

ICON Model Tutorial

February/March 2017



Working with the ICON Model

Practical Exercises
for NWP Mode and ICON-ART

For NWP Mode:

D. Reinert, F. Prill, G. Zängl,
A. Fernandez del Rio, R. Potthast

For ICON-ART:

D. Rieger, J. Schröter, J. Förstner, C. Walter,
R. Ruhnke, B. Vogel

Acknowledgments

This tutorial is based on the extensive COSMO Model Tutorial written by U. Schättler, U. Blahak, M. Baldauf et al. (2016).

Chapter 9 was provided by R. Potthast and A. Fernandez del Rio, DWD Data Assimilation division.

Section 5.2.4 has been contributed by A. Seifert, DWD Physical Processes division.

Chapter 8 was provided by the Institute of Meteorology and Climate Research at the Karlsruhe Institute of Technology (KIT).

Bibliography

U. Schättler, U. Blahak, M. Baldauf, A. Smalla, C. Barbu, R. Dumitrache, B. Maco et al. *COSMO-Model Tutorial. Working with the COSMO-Model. Practical Exercises*, 2016.

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	2
1. Installation of the ICON Model Package	3
1.1. The ICON Model Package	3
1.1.1. Directory Layout	3
1.1.2. Libraries Needed for Data Input and Output	5
1.2. Configuring and Compiling the Model Code	7
1.2.1. Computer Platforms	7
1.2.2. Configuring and Compiling	8
1.3. The DWD ICON Tools	9
1.3.1. General Overview	9
1.3.2. Configuring and Compiling the DWD ICON Tools	11
1.4. Exercise	12
2. Necessary Input Data	13
2.1. Horizontal Grids	13
2.1.1. ICON Grid Files	13
2.1.2. Grid Generator	15
2.1.3. Download of Predefined Grids	17
2.1.4. Which Grid File is Related to My Simulation Data?	18
2.2. External Data Files	18
2.2.1. ExtPar Products	18
2.2.2. Web-based Generation of Grids and External Parameters	19
2.3. Initial and Boundary Conditions	21
2.3.1. DWD Analysis Data	21
2.3.2. ECMWF IFS Initial Data	26
2.3.3. ICON-to-ICON Initial Data	28
2.3.4. Boundary Data From a Driving Model (ICON-LAM Mode)	28
2.4. Exercises	32
3. Running Idealized Test Cases	37
3.1. Namelist Input for the ICON Model	37
3.2. Vertical coordinates	38
3.2.1. Terrain-following Hybrid Gal-Chen Coordinate	38
3.2.2. SLEVE Coordinate	39
3.3. Jablonowski-Williamson Baroclinic Wave Test	40
3.3.1. Main Switches for the Idealized Test Case	42

3.3.2. One-Way Nesting and Two-Way-Nesting	43
3.3.3. Specifying the Computational Domain(s)	44
3.3.4. Basic Control Variables	45
3.4. Exercises	46
4. Real Data Test Cases	49
4.1. ICON Time-Stepping	49
4.2. Model Initialization	51
4.2.1. Basic Settings for Running Real Data Runs	51
4.2.2. Starting from DWD Analysis	53
4.2.3. Starting from DWD Analysis <i>with IAU</i>	54
4.2.4. Starting from IFS Analysis	55
4.3. Settings for the Model Output	56
4.4. Parallelization Aspects	58
4.4.1. Settings for Parallel Execution	58
4.4.2. Bit-Reproducibility	61
4.4.3. Basic Performance Measurement	61
4.5. Exercises	62
5. Limited Area Mode	71
5.1. Running ICON-LAM	71
5.1.1. Limited Area Mode vs. Nested Setups	71
5.1.2. Nudging in the Boundary Zone	72
5.1.3. Initialization Mode	72
5.1.4. Naming Scheme for Lateral Boundary Data	73
5.1.5. Pre-Fetching of Boundary Data (Mandatory)	74
5.2. ICON Physics in a Nutshell	74
5.2.1. Overview	74
5.2.2. Details of ICON's Physics-Dynamics Coupling	74
5.2.3. Isobaric vs. Isochoric Coupling Strategies	77
5.2.4. Cloud Microphysics	77
5.3. Implementing Own Diagnostics	78
5.4. Exercises	81
6. ICON NWP Mode: Further Features	85
6.1. Reduced Model Top for Moist Physics	85
6.2. Reduced Radiation Grid	86
6.3. Internal Post-Processing	87
6.4. Checkpointing and Restart	88
6.5. Exercises	89
7. Post-Processing and Visualization	99
7.1. Retrieving Data Set Information	99
7.1.1. ncdump	99
7.1.2. CDO – Climate Data Operators	100
7.2. Plotting Data Sets on Regular Grids with ncview	101
7.3. Plotting Data Sets on the Triangular Grid	101
7.3.1. NCL – NCAR Command Language	101

7.3.2. NCL Step-by-step Tutorial	103
7.3.3. GMT – Generic Mapping Tools	108
8. Running ICON-ART	111
8.1. General Remarks	111
8.2. ART Directory Structure	111
8.3. Installation	113
8.4. Configuration of an ART Job	114
8.4.1. Recommended ICON Namelist Settings	114
8.4.2. ART Namelists	114
8.4.3. Tracer Definition with XML Files	118
8.4.4. Modes Definition with XML Files	119
8.4.5. Point Source Module: pntSrc	120
8.4.6. Volcanic Ash Control	121
8.5. Output	122
8.6. Exercises	123
9. ICON’s Data Assimilation System	127
9.1. Data Assimilation	127
9.1.1. Variational Data Assimilation	127
9.1.2. Ensemble Kalman Filter	129
9.1.3. Hybrid Data Assimilation	129
9.1.4. Surface Analysis	130
9.2. Assimilation Cycle at DWD	130
Appendix A. The Computer System at DWD	133
Appendix B. Troubleshooting for the ICON Model	135
Appendix C. Table of ICON Output Variables	139
Bibliography	147
Index of Namelist Parameters	151

0. Preface

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

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

- 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, and
- 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 greatly facilitates the assimilation of satellite data) whereas the nested domains extend only into the lower stratosphere in order to save computing time.

The ICON modelling framework became operational in DWD's forecast system in January 2015. During the first six months only global simulations were executed with a horizontal resolution of 13 km and 90 vertical levels. Starting from July 21st, 2015, model simulations have been complemented by a nesting region over Europe.

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 – 4.
- New users who are interested in the *regional* model should read Chapter 5 in addition.

- More advanced topics are covered by Chapters 6 – 9.

To some extent this document can also be used as a reference manual. To this end, we refer to the index of namelist parameters on page 151.

Each of the chapters concludes with a number of exercises revisiting the topics from that part. Paragraphs describing common pitfalls and containing details for advanced users are marked by the symbol .

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.

For **data requests** with respect to DWD operational data products please contact datenservice@dwd.de. Access to the **grid generator web service** (see Section 2.2.2) requires a user account. To this end, please contact Klima.Vertrieb@dwd.de.

0.3. Further Documentation

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

The pre- and post-processing tools of the DWD ICON Tools collection are described in more detail in the DWD ICON Tools manual [Prill \(2014\)](#).

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.

¹An individual licensing procedure has not yet been released by February 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 development branch (state *February 2017*). 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).
- **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.

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.

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

Subdirectory `build`

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

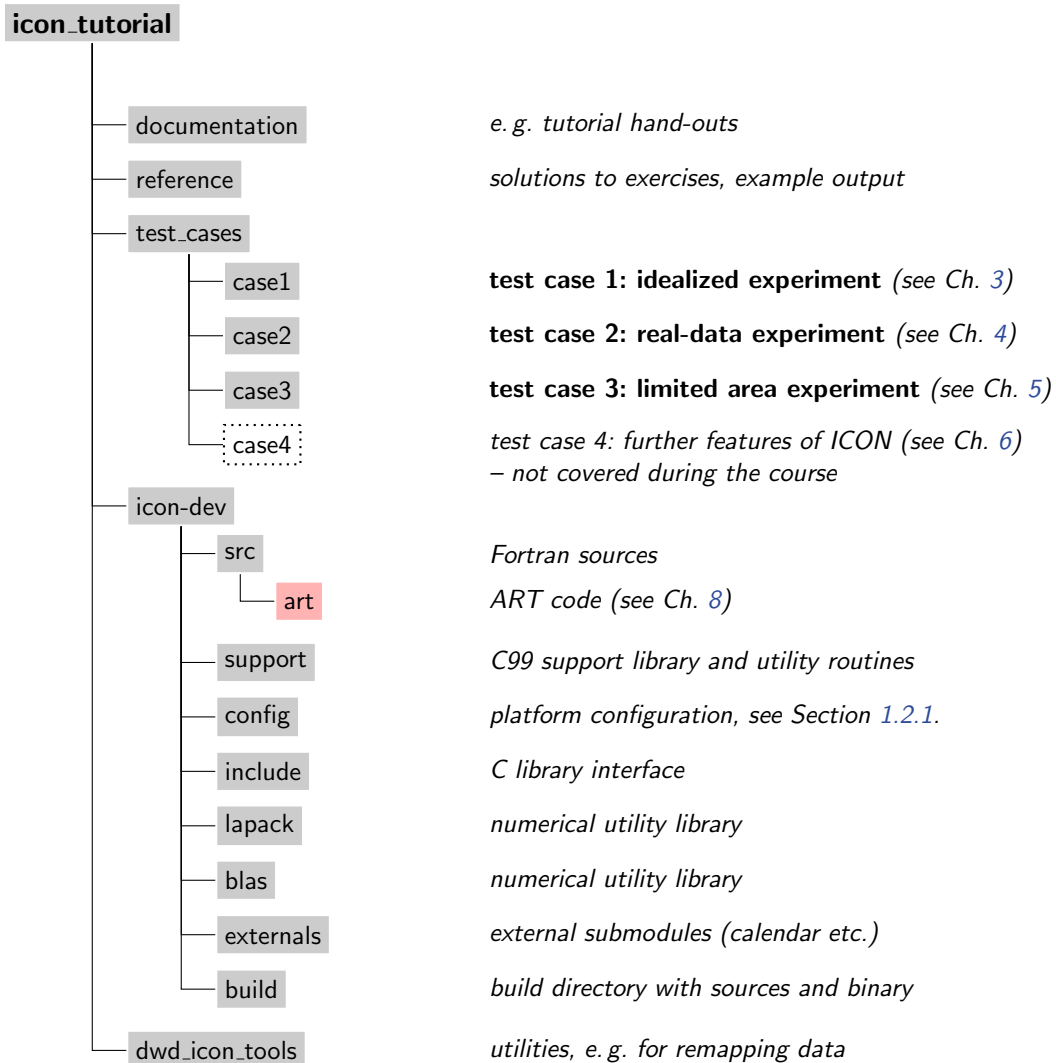


Figure 1.1.: Directory structure of the ICON model and of the directories containing the test case data under the root tree.

Subdirectory `config`

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

Subdirectory `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 subdirectories:

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

The configuration of ICON run-time settings is performed 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 implemented in 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.zmaw.de/projects/cdi>.

The ECMWF GRIB-API – `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 setting meta-data, the *GRIB-API* uses the so-called key/value approach. Indirect use of this GRIB-API library in the ICON model is implemented through the CDI.

