
 6 : DWD      unknown instant    60   1   172740  1  P16  : QV
 7 : DWD      unknown instant    60   1   172740  1  P16  : QC
 8 : DWD      unknown instant    60   1   172740  1  P16  : QI
 9 : DWD      unknown instant    60   1   172740  1  P16  : QR
10 : DWD      unknown instant    60   1   172740  1  P16  : QS
11 : DWD      unknown instant    61   3   172740  1  P16  : HHL

Grid coordinates :
 1 : unstructured      : points=172740
                        grid : number=99 position=1
                        uuid  : 890df70c-a566-3b91-438f-f527deb82760

Vertical coordinates :
 1 : generalized_height : levels=60
                        height : 1.5 to 60.5 by 1
                        bounds  : 1-2 to 60-61 by 1
                        zaxis   : number = 4
                        uuid    : a0919ba7-df84-ce2b-c4c2-3c215c033520
 2 : generalized_height : levels=61
                        height : 1 to 61 by 1
                        zaxis   : number = 4
                        uuid    : a0919ba7-df84-ce2b-c4c2-3c215c033520
 3 : generalized_height : levels=61
                        height : 0.5 to 30.5 by 0.5
                        bounds  : 1-0 to 61-0 by 0.5
                        zaxis   : number = 4
                        uuid    : a0919ba7-df84-ce2b-c4c2-3c215c033520

Time coordinate : 1 step
RefTime = 2017-01-12 00:00:00 Units = minutes Calendar = proleptic_gregorian
YYYY-MM-DD hh:mm:ss YYYY-MM-DD hh:mm:ss YYYY-MM-DD hh:mm:ss YYYY-MM-DD hh:mm:ss
2017-01-14 00:00:00

```

- Run the NCL script `case_realdata/water_budget.ncl`, to get a deeper insight into the model's spin up properties and water budget when started from DWD analysis. The script generates time series of vertically integrated water vapour `tqv` and condensate `tqx` from the model level output in `case_realdata/output/exp02_dwdini`.

Compare the results to Figure D.2 which shows the corresponding results when ICON is started from IFS analysis. Note that Figure D.2 has been produced for a longer time range.

Which analysis data set leads to an increased spin up/down in this particular case?

native analysis (DWD)

non-native analysis (IFS)



EX 5.6

The additional NCL plots will give you some insight into the global atmospheric water budget

$$\frac{dQ_t}{dt} = P - E + R,$$

where Q_t is the vertically integrated atmospheric water content and dQ_t/dt is the rate of change over time. P is the amount of total precipitation, E is the total evaporation, and R is a residuum. Is the budget closed (i.e. is $R = 0$ in your model run)?

Optional: Checking for Bit-Reproducibility



EX 5.7

- Compare the model level output of `run_ICON_R03B06_dwdini` with the output that was produced by your modified script `run_ICON_R03B06_dwdini_fast`. You can use `cdo infov data-file.nc` for getting information about the contents of your output file. You should dump this information into text files,

```
cdo infov data-file.nc > data-file.infov
```

so that you can compare them later on. Due to the limited number of digits printed by CDO, this is no check for bit-reproducibility in a strict mathematical sense, however experience showed that `cdo infov` is very reliable in revealing reproducibility issues.

- Now compare the model level output produced by your modified script `run_ICON_R03B06N7_dwdini` with the output produced by `run_ICON_R03B06_dwdini`. Is the result different from the previous comparison? Do you have an explanation for this?

Hint: For a convenient comparison of ASCII files you may use the `tkdiff` utility.

D.6. Running ICON-LAM

The exercises in this section require a number of grid files, external parameters and input data. In particular, initial data and driving boundary data are required. Both were produced in the real data exercise, Ex. D.5.5. The preprocessing of this data is explained in Ex. D.2.4 (see p. 194).

Running ICON in Limited Area Mode

In this exercise we will run ICON in limited area mode. The model will be driven by initial and boundary data which have been produced in Ex. D.2.4.