In addition to the interface routines, 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 we refer to the ECMWF web page:

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



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

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

- The GRIB-API 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 library can be built with `make` and then `make install`.

GRIB Definition Files

An installation of the GRIB-API 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. For example, these definition files contain information about the variable short name and the corresponding GRIB code triplet.

The short name, e.g., “`t`” for temperature, is not stored in data files, in contrast to the corresponding GRIB triplet. The definition file therefore constitutes an essential link: If the definition files on two institutes do not match 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`).

In theory, this 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 can be obtained via Github

<https://github.com/erget/grib-api.definitions.edzw>

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

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 7.1.1).

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. 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 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 cluster (“`xce.dwd.de`“)

432 compute nodes Intel Haswell (2 CPUs/node, 12 cores/CPU, 62 GiB memory)

544 compute nodes Intel Broadwell (2 CPUs/node, 18 cores/CPU)

MPI:	Cray MPICH 7.0.1
NetCDF:	Version 4.3.2
Compiler:	Cray Fortran v8.4.1

Fortran Compiler	Minimum Version
GNU	gcc v5.1.0
Cray	cftn v8.4.1
Intel	ifort v16.0.0
NAG	nagfor v6.0.1064 ¹

Table 1.1.: Minimum requirements for Fortran compilers for building the ICON code (state *February 2017*)

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.

Minimum requirements for various Fortran compilers are provided in Table 1.1.

The ICON model supports different modes of parallel execution, see Section 4.4 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 multicore 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.

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

¹This is one possible compiler version rather than the minimum requirement. Older versions might work as well.

and software) and sets user specified preferences, e.g. the compiler. 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 subdirectory `icon-dev`. 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 subdirectory with the name of your computer architecture is created within the `build` directory. In this subdirectory, a `bin` subdirectory containing the binary `icon` and several further subdirectories 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.

ICONREMAP

– Used in Sections 2.3.2, 2.3.4

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 resolution.

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 independently at startup.

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.

ICONSUB

– Used in Section 2.3.4

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. Finally, the extracted data is stored in GRIB2- or NetCDF file format.

Multiple sub-areas can be extracted in a single run of `iconsub`.

ICONGRIDGEN

– Used in Section 2.1.2

The `icongridgen` tool is a simple grid generator. 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.

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 subdirectory `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
nag_mpi	thunder	OpenMP + MPI	NAGFOR
nag	thunder	OpenMP	NAGFOR
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
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`, `icondelaunay`) 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 for reading data in GRIB2 format. The GRIB-API is indirectly accessed by the Climate Data Interface (CDI).

1.4. Exercise

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 subdirectory

`/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.

Install also the DWD ICON Tools:

- Change into the subdirectory `dwd_icon_tools` and build the DWD ICON Tools according to Section 1.3.

2. Necessary Input Data

Besides the source code of the ICON package and the 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

2.1.1. ICON Grid Files

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 file is required. For model runs with nested grids, additional files of the nested domains are necessary. Optionally, a reduced radiation grid for the global domain may be used.

The following nomenclature has been established: In general, by $RnBk$ we denote a grid that originates from an icosahedron whose edges have been initially divided into n parts, followed by k subsequent edge bisections. See Figure 2.1 for an illustration of the grid creation process. The total number of cells in a global ICON grid $RnBk$ is given by $n_{\text{cells}} := 20 n^2 4^k$. The effective mesh size can be estimated as

$$\overline{\Delta x} = \sqrt{S_{\text{earth}}/n_{\text{cells}}} \approx 5050/(n 2^k) \text{ [km]}, \quad (2.1)$$

where S_{earth} denotes the earth's surface. Note that by construction, each vertex of a global grid is adjacent to exactly 6 triangular cells, with the exception of the original vertices of the icosahedron, the *pentagon points*, which are adjacent to only 5 cells.

The 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

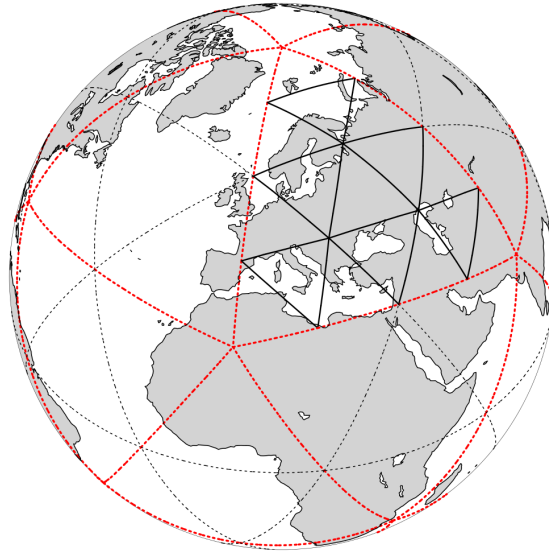


Figure 2.1.: Illustration of the grid construction procedure. The original spherical icosahedron is shown in red, denoted as $R1B00$ following the nomenclature described in the text. In this example, the initial division ($n=2$; black dotted), followed by one subsequent edge bisection ($k=1$) yields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

- `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: `[2, #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`.
- `zonal/meridional_normal_primal_edge`: components of the normal vector at the triangle edge midpoints.

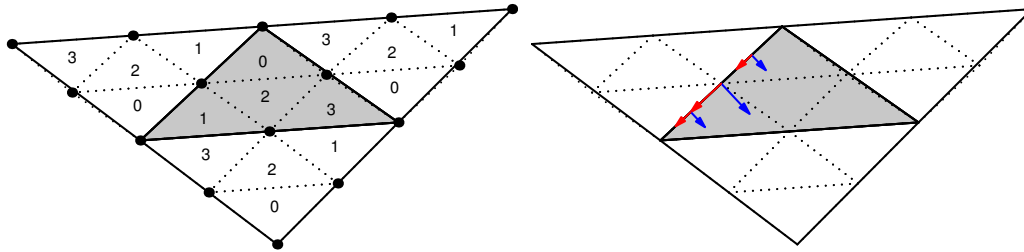


Figure 2.2.: Illustration of the parent-child relationship in refined grids. *Left:* Triangle subdivision and local cell indices. *Right:* The grids fulfil the ICON requirement of a right-handed coordinate system $[\vec{e}_t, \vec{e}_n, \vec{e}_w]$.

Refinement Information

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. This optional *grid connectivity* file (suffix `-grfinfo.nc`) 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.

2.1.2. Grid Generator

Introductory Remarks

There are (at least) three grid generation tools available for the ICON model: The ICON model itself is shipped together with a standalone tool `grid_command`. The executable file of the grid generator `grid_command` is created automatically during the build process and is located in the same subdirectory as the model binary. We refer to the documentation `icon-dev/doc/Namelist_overview.pdf` for details. A different grid generation tool has been developed at the Max-Planck-Institute for Meteorology by L. Linardakis. Finally, another grid generator is contained in the DWD ICON Tools.

In this section we will discuss the grid generator 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 2.2.2. It is important to note, however, that this grid generator is not capable of generating grids with unusual root subdivisions or non-spherical geometries like torus grids.

Grid Generation Example

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: As a second constraint, in the case that the parent grid itself is a bounded regional grid, no cells can be refined that are part of the indexing region (of width `bdy_indexing_depth`) in the vicinity of the parent grid’s boundary.

Settings for ICON-LAM

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

- *Identifying the grid boundary zone.* In Section 2.3.4 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 2.3.4.

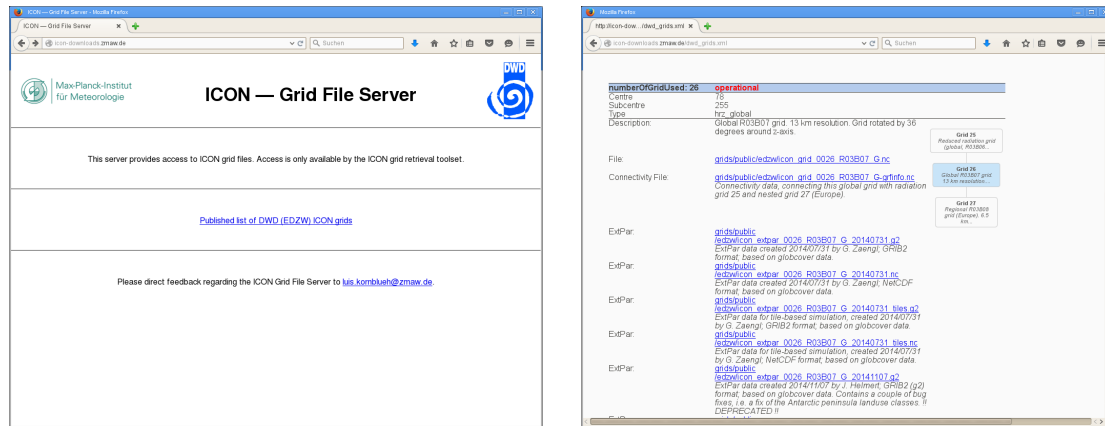


Figure 2.3.: Screenshots of the ICON download server hosted by the ZMAW in Hamburg.

- *Generation of a coarse-resolution radiation grid* (see Section 6.2 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.

2.1.3. Download of Predefined Grids

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

The contents of the primary storage directory are regularly mirrored to a public web site for download, see Figure 2.3 for a screenshot of the ICON grid file server. The download server can be accessed via

<http://icon-downloads.zmaw.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`).

2.1.4. 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. 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:

- **numberOfGridUsed**
This is simply an integer number, as explained in the previous section. 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 (defensive I/O).

2.2. External Data Files

2.2.1. ExtPar Products

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.

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 and domain-specific data files. For a more detailed overview of ExtPar, the reader is referred to the *User and Implementation Guide* of ExtPar. The ExtPar preprocessor is a COSMO software and not part of the ICON training course release.

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 4.3 and Appendix C).



Additional information for surface tiles: ExtPar data is available with and without additional information for surface tiles.

Tiles are a means to adequately represent the sub-grid surface heterogeneity in each surface grid element. Following the basic idea of [Avissar and Pielke \(1989\)](#), patches of the same surface type occurring within a grid element are regrouped into homogeneous classes (tiles). The surface energy balance and soil physics are then computed separately for each tile, using parameters characteristic for each surface type (z_0 , leaf area index LAI, albedo, ...). The contributions from the different tiles are areally weighted to finally provide the cell-averaged atmospheric forcing.

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 (`lnd_nml/ntiles = 1`).

The data files do not differ in the number of fields, but only 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.



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 subdirectory `icon-dev/data`.

2.2.2. Web-based Generation of Grids and External Parameters

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.