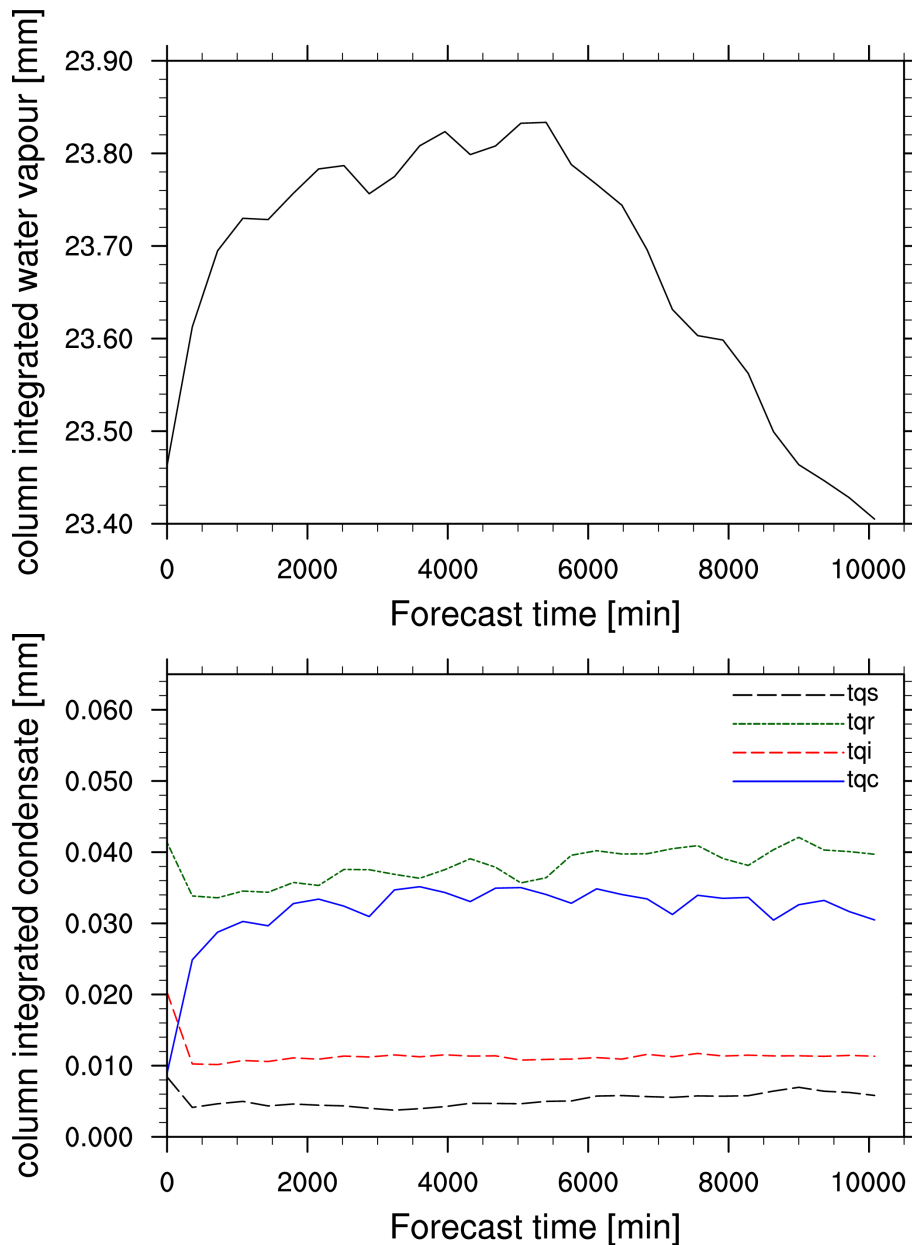


Figure D.2.: Time series of area averaged column integrated specific moisture $\langle tqv \rangle$ (top) and condensate classes $\langle tqx \rangle$ (bottom) for a 7-day forecast **started from IFS analysis** fields. Start date was 2017-01-12T00:00:00. A spin up in $\langle tqv \rangle$ and initial adjustments in $\langle tqx \rangle$ is clearly visible.

Open the run script `case_lam/run_ICON_R3B08_lam` and prepare it for running a **48 hour forecast** on a **limited area grid over Germany** (see Figure D.3). As for the global run, the start date is

2017-01-12T00:00:00 , i.e. January 12, 2017.



EX 6.1

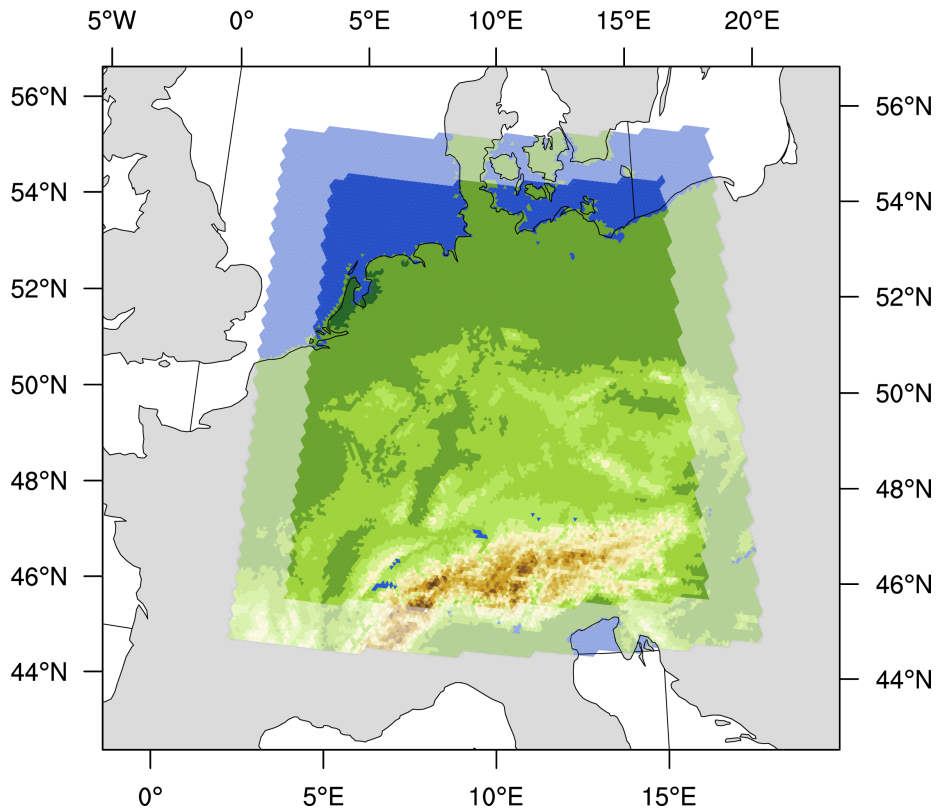


Figure D.3.: Illustration of the local grid used in Exercises D.6.1–D.8.1. The horizontal grid spacing is ≈ 6.5 km (which corresponds to R3B8 in ICON nomenclature). The boundary region is highlighted, where nudging towards the driving data is performed. The driving data for this test case has been created in Ex. D.2.4.

The run script is geared up insofar as all soft-links which specify the model binary, grids, initial and boundary data are already set. Your main task will be to set up the ICON namelists for a limited area run.

Basic settings:

- Set the correct start and end date:
`ini_datetime_string`, `end_datetime_string`.
- Switch on the limited area mode by setting `l_limited_area` and `init_mode` (see Section 6.3).

Initial data:

- Specify the initial data file via `dwdfg_filename`. Since we do not make use of additional analysis information from a second data file, remember to set `lread_ana` accordingly (see Section 6.3).

Boundary data:

- Specify the lateral boundary grid and data via `latbc_boundary_grid`, `latbc_path` and `latbc_filename`. For the latter you will have to make use of the keywords `<y>`, `<m>` `<d>` and `<h>` (see Section 6.4.1).
- What is the time interval between two consecutive boundary data files?
The command `cdo sinfov` may be helpful.

– Answer: s

Set the namelist parameter `dtime_latbc` accordingly.

- Inspect the number of vertical levels in a boundary data file. Is it different from the number of vertical levels that is used by the model itself?

– Answer:

Running the model and inspecting the output

- Extend the lat-lon output namelist by the 2 m temperature, surface pressure, mean sea level pressure, and 10 m gusts. See Appendix C for variable names and description.
- Submit the job to the Cray XC 40.
- After the job has finished, inspect the model output. You should find multiple files in the output directory `case_lam/output/exp03_R3B08_dwdlam` which contain hourly output on model levels remapped to a regular lat-lon grid.
 - Take a look at the output fields by using `ncview`. How would you characterize the overall weather situation for that time period?

southfoehn over the alpine region

strong frontal system passing over Germany

weak frontal system passing over Germany

anticyclonic situation with very low winds

Temporal Resolution of the Boundary Data

By playing around with the temporal resolution of the boundary data you will get some idea how this might affect your simulation results.

Create a copy of your run script `run_ICON_R3B08_lam` and name it `run_ICON_R3B08_lam_lowres`. Replace the output directory name (`EXPDIR`) by `exp03_R3B08_dwdlam_lowres` in order to avoid overwriting your results.

- Halve the time resolution of your forcing (boundary) data, see Section 6.3 for the namelist parameter. Write down your chosen value:

**EX 6.2**

– Answer: s

- Submit the job to the Cray XC 40.
- Compare the results with your previous run. Does the time frequency with which the boundary data are updated have a significant impact on the results? You can visualize cross sections with the NCL script `case_lam/plot_cross_section.ncl` and/or make use of `ncview`.