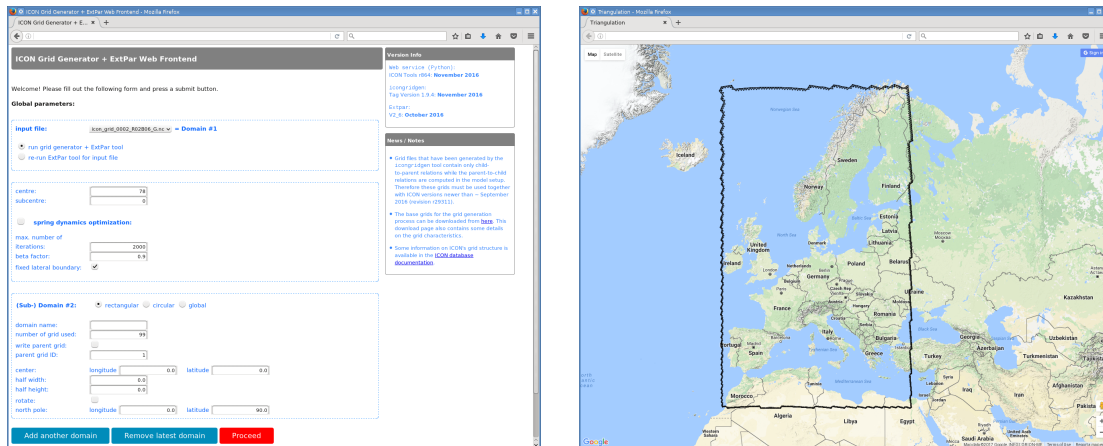


Figure 2.4.: Web browser screenshot of the web-based ICON grid generator tool. *Left*: Web form. *Right*: HTML visualization of the resulting grid based on Google Maps.

A user account is required for access: Please contact Klima.Vertrieb@dwd.de to this end. Then, visit the web page of the *pamore data service*

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

and, after logging into the web site, choose *External Parameters for ICON and COSMO* → *External Parameters and Grid Files for ICON*.

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 Section 2.1.2 of this tutorial. 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!** 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.



Surface tiles: The web-based generator can produce ExtPar data with and without additional information for surface tiles (see the explanation on p. 19). The – recommended – default has the tile data enabled. This setting can be changed by disabling the corresponding checkbox in the HTML form.



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.

2.3. Initial and Boundary 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*¹. 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.

2.3.1. DWD Analysis 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 9 for more information on DWD's data assimilation system.

When started in this *DWD initialization mode*, the model loads two files: 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. This distinction is not strictly necessary for the initialization process, however it helps to clarify which fields have been touched and updated by the assimilation system (by making use of observations) and which not. The *DWD initialization mode* comes in two flavours, which differ in the way the model state is pulled towards the analysed state.

Non-Incremental Update

The *non-incremental update* is conceptually the easiest approach: It starts the model from the full analysis directly. However, this comes at the price of a somewhat increased noise level at the beginning of the simulation, due to a missing filtering procedure.

Both the model's current state, the *first guess*, and the analysis must have the same validity time. Table 2.1 provides an overview of the fields that ICON expects to be contained in the *first guess* and *analysis* file at 00UTC. A "yes" in column 2 and 4 indicates that the corresponding variable is expected to be present in the first guess and analysis file, respectively. This table shows the optimum situation in the sense that all requested fields have been found in the input files. This table is part of the ICON runtime log output.

¹Note that for so-called *idealized test cases* no initial conditions must be read in. All necessary state variables are preset by analytical values.

As will be explained in Section 9.2, the atmospheric analysis is performed more frequently compared to the surface analysis. Therefore, the analysis product provided at times different from 00UTC usually contains only a subset of the fields provided at 00UTC. Consequently, Table 2.1 will differ for non-00UTC runs in the sense that the fields

`fr_seaice` `h_ice` `t_ice` `t_so` `w_so`

will be read from the first guess file instead of the analysis file.

Incremental Analysis Update

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 analysed 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 00UTC. 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 00UTC. The only exceptions are `FR_ICE` and `T_SO`, which must be full fields (see Table 2.2).
- 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 22:30UTC of the previous day. This is because “dribbling” of the analysis increments is performed over the symmetric 3 h time window $[00UTC - 1.5h, 00UTC + 1.5h]$. See Section 4.2.3 for an illustration of this process.

Table 2.2 provides an overview of the fields that ICON expects to be contained in the first guess and analysis file at 00UTC. ICON internal variable names are given in column 1. A “yes” in columns 2 or 4 indicates that the corresponding variable is expected to be present in the first guess and analysis file, respectively. This table shows the optimum situation in the sense that all requested fields have been found in the input files. This table is part of the ICON runtime log output.

As already explained in the previous section, the analysis product provided at times different from 00UTC will only contain a subset of the fields provided at 00UTC. Table 2.2 will differ for non-00UTC runs in the way that the fields

`fr_seaice` `t_so` `w_so`

will be read from the first guess and not from the analysis file.

variable	FG read attempt	FG found	ANA read attempt	ANA found	data from
vn	yes	yes	no		fg
w	yes	yes	no		fg
rho	yes	yes	no		fg
theta_v	yes	yes	no		fg
qc	yes	yes	no		fg
qi	yes	yes	no		fg
qr	yes	yes	no		fg
qs	yes	yes	no		fg
tke	yes	yes	no		fg
gz0	yes	yes	no		fg
t_g	yes	yes	no		fg
t_mnw_lk	yes	yes	no		fg
t_wml_lk	yes	yes	no		fg
h_ml_lk	yes	yes	no		fg
t_bot_lk	yes	yes	no		fg
c_t_lk	yes	yes	no		fg
t_b1_lk	yes	yes	no		fg
h_b1_lk	yes	yes	no		fg
qv_s	yes	yes	no		fg
w_i	yes	yes	no		fg
t_so	yes	yes	yes	yes	both
w_so_ice	yes	yes	no		fg
w_snow	yes	yes	no		fg
rho_snow	yes	yes	no		fg
qv	yes	yes	yes	yes	ana
u	no		yes	yes	ana
v	no		yes	yes	ana
temp	no		yes	yes	ana
pres	no		yes	yes	ana
t_ice	yes	yes	yes	yes	ana
h_ice	yes	yes	yes	yes	ana
fr_seaice	yes	yes	yes	yes	ana
w_so	yes	yes	yes	yes	ana
t_snow	yes	yes	yes	yes	ana
h_snow	yes	yes	yes	yes	ana
freshsnow	yes	yes	yes	yes	ana

Table 2.1.: First guess (FG) and analysis (ANA) input fields as required when starting from DWD analysis at 00UTC **without IAU**. “yes/no” in columns 2 and 4 indicates whether a field is expected or not, while “yes/no” in columns 3 and 5 shows whether a field was found and read, or not. Finally, column 6 indicates whether the respective field was taken from the FG or ANA or both sources.

IAU and limited area: For limited area runs it is not possible to make use of the IAU method.



variable	FG read attempt	FG found	ANA read attempt	ANA found	data used
vn	yes	yes	no		fg
w	yes	yes	no		fg
rho	yes	yes	no		fg
theta_v	yes	yes	no		fg
qv	yes	yes	yes	yes (I)	both
qc	yes	yes	no		fg
qi	yes	yes	no		fg
qr	yes	yes	no		fg
qs	yes	yes	no		fg
tke	yes	yes	no		fg
gz0	yes	yes	no		fg
t_g	yes	yes	no		fg
t_ice	yes	yes	no		fg
h_ice	yes	yes	no		fg
t_mnw_lk	yes	yes	no		fg
t_wml_lk	yes	yes	no		fg
h_ml_lk	yes	yes	no		fg
t_bot_lk	yes	yes	no		fg
c_t_lk	yes	yes	no		fg
t_b1_lk	yes	yes	no		fg
h_b1_lk	yes	yes	no		fg
qv_s	yes	yes	no		fg
w_i	yes	yes	no		fg
t_so	yes	yes	yes	yes	both
w_so	yes	yes	yes	yes (I)	both
w_so_ice	yes	yes	no		fg
t_snow	yes	yes	no		fg
rho_snow	yes	yes	no		fg
h_snow	yes	yes	yes	yes (I)	both
freshsnow	yes	yes	yes	yes (I)	both
snowfrac_lc	yes	yes	no		fg
u	no		yes	yes (I)	ana
v	no		yes	yes (I)	ana
temp	no		yes	yes (I)	ana
pres	no		yes	yes (I)	ana
fr_seaice	yes	yes	yes	yes	ana

Table 2.2.: First Guess (FG) and Analysis (ANA) input fields as required when starting from DWD analysis at 00UTC **with IAU**. “yes/no” in columns 2 and 4 indicates whether a field is expected, while “yes/no” in columns 3 and 5 shows whether a field was found and read. The marker (I) in column 5 highlights analysis increments as opposed to full analysis fields. Finally, column 6 indicates whether the respective field was taken from the FG or ANA or both sources.

Downloading DWD Analysis

The ICON code contains a script for the automatic request of native (DWD) analysis data from DWD’s meteorological data management system SKY. It is located in the subdirectory

```

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

`icon-dev/scripts/preprocessing/sky4icon`

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, starting from January 1, 2015 until today.

In order to retrieve, for example, 6-hourly initial data from July 1, 2015 00UTC until July 3, 2015 00UTC for an IAU-based model initialization, the following command line is used:

```
./sky4icon 20150701000000 20150703000000 --increment 6
```

A full set of command-line options can be obtained via `sky4icon.py -h`, see Fig. 2.5.

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 km horizontal resolution (member EMEMBER), as an alternative to the high-resolution (13 km) deterministic analysis produced by the EnVAR system.



A pre-requisite for running the script is a valid account for the database “roma” (contact datenservice@dwd.de). An alternative way to get the data is to contact DWD’s data center (see Section 0.2).

2.3.2. 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). Initializing the ICON model with IFS analysis files requires an additional pre-processing step using the DWD ICON Tools. The details of this procedure are given below.

Downloading IFS Data

The ICON code contains a script for the automatic request for IFS data from the MARS data base. A full list of mandatory IFS analysis fields is provided in Table 2.3. 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.

The script for importing from MARS is part of the ICON source code repository. It is located in the subdirectory

```
icon-dev/scripts/preprocessing/mars4icon_smi
```



Important note:

The `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`

Remapping the IFS Data

After the successful MARS download, the IFS data must be interpolated from a regular grid onto the ICON 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 IFS data has the following structure:

```
&remap_nml
  in_grid_filename = "${IFS_FILENAME_GRB}"
  in_filename      = "${IFS_FILENAME_GRB}"
  in_type          = 1                ! regular grid
  out_grid_filename = "${ICON_GRIDFILE}"
```



```

out_filename      = "${IFS_FILENAME_NC}"
out_type          = 2                ! ICON grid
out_filetype      = 4                ! NetCDF format
/

! DEFINITIONS FOR IFS INPUT DATA
!
&input_field_nml ! temperature
  inputname       = "T"
  outputname      = "T"
/
&input_field_nml ! horiz. wind comp. u and v
  inputname       = "U", "V"
  outputname      = "VN"
/
&input_field_nml ! vertical velocity
  inputname       = "OMEGA"
  outputname      = "W"
/
&input_field_nml ! soil moisture index layer 1
  inputname       = "SWVL1"
  outputname      = "SMIL1"
/
...

```

The control file must contain a separate namelist `input_field_nml` for each field of the list of mandatory input fields given in Table 2.3. The u and v wind components require special treatment. These must be interpolated to edge-normal wind components v_n (see the namelist above).

Note that 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 `SWVL x` . The ICON model, however, expects them to be named `SMI x` . Therefore, the proper output name `SMI x` must be specified explicitly in the namelist `input_field_nml` of `iconremap` (see the example namelist above).

The DWD ICON Tools contain example run scripts for `iconremap` for a small number of computing environments. For the Cray XC 40 environment, see

```
dwd_icon_tools/example/runscripts/xce_ifs2icon.run
```

for a script that performs an interpolation of IFS data. Of course, the namelist parameters which are specified in this file can be also ported for corresponding runs of `iconremap` on other platforms. A detailed documentation of the ICON remap namelist parameters can be found under `dwd_icon_tools/doc/icontools_doc.pdf`, i. e. Prill (2014).

2.3.3. ICON-to-ICON Initial Data

Sometimes 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.

For limited area runs, the set of variables which must be interpolated onto the target resolution is identical to the first guess dataset in Table 2.1. To be more precise, all variables for which the data source in Table 2.1 indicates “fg” or “both” are necessary and must be remapped. Either analysis or forecast datasets can be used. Due to the necessary remapping step, however, it is not possible to use tiled surface data (the surface tile approach is used operationally in ICON). Instead, aggregated surface fields must be remapped. Remapping of the tiled datasets makes no sense, since the tile-characteristics can differ significantly between a source and target grid cell.

The DWD ICON Tools contain a run script `xce_remap_inidata` which performs the remapping. After adjusting the necessary file and directory names in

```
dwd_icon_tools/icontools/xce_remap_inidata
```

the script can be submitted to the PBS batch system of the Cray XC 40.

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.



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.
- 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 and transfer it back to `W_SO` afterwards.

2.3.4. Boundary Data From a Driving Model (ICON-LAM Mode)

During the limited area simulations with ICON, boundary conditions are updated periodically by reading input files. To this end, forecast or analysis data sets from a *driving model* need to be available. These data sets may originate from ICON, IFS and COSMO-DE.² Between two lateral boundary data samples the boundary data is linearly interpolated.

In this section we shortly describe the process of generating boundary data for these runs. The boundary data files must contain the following set of variables

²Data sets from other global or regional models may work as well, but have not been tested yet.

U, V, W, THETA_V, DEN, QV, QC, QI, QR, QS, HHL
or, alternatively,
U, V, W, T, P, QV, QC, QI, QR, QS, HHL.

The basic preprocessing steps for ICON-LAM are visualized in Figure 2.6.

ICON-LAM Preprocessing Script

The DWD ICON Tools contain a run script `xce_limarea` which processes a whole directory of data files (“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.

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.

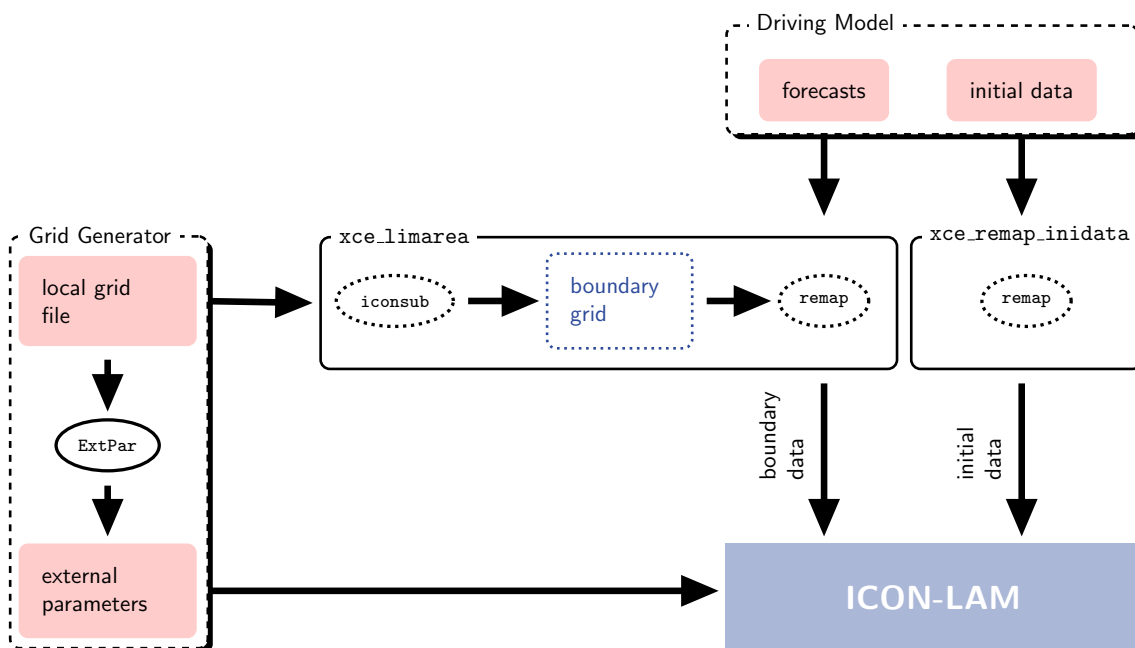


Figure 2.6.: Preprocessing steps for the limited area model ICON-LAM. The grid generation process is described in Sections 2.1 – 2.2. The initial data processing is covered by Section 2.3.3. Finally, the script `xce_limarea` for extracting the boundary data is described in Section 2.3.4.



Important note: The 3D field HHL field (geometric height of model half levels above mean sea level) is constant data. However, due to technical reasons, HHL is required in every boundary data file.

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   = "grid_file.nc",
  output_type     = 4,
  lwrite_grid     = .TRUE.,
/
&subarea_nml
  ORDER           = "grid_file_lbc.nc",
  grf_info_file   = "grid_file.nc",
  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 2.7.

Creating Boundary Data

Any of the data sources explained in the Sections 2.3.1 and 2.3.2 can be chosen for the extraction of boundary data. To be more precise, it is possible to extract boundary data from ICON, IFS, and COSMO-DE forecasts.

To this end, 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 = "input_grid_file"
  in_filename      = "input_data_file"
  in_type          = 2
  out_grid_filename = "grid_file_lbc.nc"
  out_filename     = "data_file_lbc.nc"
  out_type         = 2
  out_filetype     = 4
/
```

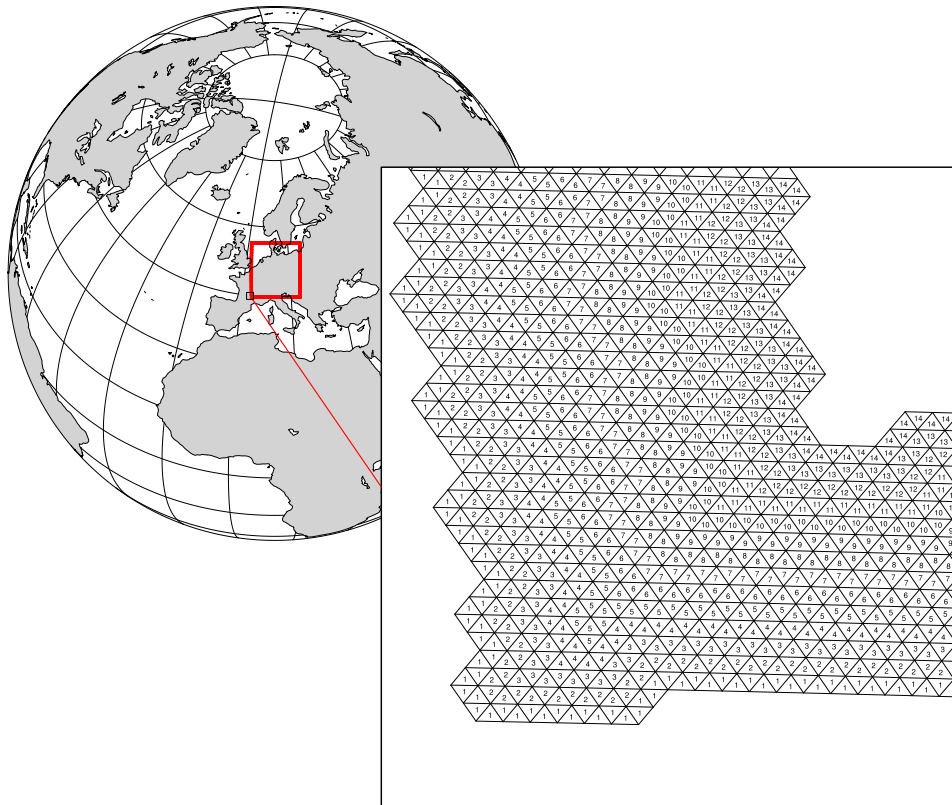


Figure 2.7.: Illustration of the ICON-LAM boundary zone. The cells in this boundary zone are identified by their `refin_c_ctrl` index, e.g. `refin_c_ctrl = 1, ..., 14`.

Here, 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 5.1.4 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 process the whole list of boundary data samples (this is automatically done within the `xce_limarea` script).

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. Exercises

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

Download of global data: Retrieve the necessary grids and external parameter files from the ICON download server (see Section 2.1.3).

- Open the download page for the pre-defined ICON grids <http://icon-downloads.zmaw.de> in your web browser.
- Pick the R2B06 grid no. 24 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 input subdirectory of `case2`. Download the grid file into this subdirectory by typing `wget link location -e http-proxy=ofsquid.dwd.de:8080`.
- Repeat the last steps for downloading the grid connectivity information (see Section 2.1.1 for explanation) and for the R2B07_N02 grid no. 28 (EU-nest).
- Pick the NetCDF version of the most recent ExtPar (external parameter) file in the browser list (creation date 2015-08-05) and perform the previous steps in order to download this file as well. Watch out that the ExtPar file matches with the grid, i. e. make sure that both filenames contain the same number (24). The ExtPar data file must have the `tile` suffix. Repeat this step for the ExtPar file that matches with the grid no. 28.
- Download the R2B05 grid no. 23 together with its grid connectivity data. This is the reduced (coarser) grid for radiation.
- Take a look at the grid data and the external parameters using the `ncdump` utility, see Section 7.1.1 for details. Find out whether the external parameter fields are defined at the grid vertices, edge midpoints or cells.



EX 2.2

Retrieving DWD analysis data for global forecast runs:

- Request native analysis data from DWD's meteorological data management system SKY for the date 2017-01-12T00:00:00 at a horizontal resolution of 40 km using the `sky4icon` script, see Section 2.3.1:

```
./sky4icon 20170112000000 20170112000000 --mode 1 \  
--ensemble --emember 1
```

- 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.

Retrieving IFS data:

Prepare the initial data for ICON to start from an IFS analysis.

- Start the `mars4icon` script and download an IFS analysis file for the date 2017-01-12T00:00:00, see Section 2.3.2.
- Interpolate the data horizontally from the lat/lon grid onto the triangular ICON grid using `iconremap`. The run script `case4/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 2.3.2.
- 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 2.3. Besides, find out which field(s) are not defined on cell circumcenters.



EX 2.3

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. 4.5.

Local grid file and its external parameters:

- In the directory `case3/input` you find an archive file which is the result of the web-based grid generator described in Section 2.2.2.

Uncompress this file and visualize its content with the NCL file `case3/plot_grid.ncl`. See Figure 5.2 for a reference.
- Investigate the grid file with the `ncdump` utility (see Section 7.1.1): 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.



EX 2.4

The subdirectory `case3` 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. 4.5 for creating the initial data (`SOURCE`).

- Identify the resolution of both grids in ICON’s `RxBY` nomenclature. (You could make use of `ncdump`.)

<code>SOURCE grid</code>	<code>remap data</code>	<code>TARGET grid</code>
<input style="width: 100px; height: 20px;" type="text"/>	→	<input style="width: 100px; height: 20px;" type="text"/>

- If the source grid has a horizontal resolution of ≈ 20 km, what is the resolution 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 `case3/output`.