D.8. Programming ICON



EX 8.1

In this exercise we will implement two new diagnostic fields (2D variables):

RHI_MAX: Relative humidity over ice (hourly maximum in the column, [%])

QI_MAX: Maximum cloud ice content (hourly max. in the column, $\left[\frac{\text{kg}}{\text{kg}}\right]$)

This requires to modify the ICON code, where details are given in Section 8.4.

Remember to create a backup copy of all source files which you modify in this exercise! Subsequent exercises will be based upon the unmodified sources!

Step-by-step checklist:

- Open the module `mo_nonhydro_types` (sub-directory `src/atm_dyn_iconam`).

Insert two new 2D variable pointers in `TYPE(t_nh_diag): RHI_MAX, QI_MAX`

Got it? Did it!

- Open the module `mo_nonhydro_state` (sub-directory `src/atm_dyn_iconam`).

At the end of the subroutine `new_nh_state_diag_list`: initialize meta-data variables for GRIB2 and NetCDF with `discipline = parameterCategory = parameterNumber = 255` and place the `add_var` calls for the two new fields.

Got it? Did it!

Do not forget to specify the hourly reset of the quantities.

- Open the module `mo_nh_stepping` (sub-directory `src/atm_dyn_iconam`).

Create an (empty) subroutine `calculate_diagnostics` and place a call to this subroutine immediately before the call to the output routine.

Got it? Did it!

Fill your subroutine with a 2D loop over all grid points, calculate the two new quantities.

Two hints:

Got it? Did it!

- The module `mo_util_phys` (sub-directory `src/atm_phy_nwp`) may offer some help with respect to `RHI_MAX`.
- Exner pressure field: values for domain “jg” are accessed via `p_nh_state%prog(nnow(jg))%exner(:, :, :)`.
- 3D tracer field QI (same for QV): values for domain “jg” are accessed via `p_nh_state%prog(nnow_rcf(jg))%tracer_ptr(iqi)%p_3d(:, :, :)`. The constants `iqv`, `iqi` can be found in `mo_run_config`.
- Open the namelist of the previous test case [D.6.1](#) and insert the new fields in the output specification.

Compile and run the model. Visualize the results (`ncview`).

D.10. Running ICON-ART

In order to start with the exercises, you have to copy and unpack the ART code inside your ICON source directory as described in section 10.3.

You will find the ART code at:

`/e/uwork/trng024/packages/art.tar.gz.`

Folders with the input data for all ART exercises can be found at:

`/e/uwork/trng024/packages/ART-INPUT.`

After you have copied the source code, you have to install ICON-ART. For this purpose, proceed as described in Section 10.3. After a successful compilation of ICON-ART, you can start with the experiments, that were prepared for this purpose:

Point Sources in ICON-ART LAM



EX 10.1

In this exercise, you will learn to add your own tracers with emissions from point sources. You are free to choose the tracer(s) to transport and the location of the point source(s).

In order to perform the simulation, you have to do the following steps:

- Inside `$ARTDIR/runctrl.examples/run_scripts` you will find the run script for this test case called `exp.art.trng18.case1.pntSrc`
- inside the ART-INPUT folder you will find a folder called `CASE1-PNTSRC` containing all input data required for this test case
- Edit the run script according to the namelist parameters you find in Section 10.4. You will find a ? at all places where you have to edit something.
- Create the XML files that are needed to define tracers (see section 10.4.3) and point sources (see section 10.4.5).
- Submit the job.
- Visualize your results using `ncview`.

Very Short-lived Substances



EX 10.2

Biogenic emitted Very Short-lived Substances (VSLs) have a short chemical lifetime in the atmosphere compared to tropospheric transport timescales. As the ocean is the main source of the most prominent VSLs, bromoform (CHBr_3) and dibromomethane (CH_2Br_2), this leads to large concentration gradients in the troposphere. The tropospheric depletion of CHBr_3 is mainly due to photolysis, whereas for CH_2Br_2 the loss is dominated by oxidation by the hydroxyl radical

(OH) both contributing to the atmospheric inorganic bromine (Bry) budget. Once released, active bromine radicals play a significant role in tropospheric as well as stratospheric chemistry as they are in particular involved in ozone destroying catalytic cycles. Although the bromine budget in the stratosphere is dominated by the release of Bry from long-lived source gases (e.g. halons) which is relatively well understood the contribution of biogenic VSLS to stratospheric bromine is still uncertain.

In this exercise the fast upward transport of both VSLS from the lower boundary into the upper troposphere / lower stratosphere (UTLS) due to the super-typhoon Haiyan will be simulated.

In order to perform the simulation, you have to do the following steps:

- Inside the `ART-INPUT` folder you will find a folder called `CASE2-VSLS` containing all input data required for this test case.
- Inside `$ARTDIR/runctrl_examples/run_scripts` you will find the run script for this test case called `exp.art.trng18.case2.vsls`. Edit the run script according to the namelist parameters you find in Section 10.4. You will find a ? at all places where you have to edit something.
- Create the XML file that is needed to define tracers (see section 10.4.3). For this configuration, you will have to add tracers for `CHBr3` and `CH2Br2`.
- Don't forget to modify your output namelist accordingly.
- Submit the job.
- Visualize your results using `ncview`.

Volcano Eruption

The eruption of a volcano can have a significant radiative impact as well as an impact on air traffic. In this example, you will simulate the dispersion of volcanic ash particles. Within ICON-ART, you have the choice of treating volcanic ash as monodisperse tracers or with prognostic mass and number (i.e. 2-moment aerosol). In this example, we will make use of the 2-moment description. In order to perform the simulation, you have to do the following steps:

- Inside the `ART-INPUT` folder you will find a folder called `CASE3-VOLC` containing all input data required for this test case.
- Inside `$ARTDIR/runctrl_examples/run_scripts` you will find the run script for this test case called `exp.art.trng18.case3.volc`. Edit the run script according to the namelist parameters you find in Section 10.4. You will find a ? at all places where you have to edit something. For `cart_volcano_file`, you can use the file `ART-INPUT/CASE2-VOLC/volcano_list_Eyjafjoell.txt`.



EX 10.3

- Create the XML file that is needed to define tracers (see section 10.4.3). You also have to create an XML file specifying the aerosol modes in the simulation (see section 10.4.4). For the 2-moment description of volcanic ash, you will need the modes `asha`, `ashb` and `ashc` and the according tracers.
- You can use the namelist switch `lart_diag_out` to obtain diagnostical properties like aerosol optical depth and median diameters.
- Don't forget to modify your output namelist accordingly.
- Submit the job.
- Visualize your results using `ncview`.

Sea Salt Aerosol



EX 10.4

Sea salt aerosol is one of the main contributors to natural atmospheric aerosol. With its high hygroscopicity it is a very efficient cloud condensation nuclei (CCN). Within ICON-ART, sea salt aerosol is described as a log-normally distributed aerosol in three modes with prognostic mass mixing ratios and prognostic number mixing ratios. This simulation includes also a nested domain covering Europe and North Africa with a higher spatial resolution. In order to perform the simulation, you have to do the following steps:

- Inside the `ART-INPUT` folder you will find a folder called `CASE4-SEAS` containing all input data required for this test case.
- Inside `$ARTDIR/runctrl_examples/run_scripts` you will find the run script for this test case called `exp.art.trng18.case4.seas`. Edit the run script according to the namelist parameters you find in section 10.4. You will find a ? at all places where you have to edit something.
- Create the XML file that is needed to define tracers (see section 10.4.3). You also have to create an XML file specifying the aerosol modes in the simulation (see section 10.4.4). For the 2-moment description of sea salt, you will need the modes `seasa`, `seasb` and `seasc` and the according tracers.
- Don't forget to modify your output namelist accordingly.
- Submit the job.
- Visualize your results using `ncview`.

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).
- 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.
- Barker, H. W., G. L. Stephens, P. T. Partain, et al., 2003: Assessing 1D atmospheric solar radiative transfer models: Interpretation and handling of unresolved clouds. *J. Clim.*, **16**(16), 2676–2699.
- Bechtold, P., 2017: Atmospheric moist convection. In *Meteorological Training Course Lecture Series*, ECMWF, 1–78.
- 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.
- 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.
- 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**, 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., M. A. Giorgetta, R. Brokopf, M. Esch, S. Fiedler, C. Hohenegger, L. Kornbluh, T. Mauritsen, C. Nam, A. K. Naumann, K. Peters, S. Rast, E. Roeckner, M. Sakradzija, H. Schmidt, J. Vial, R. Vogel, and B. Stevens, 2018: ICON-A, the atmospheric component of the ICON Earth System Model - Part 2. *J. Adv. Model Earth Sy.*, *under review*.
- Davies, H., 1976: A lateral boundary formulation for multi-level prediction models. *Q. J. R. Meteorol. Soc.*, **102**, 405–418.
- Davies, H., 1983: Limitations of some common lateral boundary schemes used in regional NWP models. *Mon. Weather Rev.*, **111**, 1002–1012.

- 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.