Remapping of boundary data:

- The subdirectory `case3` contains a copy of the DWD ICON Tools script `xce_limarea`, see Section 2.3.4. 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 `case3/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?

Table 2.3.: Mandatory IFS analysis fields on a regular lat-lon grid, as retrieved by the script `mars4icon_smi`.

ShortName (ECMWF)	Description
U, V	horizontal velocity components
OMEGA	vertical velocity
T	Temperature
FI	geopotential (at model levels)
QV	specific humidity
CLWC	cloud liquid water content
CIWC	cloud ice content
CRWC	rain water content
CSWC	snow water content
SST	sea surface temperature
CI	sea-ice cover
LNSP	logarithm of surface pressure
Z	surface geopotential
TSN	snow temperature
SD	water content of snow
RSN	density of snow
ASN	snow albedo
SKT	skin temperature
STL1	soil temperature level 1
STL2	soil temperature level 2
STL3	soil temperature level 3
STL4	soil temperature level 4
SWVL1	soil moisture index (SMI) layer 1
SWVL2	soil moisture index (SMI) layer 2
SWVL3	soil moisture index (SMI) layer 3
SWVL4	soil moisture index (SMI) layer 4
SRC	water content of interception storage
SR	surface roughness
LSM	land/sea mask

3. 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, all initial conditions are computed within the ICON model itself, based on analytical functions. These are either evaluated pointwise 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.

3.1. 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-dev/doc/Namelist_overview.pdf`

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 also be found in the namelist documentation, mentioned above.

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 below.

3.2. Vertical coordinates

The total number of vertical levels has to be specified separately 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.

The treatment of terrain in ICON is handled through the use of height-based terrain following coordinates. Two formulations are available, which are briefly described below.

3.2.1. 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

$$\begin{aligned} z(x, y, \eta) &= \frac{(H - B'(\eta) h(x, y))}{H} \eta + B'(\eta) h(x, y) \\ &= \eta + B'(\eta) \left(1 - \frac{\eta}{H}\right) h(x, y), \end{aligned} \quad (3.1)$$

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 (3.1) is also written in the discretized form

$$z_h(x, y, k) = A(k) + B(k) h(x, y), \quad k = 1, \dots, \text{nlev} + 1 \quad (3.2)$$

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 SLEVE vertical 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 (3.2)) from which the half level heights $z_h(x, y, k)$ can be deduced. $A(k)[\text{m}]$ are fixed height values, with $A(1)$ defining the model top height H and $A(\text{nlev} + 1) = 0$ m. The dimensionless values $B(k)$ control the

vertical decay of the topography signal, with $B(1) = 0$ and $B(\text{nlev} + 1) = 1$. Thus, at each horizontal grid point $z_h(x, y, 1)$ is the model top height, while $z_h(x, y, \text{nlev} + 1)$ is the surface height.

The structure of the expected input file is depicted in Table 3.1. Example files can be found in `icon-dev/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, `grid_nml/vertical_grid_filename!`

```
# 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 3.1.: Structure of vertical coordinate table as expected by the ICON model.

3.2.2. 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.

Configuring the SLEVE Coordinate

The main switch for selecting the SLEVE vertical coordinate is

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

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`.

3.3. 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 midlatitudes 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

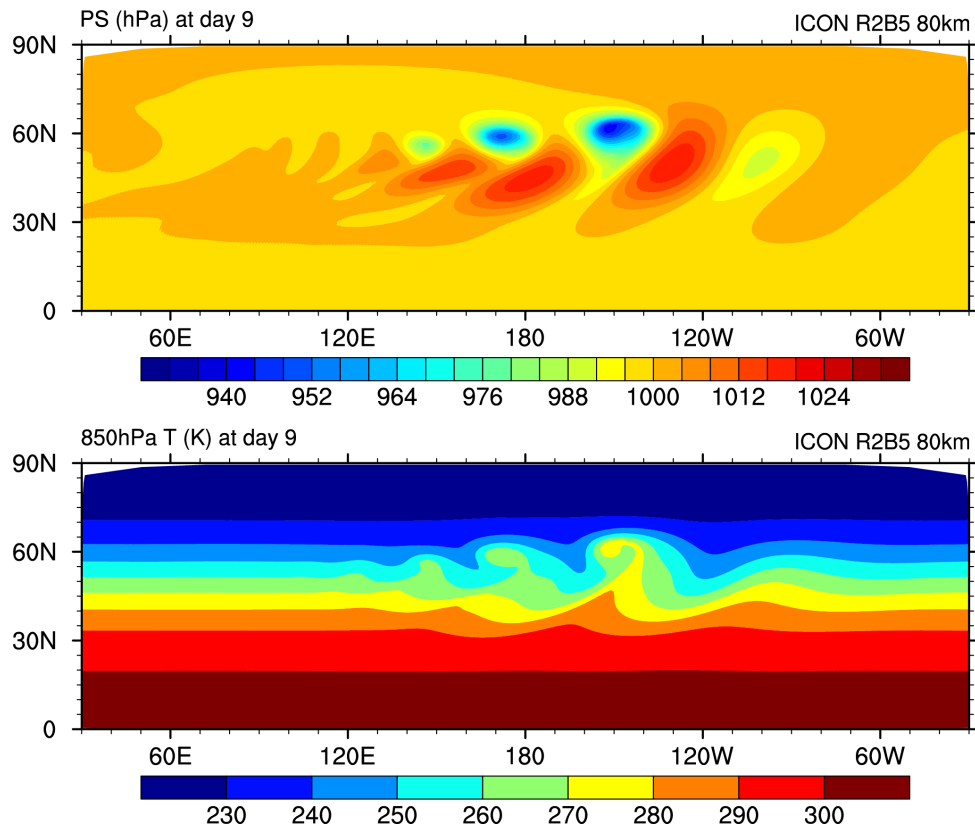


Figure 3.1.: Surface Pressure and 850 hPa Temperature at day 9 for the Jablonowski-Williamson test case on a global R2B5 grid.

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 3.1). If the integration is continued, additional instabilities become more and more apparent especially near the pentagon points (see Section 2.1.1), 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 behaviour 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 3.2. See Jablonowski et al. (2008) for further information on the initial distributions.

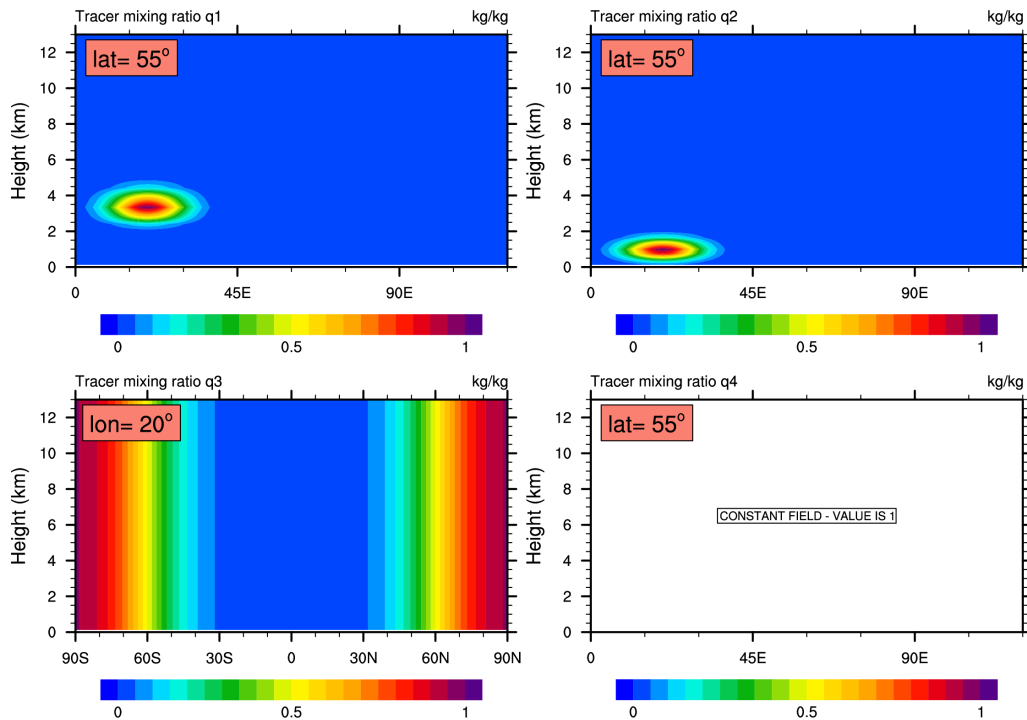


Figure 3.2.: Initial tracer distributions which are available for the Jablonowski-Williamson test case. Tracer q_3 only depends on the latitudinal position, and tracer q_4 is constant.

3.3.1. 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.

`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 5.1. In general, the setting of `iforcing` depends on the selected testcase.

`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`. If set to `.FALSE.`, transport of passive tracers is switched off completely.

Namelist `nh_testcase_nml`:

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

Main switch for selecting a testcase. `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.

3.3.2. One-Way Nesting and Two-Way-Nesting

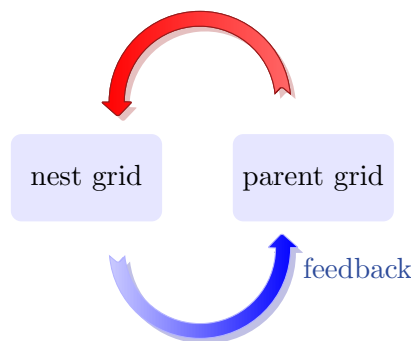
While the above switches are necessary to specify the type of simulation (idealized vs. real-case), nothing has been specified so far regarding the computational domain. ICON has the capability for running

- global simulations on a single global grid,
- global simulations with refined nests (so called patches or domains), and
- limited area simulations, see Chapter 5.

Here, the use of nests requires an additional remark. The refined nests are tightly embedded into the global simulation in the sense that the prognostic variables are synchronized in every (dynamical-core) time step of the parent domain.

In more detail this can happen in two ways:

In a *one-way nested* simulation the prognostic fields (or, in the default setup: the respective tendencies) are prolonged within the nest boundary region from the coarser parent grid to the finer nest grid. They are incorporated into the next iteration of the nest's dynamical core, but the fine-scale variables do not influence the global state.



On the other hand, the result state of the nested grid may also be transferred back to the coarser parent grid in a feedback loop. In ICON this is called a *two-way nested* simulation.



Note for advanced users: 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!

3.3.3. 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).

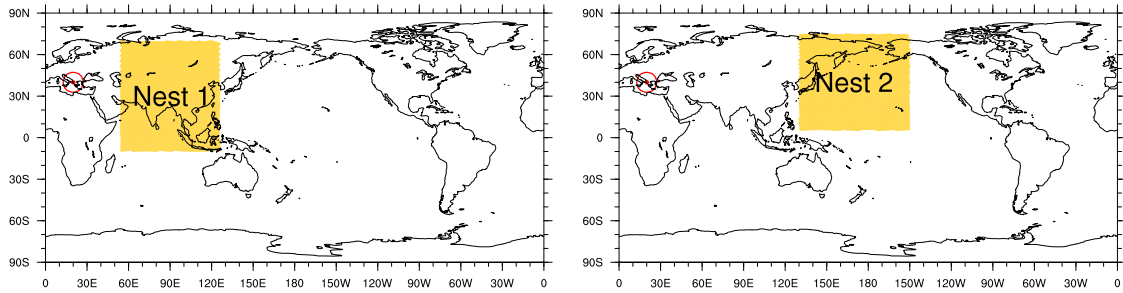


Figure 3.3.: Location of available nests for the baroclinic wave test case. The perturbation triggering the baroclinic wave is centered at (20°E, 40°N) (red circle).

`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. A value of 0 indicates that a domain has no parent domain (i.e. the global domain).

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 3.3)

```
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 2.1.1.