- Doms, G., J. Förstner, E. Heise, H.-J. Herzog, D. Mironov, M. Raschendorfer, T. Reinhardt, B. Ritter, R. Schrodin, J.-P. Schulz, and G. Vogel, 2011: *A Description of the Nonhydrostatic Regional COSMO Model. Part II: Physical Parameterization*. Consortium for Small-Scale Modelling.
- 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*. IFS Documentation. ECMWF, 75–95.
- 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.
- 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., R. Brokopf, T. Crueger, M. Esch, S. Fiedler, J. Helmert, C. Hohenegger, L. Kornbluh, M. Kohler, E. Manzini, T. Mauritsen, C. Nam, S. Rast, C. Reick, D. Reinert, M. Sakradzija, H. Schmidt, R. Schnur, L. Silvers, H. Wan, G. Zangl, and B. Stevens, 2018: ICON-A, the atmospheric component of the ICON Earth System Model - Part I. *J. Adv. Model Earth Sy.*, *under review*.
- 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.
- Heinze, R., A. Dipankar, C. C. Henken, C. Moseley, O. Sourdeval, S. Trömel, X. Xie, P. Adamidis, F. Ament, H. Baars, et al., 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. In *COSMO Technical Reports No. 9*, Consortium for Small-Scale Modelling, 19pp.
- Hesselberg, A., 1925: Die Gesetze der ausgeglichenen atmosphärischen Bewegungen. *Beitr. Phys. Atmos.*, **12**, 141160.
- 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.
- 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.

- 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.
- Köhler, M., M. Ahlgrimm, and A. Beljaars, 2011: Unified treatment of dry convective and stratocumulus-topped boundary layers in the ECMWF model. **137**, 43–57.
- 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.
- 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**, 15640.
- Lauritzen, P. H., C. Erath, and R. Mittal, 2011: 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, 2011: *Numerical Techniques for Global Atmospheric Models*. Springer, 556, first edition.
- 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.
- 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.
- Lin, S.-J. and R. B. Rood, 1996: Multidimensional flux-form semi-lagrangian transport schemes. *Mon. Weather Rev.*, **124**(9), 2046–2070.
- 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.
- Lundgren, K., B. Vogel, H. Vogel, and C. Kottmeier, 2013: Direct radiative effects of sea salt for the mediterranean region under conditions of low to moderate wind speeds. *J. Geophys. Res.*, **118**(4), 1906–1923.
- Mellor, G. L. and T. Yamada, 1982: Development of a turbulence closure model for geophysical fluid problems. *Rev. Geophys.*, **20**(4), 851–875.

- 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, 2012a: Parameterisation of sea and lake ice in numerical weather prediction models of the German weather service. *Tellus A*, **64**(0).
- Mironov, D., B. Ritter, J.-P. Schulz, M. Buchhold, M. Lange, and E. Machulskaya, 2012b: Parameterization of sea and lake ice in numerical weather prediction models of the German Weather Service. *Tellus A*, **64**, 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.
- 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), 16663–16682.
- Narcovich, F. J. and J. D. Ward, 1994: Generalized Hermite interpolation via matrix-valued conditionally positive definite functions. *Math. Comp.*, **63**, 661–687.
- Neggers, R. A. J., M. Köhler, and A. C. M. Beljaars, 2009: A dual mass flux framework for boundary layer convection. Part I: Transport. *J. Atmos. Sci.*, **66**(6), 1465–1487.
- 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.
- Pincus, R. and B. Stevens, 2013: Paths to accuracy for radiation parameterizations in atmospheric models. *J. Adv. Model Earth Sy.*, **5**(2), 225–233.
- 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.
- Prill, F., 2014: *DWD ICON Tools Documentation*. Deutscher Wetterdienst (DWD). dwd.icon.tools/doc/icontools.doc.pdf.
- Raschendorfer, M., 2001: The new turbulence parameterization of LM. In *COSMO News Letter No. 1*, Consortium for Small-Scale Modelling, 89–97.
- Rast, S., 2017: Using and programming ICON – a first introduction. Course at Hamburg University 2017.
- Rieger, D., M. Bangert, I. Bischoff-Gauss, J. Förstner, K. Lundgren, D. Reinert, J. Schröter, H. Vogel, G. Zängl, R. Ruhnke, and B. Vogel, 2015: ICONART 1.0 a new online-coupled model system from the global to regional scale. *Geosci. Model Dev.*, **8**(6), 1659–1676.

- Rotta, J., 1951a: Statistische theorie nichthomogener turbulenz. *J. Z. Physik*, **129**(6), 547–572.
- Rotta, J., 1951b: Statistische theorie nichthomogener turbulenz. *J. Z. Physik*, **131**(1), 51–77.