3.3.4. Basic Control Variables

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 4.1.

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 3.1 contains three output namelists. The details of the model output specification are discussed in Section 4.3.

3.4. Exercises

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 3.1

Preparations:

Retrieve the necessary grid file from the ICON download server (see Section 2.1.3):

- Open the download page for the pre-defined ICON grids <http://icon-downloads.zmaw.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 subdirectory `test_cases/case1/input`. Download the grid file into this subdirectory 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/case1`. 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 3.3.1 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`.
- Go to the output directory of `case1`. 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 7.1.
What output interval (in h) has been used? – Answer: h
- Visualize the output with one of the tools described in Section 7. An NCL script named `JW_plot.ncl` is available in `test_cases/case1`.
- Compare the output of the NCL script with the reference `JABW_DOM01.ps` given in subdirectory `reference`.

Repeat the run of the previous exercise with a nested subdomain.

- 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 `atmo_dyn_grids`, `num_lev` (`run_nml`) and `dynamics_parent_grid_id` (`grid_nml`), see Section 3.3.3 for additional help. The same output fields as for the global domain will be generated for the regional domain(s).
The location of the prepared nests is depicted in Figure 3.3. 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 7. When using NCL (`JW_plot.ncl`), you will have to adapt `workdir` and `domain_nr`.
- Also 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?



EX 3.2

Optional Exercise: Tracer Advection



EX 3.3

The JW test case provides a set of four pre-defined idealized tracer fields (see Figure 3.2). Here, we will learn how to enable and control the transport of passive tracers in idealized tests.

- Enable tracer advection:
 - Go to the directory `test_cases/case1` 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 3.2). 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 `ml_varlist` and `pl_varlist`).

For idealized testcases, 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?

4. Real Data Test Cases

In this lesson you will learn about the ICON time-stepping and how to initialize and run the ICON model in a realistic NWP setup. Data provided by DWD's Data Assimilation Coding Environment (DACE) will serve as initial conditions. Before we conclude this chapter with some exercises, the different mechanisms for parallel execution of the ICON model will be discussed.

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

An illustration of the relationship between the time steps can be found in Figure 4.1. More details on the physics-dynamics coupling will be presented in Section 5.2.2.

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

$$\Delta \tau = 1.8 \cdot 10^{-3} \overline{\Delta x} \frac{\text{s}}{\text{m}}, \quad (4.1)$$

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

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

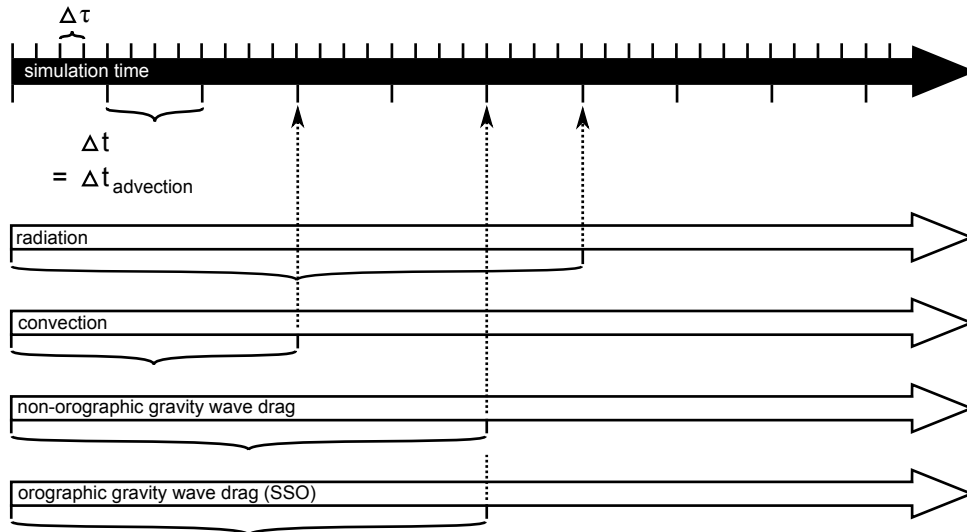


Figure 4.1.: ICON internal time stepping. Sub-cycling of dynamics with respect to transport, fast-physics, and slow-physics. Δt denotes the time step for transport and fast physics and $\Delta\tau$ denotes the short time step of the dynamical core. The time step for slow-physics can be chosen individually for each process. Details of the physics-dynamics coupling will be discussed in Chapter 5.2.

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.



Time step for nested domains: In case of nested setups, the time step Δt needs to be specified for the global domain only. The adaption for nested regions is done automatically, by multiplying Δt with a factor of 0.5 for each nesting level. This factor is hard-coded.

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 1000s, which becomes relevant when Δx is larger than about 125 km.

Even longer time steps than Δt can be used for the so-called slow-physics parameterizations, i.e. radiation, convection, non-orographic gravity wave drag, and orographic gravity wave drag. These parameterizations provide tendencies to the dynamical core, allowing them to be called individually at user-specified time steps. The related namelist switches are `dt_rad`, `dt_conv`, `dt_gwd` and `dt_sso` in `nwp_phy_nml`. If the slow-physics time step is not a multiple of the advective time step, it is automatically rounded up to the next advective time step. A further recommendation is that `dt_rad` should be an integer multiple of `dt_conv`, such that radiation and convection are called at the same time. The time-splitting is schematically depicted in Figure 4.1.

4.2. 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 and land,
- boundary data in the case of limited area runs.

ICON is capable of reading analysis data from various sources (see Section 2.3), including data sets generated by DWD's Data Assimilation Coding Environment (DACE) and interpolated IFS data. Boundary data for limited area runs can be taken from forecasts/analysis of the latter two models as well as from COSMO-DE. In the following we provide some guidance on the basic namelist settings for real data runs, according to the data set at hand and chosen initialization mode.

4.2.1. 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. Many of them have already been discussed in Section 3.1, so we will mainly concentrate on their settings for real data runs. As before, settings appropriate for the exercises in this chapter are given in [red](#).

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.

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**)

Transport of (passive) tracers must be switched on. This is necessary for the transport of cloud and precipitation variables. Details of the transport schemes can be controlled via `transport_nml`.

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 3.3.3 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 0 indicates that a domain has no parent domain (i.e. the global domain).

`radiation_grid_filename` (**namelist** `grid_nml`, **string parameter**)

If the radiative transfer computation should be performed on a coarser grid than the dynamics (one level coarser, resolution $2\Delta x$), the name(s) of the grid file(s) to be used for radiation must be specified here. See Section 6.2 for further details.

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 parameter.

`extpar_filename` (**namelist** `extpar_nml`, **string parameter**)

Filename(s) of external parameter input file(s). 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 4.2.2 for additional keywords.

Namelist `initicon_nml`:

As mentioned previously, ICON allows for different real data initialization modes. 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 DWD analysis without the incremental analysis update (IAU) procedure (see Section 2.3.1): `init_mode = 1`
- start from interpolated IFS analysis (see Section 2.3.2): `init_mode = 2`
- start atmosphere from interpolated IFS analysis and soil/surface from interpolated ICON/GME fields: `init_mode = 3`
- start from DWD analysis, and make use of the IAU procedure to reduce initial noise (see Section 2.3.1): `init_mode = 5`
- The initialization modes `init_mode = 4` and `init_mode = 7` start from interpolated COSMO-DE or ICON/IFS forecasts for limited area runs. Limited area simulations are a feature which will be addressed in Chapter 5.

The most relevant modes are modes 1, 2, 5 and 7. Modes 1, 2 and 5 are relevant for global ICON simulations with and without nests. They will be explained in more detail below.

ICON supports NetCDF and GRIB2 as input format for the DWD input fields. In this context it is important to note that the field names that are used in the GRIB2 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 DWD GRIB2 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 short names. An example can be found in `icon-dev/run/ana_varnames_map_file.txt`.

4.2.2. Starting from DWD Analysis

Given that a valid DWD analysis data set for non-incremental update is available (see Section 2.3.1), starting from DWD analysis data is basically controlled by the following three namelist parameters:

`init_mode = 1` (**namelist** `initicon_nml`, **integer value**)

To start from 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 (3h forecast) fields, which served as background fields for the assimilation process. The second one contains the analysis fields produced by the assimilation process.

`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 resolution $R_x B_y$ and the domain number DOM_{ii} . Filenames are treated case sensitive.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 R_x (single digit)
<code><nroot0></code>	grid root division R_x (two digits)
<code><jlev></code>	grid bisection level B_y (two digits)
<code><idom></code>	domain number (two digits).

4.2.3. Starting from DWD Analysis with IAU

As described in Section 2.3.1, incremental analysis update is a means to reduce the initial noise which typically results from small scale non-balanced modes in the analysis data set. Given that a valid DWD analysis data set for incremental update is available (see Section 2.3.1), starting from DWD analysis data is basically controlled by the following namelist parameters:

¹More precisely this behaviour depends on the file system: UNIX-like file systems are case sensitive, but the HFS+ Mac file system (usually) is not.

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 second one contains the analysis fields (mostly increments) produced by the assimilation process.

dwdfg_filename (namelist initicon_nml, string parameter)

Filename(s) of the DWD first guess input file(s) for each domain. See Section 4.2.2 for an explanation of the filename structure.

dwdana_filename (namelist initicon_nml, string parameter)

Filename(s) of the DWD analysis input file(s) for each domain. See Section 4.2.2 for an explanation of the filename structure.

The behaviour 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 2.3.1 and depicted in Figure 4.2, you have to make sure that the first guess is shifted ahead in time by $-0.5 dt_iau$ w.r.t. the analysis. The model start time `ini_datetime_string` must match the validity time of the analysis.

4.2.4. 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.

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 4.2.2, the filenames have the form

```
ifs2icon_filename = "ifs2icon_RnrootBjlev_DOMidom.nc"
```

Remember to make sure that the model start time given by `ini_datetime_string` matches the validity date of the analysis input file.

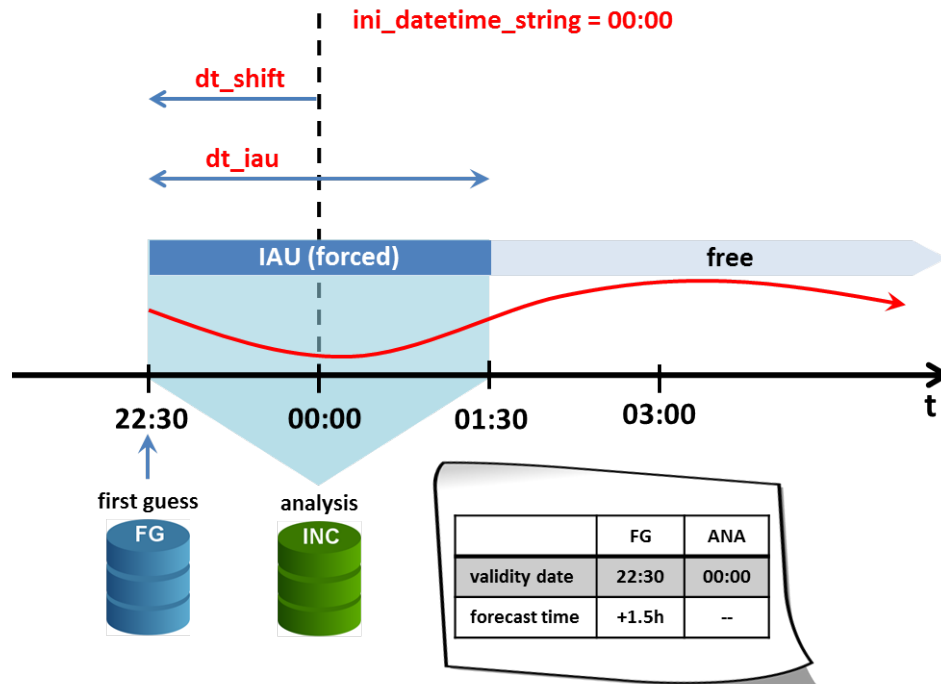


Figure 4.2.: Schematic illustrating typical settings for a global ICON forecast run starting from a DWD analysis **with IAU** at 00UTC. IAU is performed over a 3 h time interval (dt_iau), with the model start being shifted ahead of the nominal start date by 1.5 h (dt_shift).

4.3. 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 6.3](#).

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 6.3.

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 model domain 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.

4.4. Parallelization Aspects

As mentioned in the introduction, 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.

4.4.1. 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 I/O server tasks
<i>Restart PEs</i>	for asynchronous restart writing (see Section 6.4)
<i>Prefetch PE</i>	for asynchronous read-in of boundary data in limited area mode (see Section 5.1.5)
<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 6.4), 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

```
#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:

<code>#PBS -q xc_norm_h</code>	To put a job to the (Haswell) compute nodes of the XC 40.
<code>#PBS -l select=\$NODES</code>	To specify the number of compute nodes.
<code>#PBS -l place=pack</code>	If only one compute node is used.
<code>#PBS -l place=scatter</code>	If more than one compute node are used.
<code>#PBS -l walltime=...</code>	This directive specifies the maximum wall-clock time (real time) that a job should take.
<code>#PBS -j oe</code>	To put standard error and standard out to the same device.
<code>#PBS -N</code>	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 (note for advanced users): ICON employs both distributed memory parallelization and shared memory parallelization, i.e. a “hybrid parallelization”. Only the former type actually performs a decomposition of the domain data, using the de-facto standard MPI. The shared memory parallelization, on the other hand, uses OpenMP directives in the source code. In fact, nearly all DO loops that iterate over grid cells are preceded by OpenMP directives. For reasons of cache efficiency the DO loops over grid cells, edges, and vertices are organized in two nested loops: “jb loops” and “jc loops”¹. Here the outer loop (“jb”) is parallelized with OpenMP.

There is no straight-forward way to determine the optimal hybrid setup, except for the extreme cases: If only a single node is used, then the global memory facilitates a pure OpenMP parallelization. Usually, this setup is only feasible for very small simulations. If, on the other hand, each node constitutes a single-core system, a multi-threaded (OpenMP) run would not make much sense, since multiple threads would interfere on this single core. A pure MPI setup would be the best choice then.

In all of the other cases, the parallelization setup depends on the hardware platform and on the simulation size. In practice, 4 threads/MPI task have proven to be a good choice on Intel-based systems. This should be combined with the *hyperthreading* 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 intersocket communication might impede the performance.

Finally, there is one special case: If an ICON run turns out to consume an extraordinarily large amount of memory (which should not be the case for a model with a decent memory scaling), then the user can resort to “investing” more OpenMP threads than it is necessary for the runtime performance. Doing so, each MPI process would have more memory at its disposal.

¹This implementation method is known as *loop tiling*, see also Section 5.3.

4.4.2. 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 4.7).

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.

4.4.3. 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.

4.5. Exercises

In this exercise you will learn how to start ICON from DWD analysis data and how to perform a multi-day forecast with 40 km resolution globally and 20 km resolution 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. 5.1.

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 Linux cluster `lce`!**

Input Data

Note:

The exercises in this section require a number of grid files, external parameters and input data. This data is already available in the directory `case2/input`. Their creation process is explained in Ex. 2.1 and 2.2 (see p. 32).

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. Table 4.1 provides a list of all input files in the input directory (left column) together with the corresponding symbolic links in the experiment directory (right column). Note that, in general, the original names differ from the symbolic link names which need to match the default filename structure expected by the model.

original name		symbolic link name
grid files		
icon_grid_0023_R02B05_R[-grfinfo].nc	→	iconR2B05_DOM00[-grfinfo].nc
icon_grid_0024_R02B06_G[-grfinfo].nc	→	iconR2B06_DOM01[-grfinfo].nc
icon_grid_0028_R02B07_N02[-grfinfo].nc	→	iconR2B07_DOM02[-grfinfo].nc
extpar files		
icon_extpar_0024_R02B06_G_20150805_tiles.nc	→	extpar_DOM01.nc
icon_extpar_0028_R02B07_N02_20150805_tiles.nc	→	extpar_DOM02.nc
first guess/analysis files		
igfff20170111210000-0130R.grb	→	dwdFG_R2B06_DOM01.grb
iefff20170111210000-0130R.grb	→	dwdFG_R2B07_DOM02.grb
igaf20170112000000R.grb	→	dwdANA_R2B06_DOM01.grb
ieaf20170112000000R.grb	→	dwdANA_R2B07_DOM02.grb

Table 4.1.: List of all input files required for Exercises 4.1–4.5. The left column shows the original filenames, as found in the input directory `case2/input`, whereas the right column shows the corresponding symbolic link names in the experiment directory. With the symbolic links we avoid absolute path settings in the ICON namelists.

Starting a Global ICON Forecast from DWD Analysis

In this exercise you will learn how to run ICON in real data mode.

Open the ICON run script `case2/run_ICON_R02B06_dwdini` and prepare the script for running a global **72 hour forecast** on an R2B06 grid with 40 km horizontal resolution 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 4.2.1).
- 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 4.4.1 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`).
- Due to the appropriate choice of the symbolic links in the experiment directory (see Table 4.1), ICON is able to locate the first guess and analysis data sets



EX 4.1

automatically without specifying the namelist variables `dwdfg_filename`, `dwdana_filename`.

However, both are given in the run script for reference, using the keyword nomenclature mentioned in Section 4.2.2.

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?

<code>dwdFG_R2B06.grb</code>		<input type="checkbox"/>
<code>dwdFG_R2B6_DOM01.grb</code>		<input type="checkbox"/>
<code>dwdFG_R9B02_DOM02.grb</code>		<input type="checkbox"/>
<code>dwdfg_R2B06_DOM01.grb</code>		<input type="checkbox"/>

- Specify the name of the external parameter file (`extpar_filename`). Remember to make use of the symbolic link name! Instead of specifying the full name, try to make use of the keyword `idom`.

Settings for the Incremental Analysis Update (IAU) procedure

- Choose the appropriate initialization mode `init_mode` for starting the model from DWD analysis **with** IAU (see Section 4.2.3).
- The Incremental Analysis Update shall be performed for a 3 h time interval, centered around the nominal start date. Please choose `dt_iau` and `dt_shift` appropriately. Revisit your settings by comparing them with Figure 4.2.

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 `case2/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.

Timestepping

This exercise focuses on aspects of the ICON time-stepping scheme, explained in Section 4.1.

- Compute the *dynamics* time step $\Delta\tau$ from the specification of the physics time step Δt (`dtime`) and the number of dynamics substeps `ndyn_substeps`.

– Answer: $\Delta\tau =$ s

- Take a look at Equation (4.1) and calculate an estimate for the maximum dynamic time step which is allowed for the horizontal resolution at hand.

– Answer: $\Delta\tau_{max} =$ s

Now compare this to the time step used: Did we make a reasonable choice?



EX 4.2

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.

Performance assessment using the timer output:

- Open your run script from the previous Exercise 4.1 and enable the ICON routines for performance logging (timers). To do so, follow the instructions in Section 4.4.3.
- Repeat the model run.
- At the end of the model run, a log file is created. It can be found in your base directory `case2`. Scroll to the end of this file. You should find wall clock timer output comparable to that listed in Section 4.4.3. Try to identify

the total run time – Answer: s

the time needed by the radiation module – Answer: s



EX 4.3

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_R02B06_dwdini` named `run_ICON_R02B06_dwdini_fast`. In order to avoid overwriting your old results, replace the output directory name (`EXPDIR`) by `exp02_dwdini_fast`.



EX 4.4

- 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_R02B06_dwdini`) ran the executable in hybrid mode on 25 nodes using 12 MPI tasks/node and 4 OpenMP threads/MPI task with hyperthreading enabled.
 - Your new script (`run_ICON_R02B06_dwdini_fast`) should run the executable in hybrid mode on 50 nodes using 12 MPI tasks/node and 4 OpenMP threads/MPI task with hyperthreading enabled.

You need to adjust both the PBS settings and the `aprun` command. See Section 4.4.3 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> 25 nodes	<code>nh_solve</code> 50 nodes	<code>transport</code> 25 nodes	<code>transport</code> 50 nodes
<input type="text"/> s	<input type="text"/> s	<input type="text"/> s	<input type="text"/> s

What do you think is a more sensible measure of the effective cost: `total min rank` or `total max rank`?

– *Answer:*

- Which speedup did you achieve and what would you expect from “theory”?

– *Answer:* Speedup achieved = $\frac{T_{25nodes}}{T_{50nodes}}$ =

– *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 in Chapter 5.



EX 4.5

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 resolution of 20 km (see Figure 4.3):

- Create a copy of your run script `run_ICON_R02B06_dwdini` named `run_ICON_R02B06N7_dwdini`. Alternatively, you may create a copy of the reference version `reference/case2/run_ICON_R02B06_dwdini`.

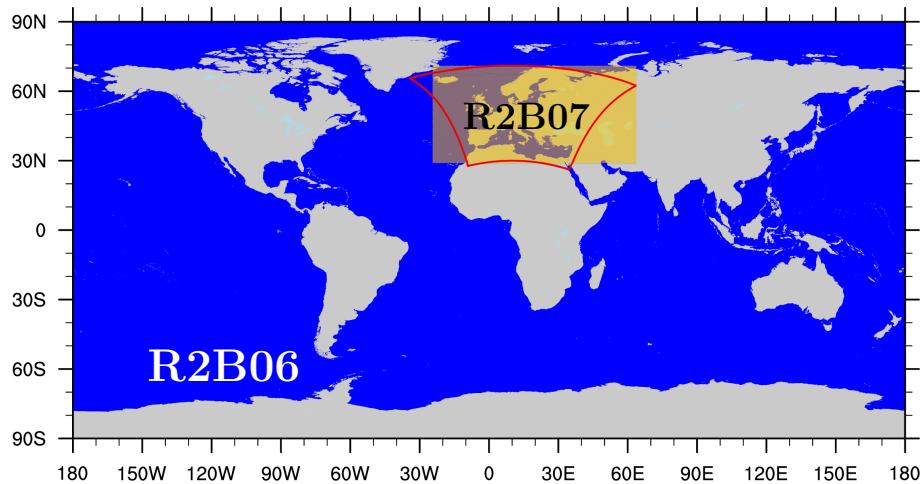


Figure 4.3.: Computational grid with a two-way nested region over Europe (yellow shading). The outline of the COSMO-EU domain (formerly used operationally by DWD) is shown in red for comparison.

- 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. Activate the namelists and fill in the missing parameters. See Section 4.3 for additional details regarding output namelists. Output 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 subdirectory “`lam_forcing`” of your output directory `exp02_dwdini`, using the filename prefix “`forcing`”. If the subdirectory 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 4.2 to see whether your output namelist is correct.

Table 4.2.: Reference output for Ex. 4.5. Structure and content of file `forcing_DOM02_20170114T000000Z.grb`