- Sadourny, R., A. Arakawa, and Y. Mintz, 1968: Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere. *Mon. Weather Rev.*, **96**(6), 351–356.
- 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, 2002: A new multi-layer soil-model. In *COSMO News Letter No. 2*, Consortium for Small-Scale Modelling, 149–151.
- Schröter, J., D. Rieger, C. Stassen, H. Vogel, M. Weimer, S. Werchner, J. Förstner, F. Prill, D. Reinert, G. Zängl, M. Giorgetta, R. Ruhnke, B. Vogel, and P. Braesicke, 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. Discuss.*
- 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.
- Seifert, A., 2008: A revised cloud microphysical parameterization for COSMO-LME. In *COSMO News Letter No. 7, Proceedings from the 8th COSMO General Meeting in Bucharest, 2006*, Consortium for Small-Scale Modelling, 25–28.
- 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.
- Siebesma, A. P. and J. W. M. Cuijpers, 1995: Evaluation of parametric assumptions for shallow cumulus convection. *J. Atmos. Sci.*, **52**, 650–666.
- 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.
- 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.
- Smagorinsky, J., 1963: General Circulation Experiments with the Primitive Equations. *Mon. Weather Rev.*, **91**, 99.
- 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.

- 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.
- Sommeria, G. and J. W. Deardorff, 1977: Subgrid-Scale Condensation in Models of Non-precipitating Clouds. *J. Atmos. Sci.*, **34**(2), 344–355.
- Strang, G., 1968: On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, **5**(3), 506–517.
- Tiedtke, M., 1989: A comprehensive mass flux scheme for cumulus parameterization in large-scale models. *Mon. Weather Rev.*, **117**(8), 1779–1800.
- 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.
- Wacker, U. and F. Herbert, 2003: Continuity equations as expressions for local balances of masses in cloudy air. *Tellus A*, **55**, 247–254.
- Wan, H., M. A. Giorgetta, G. Zängl, M. Restelli, D. Majewski, L. Bonaventura, K. Fröhlich, D. Reinert, P. Rípodas, L. Kornbluh, and J. Förstner, 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.
- 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., D. Reinert, P. Rípodas, 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.
- Zdunkowski, W. and A. Bott, 2003: *Dynamics of the atmosphere: A course in theoretical meteorology*. Cambridge University Press, 719, first edition.
- 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.

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.pdf](#)

for a complete list of available namelist parameters for the ICON model. ICON-ART specific namelists are described in the ICON-ART documentation.

initicon_nml (Namelist), 104

ana_varnames_map_file, 104

art_nml (Namelist), 160

bdy_indexing_depth, 22, 23

cart_volcano_file, 213

cldopt_filename, 42

damp_height, 111

diffusion_nml (Namelist), 97

dom, 118, 118

dt_checkpoint, 120, 121

dt_conv, 57, 57

dt_gwd, 57

dt_iau, 106, 106

dt_rad, 57, 57

dt_restart, 121, 121

dt_shift, 106, 106, 108

dt_sso, 57

dtime, 55, 100, 102, 201

dtime_latbc, 114, 209

dwdana_filename, 104, 105–107

dwdfg_filename, 104, 105, 107, 112, 113, 200, 208

dynamics_grid_filename, 99, 99, 102, 103, 197, 198, 200, 204

dynamics_nml (Namelist), 97

dynamics_parent_grid_id, 99, 99, 102, 197, 198, 204

end_datetime_string, 101, 101, 102, 200, 208

end_time, 108, 108

experimentStartDate, 102, 108

experimentStopDate, 102

extpar_filename, 103, 103, 200

extpar_nml (Namelist), 98, 103

fbk_relax_timescale, 70

filetype, 118, 118

flat_height, 55

frlake_thrhd, 84

grid_nml (Namelist), 53, 70, 93, 99, 102, 103, 108, 109, 197, 204

gridgen_nml (Namelist), 22, 23

gridref_nml (Namelist), 70

h_levels, 120

hbot_qv_substep, 64

hbot_qvsubstep, 92, 92, 93

hl_varlist, 120, 120

htop_moist_proc, 65, 92, 92, 93

i_levels, 120

iart_seasalt, 167

iart_volcano, 167

iforcing, 97, 97, 102, 196, 200

ifs2icon_filename, 107, 112

ihadv_tracer, 64, 92

il_varlist, 120, 120

in_filename, 194

in_grid_filename, 39, 194

in_type, 39

- ini_datetime_string, **101**, 101, 102, 104, 106–108, 200, 208
- init_mode, 35, **103**, 103–105, 107, **112**, 112, 113, 201, 208
- initicon_nml (Namelist), 85, 89, 103–108, 112, 113
- input_field_nml (Namelist), 35, 36, 39
- interpol_nml (Namelist), 64, 67, 111
- inwp_cldcover, **76**
- inwp_convection, **76**, **77**
- inwp_gscp, **75**, **75**, **76**, **76**, **77**
- inwp_gwd, **76**
- inwp_radiation, **76**
- inwp_sso, **76**
- inwp_surface, **76**
- inwp_turb, **76**, **80**
- io_nml (Namelist), 118, 120, 121
- itopo, **98**, **98**, 103, 196, 200
- itype_hlimit, **64**
- itype_latbc, **110**, **114**
- itype_vlimit, **62**, **64**
- ivadv_tracer, **64**
- ivctype, **53**, **55**

- l_limited_area, **109**, 208
- lart, 160
- lart_diag_out, 167, 214
- latbc_boundary_grid, **115**, 209
- latbc_filename, 114, **115**, 115, 209
- latbc_path, 114, **115**, 115, 209
- latbc_varnames_map_file, **115**
- latm_above_top, **111**
- ldynamics, **97**, 102, 196, 200
- les_nml (Namelist), 82
- lfeedback, 70
- limarea_nml (Namelist), 110, 114, 115
- llake, **76**, **84**, 85
- lnd_nml (Namelist), 42, 84, 86, 88, 201
- lread_ana, **112**, 112, **113**, 113, 208
- lredgrid_phys, **93**
- lrestart, **121**
- lrtm_filename, **42**
- lseaiice, **76**, **86**
- lshallowconv_only, **77**
- lsnowtile, 88
- lsq_high_ord, **64**
- lstrang, **58**
- ltestcase, **96**, 102, 196, 200
- ltile_coldstart, 89, 201
- ltile_init, 89
- ltimer, **126**
- ltransport, **63**, **97**, 102, 198, 200
- lvert_nest, **52**, 74
- lwrite_parent, **23**, 94

- m_levels, **118**
- master_nml (Namelist), 105, 121
- master_time_control_nml (Namelist), 102
- min_lay_thckn, **55**, 55
- ml_varlist, **118**, 118, 120
- model_base_dir, 105
- modelTimeStep, 102

- ncstorage_file, **41**
- ndyn_substeps, **55**, 56, 57, 201
- nh_test_name, **98**, 98, 196
- nh_testcase_nml (Namelist), 95, 98, 198
- nonhydrostatic_nml (Namelist), 53, 55, 65, 92, 97, 111
- nproma, **129**, 130
- nsteps, **100**, 101
- ntiles, **42**, **88**, 88, 89, 201
- ntracer, 63
- nudge_efold_width, 111
- nudge_max_coeff, 111
- nudge_zone_width, **67**, 111
- num_io_procs, 119, **123**, 200
- num_lev, **52**, 55, 74, 197, 204
- num_prefetch_proc, **115**, 115
- num_restart_proc, 122
- num_restart_procs, 121, **123**
- nwp_phy_nml (Namelist), 42, 57, 75–77, 80, 97, 111

- out_grid_filename, **39**
- out_type, **39**
- output, **117**
- output_bounds, **118**
- output_filename, **117**
- output_nml (Namelist), 100, 117–120, 188, 198
- output_nml_dict, **118**

- p_levels, **120**
- parallel_nml (Namelist), 115, 119, 121–123, 129
- pe_placement_ml, **119**

pl_varlist, **120**, 120

radiation_grid_filename, 93, **103**, 200

reg_lat_def, **120**

reg_lon_def, **120**

remap, 118, **120**

remap_nml (Namelist), 41

restart_filename, **121**, 121

restart_write_mode, **121**, 121

run_nml (Namelist), 52, 63, 74, 96, 97, 100–
102, 117, 121, 126, 160, 179, 197,
204

sleve_nml (Namelist), 55

smag_constant, **82**

start_time, **108**, 108

steps_per_file, **118**

stream_partitions_ml, **119**, 119

time_nml (Namelist), 101, 102, 121

timers_level, **126**

top_height, **55**

tracer_inidist_list, **198**, 198

tracer_names, **198**, 198

transport_nml (Namelist), 58, 62–64, 92, 97,
102, 198

turb_prandtl, **82**

turbdiff_nml (Namelist), 81

use_lakeiceana, **85**

var_in_mask, 36

vertical_grid_filename, **53**