```

File format : GRIB2
-1 : Institut Source  Ttype  Levels Num  Points Num Dtype : Parameter name
 1 : DWD      unknown instant   60  1    75948  1  P16  : U
 2 : DWD      unknown instant   60  1    75948  1  P16  : V
 3 : DWD      unknown instant   61  2    75948  1  P16  : W
 4 : DWD      unknown instant   60  1    75948  1  P16  : THETA_V
 5 : DWD      unknown instant   60  1    75948  1  P16  : DEN
 6 : DWD      unknown instant   60  1    75948  1  P16  : QV
 7 : DWD      unknown instant   60  1    75948  1  P16  : QC
 8 : DWD      unknown instant   60  1    75948  1  P16  : QI
 9 : DWD      unknown instant   60  1    75948  1  P16  : QR
10 : DWD      unknown instant   60  1    75948  1  P16  : QS
11 : DWD      unknown instant   61  3    75948  1  P16  : HHL

Grid coordinates :
 1 : unstructured          : points=75948
                               grid : number=28 position=1
                               uuid : 982dcc6e-fe2e-11e4-9128-0b03674e713a

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    : 5a98830d-0338-110a-7fd6-d9099a353200
 2 : generalized_height    : levels=61
                               height : 1 to 61 by 1
                               zaxis   : number = 4
                               uuid    : 5a98830d-0338-110a-7fd6-d9099a353200
 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    : 5a98830d-0338-110a-7fd6-d9099a353200

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

```

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 behaviour 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.

- Run the NCL script `case2/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 `case2/output/exp02_dwdini`. Compare the results to Figure 4.4 which shows the corresponding results when ICON is started from IFS analysis.

Which analysis data set leads to an increased spin up/down in this particular case?

native analysis (DWD)

non-native analysis (IFS)

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)?



EX 4.6

Optional: Checking for Bit-Reproducibility

- Compare the model level output of `run_ICON_R02B06_ifsini` with the output that was produced by your modified script `run_ICON_R02B06_ifsini_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.

Hint: For a convenient comparison of ASCII files you may use the `tkdiff` utility.



EX 4.7

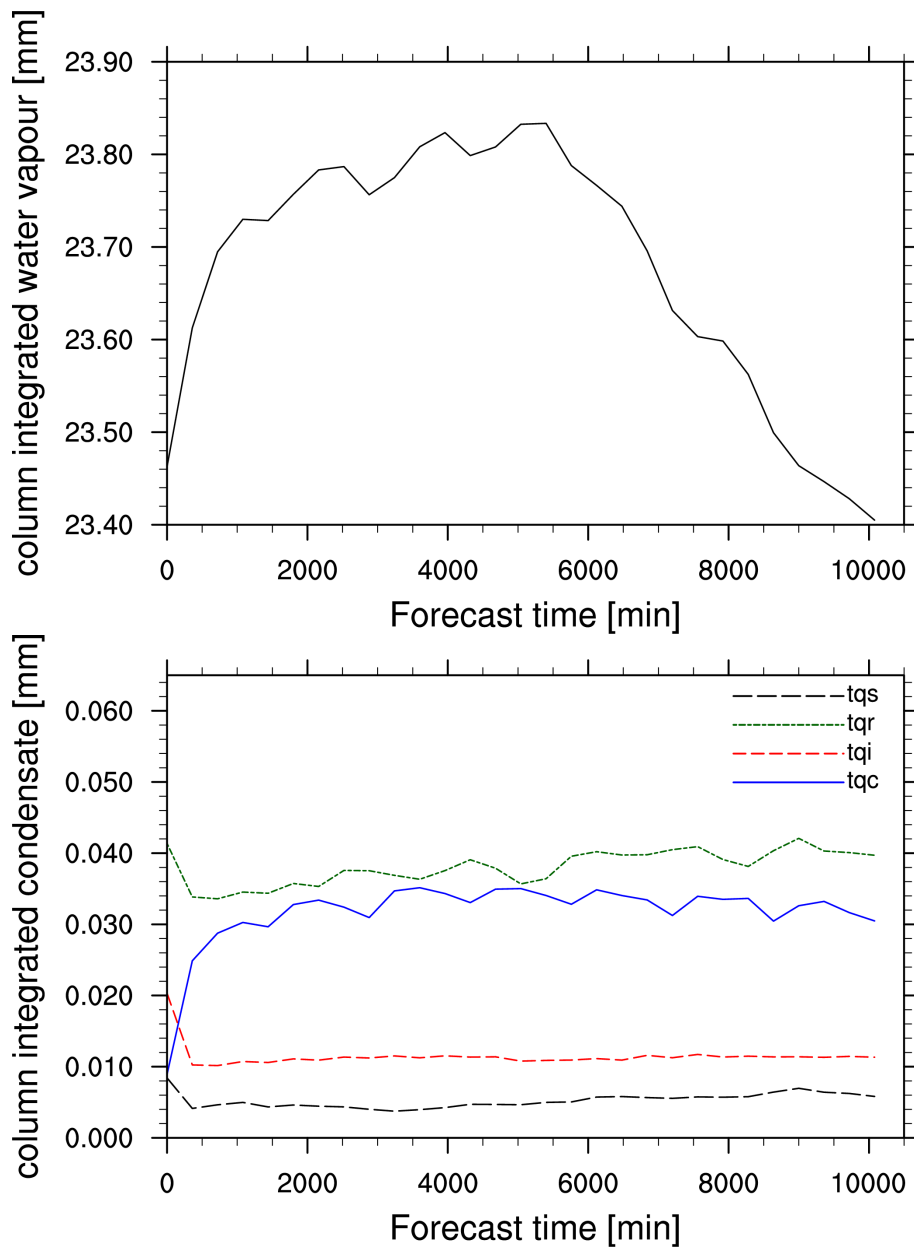


Figure 4.4.: Time series of area averaged column integrated specific moisture $\langle tqv \rangle$ (top) and condensate classes $\langle tqx \rangle$ (bottom) for a 7-day forecast **started from IFS analysis** fields. Start date was 2017-01-12T00:00:00. A spin up in $\langle tqv \rangle$ and initial adjustments in $\langle tqx \rangle$ are clearly visible.

5. Limited Area Mode

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 2.3.4. Based on these prerequisites the exercises in this chapter will explain how to actually set up and run limited area simulations.

Apart from these technical adjustments, a more detailed understanding of ICON's physics-dynamics coupling is a necessary starting point for actually *modifying and extending* the ICON model. We will provide information on this more general subject in the following sections as well. Finally, the reader will be able to implement own model diagnostics.

5.1. Running ICON-LAM

This section provides technical details on the limited area mode, in particular on how to control the read-in of boundary data.

5.1.1. Limited Area Mode vs. Nested Setups

In Section 3.3.2 (see p. 43) 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.

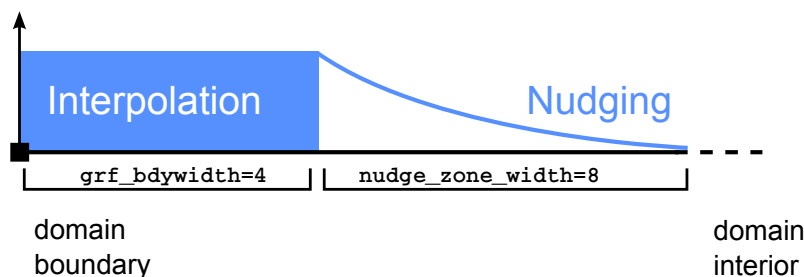
However, there exist some differences between the regional and the one-way nested mode:

¹Here, we do not take the reduced radiation grid feature into account, see Section 6.2. This serves to simplify the discussion at this point.

- ICON-LAM is driven by external boundary data – that’s an obvious difference! During the simulation time loop, the boundary conditions are updated periodically by reading input files. Between two lateral boundary data samples the boundary data is linearly interpolated.
- Updates happen (significantly) less frequently compared to one-way nesting.
- The driving model and the limited area model may run on different computer sites.
- 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 between the nested and the parent domain must match. In contrast to that the limited area mode performs a vertical interpolation of its input 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.

5.1.2. Nudging in the Boundary Zone

Within the boundary zone, the driving boundary data is partly prescribed and partly combined with the prognostic fields of the regional domain by taking a weighted mean of the two.



On the outermost four cell rows (`grf_bdywidth`) the boundary data are simply interpolated onto the domain. In the adjacent nudging zone the prognostic fields are nudged towards the driving boundary data. The nudging weights are reduced with increasing distance from the boundary. The nudge zone width in terms of cell rows can be specified in `nudge_zone_width`. It should at least comprise 8 (better 10) cell rows in order to minimize boundary artefacts.

5.1.3. Initialization Mode

Limited area runs with ICON require new initialization modes, in addition to those defined in Section 4.2.1. In these modes the read-in process will be followed by a vertical interpolation of the input fields.

init_mode (namelist `initicon_nml`, integer value)

```
init_mode = 4  start limited area run from COSMO-DE data
init_mode = 7  start limited area run from ICON data
```

nlev_latbc (namelist `limarea_nml`, integer value)

This namelist parameter specifies the number of vertical levels in the boundary data.

When we do not make use of additional analysis information, we need to set this via a namelist option:

lread_ana (namelist `initicon_nml`, logical value)

By default, this namelist parameter is set to `.TRUE.` If `.FALSE.`, ICON is started from first guess only and an analysis file is not required. The filename of the first guess file is specified via the `dwdfg_filename` namelist option, see Section 4.2.2.

Boundary data is read in 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.

5.1.4. 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 2.3.4).

Filenames: `latbc_filename`, `latbc_path` (string parameters, `limarea_nml`)

The filenames have the following form:

```
"prepiconRnrootBjlev_yyyyymmddhh.nc"
```

Here, `nroot` and `jlev` denote the grid's root subdivision and bisection level (see Section 2.1.1). The tokens `yyyy`, `mm`, `dd`, and `hh` must be replaced by year, month, day, and hour. This naming scheme can even 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 the DWD input fields. Since often field names that are used, e.g., in the GRIB2 input files do not coincide with the field names that are internally used by the ICON model, 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.

If no `latbc_varnames_map_file` is specified, then it is assumed that all required fields can be identified in the input files by their ICON-internal names.

Boundary grid: `latbc.boundary_grid` (**namelist** `limarea_nml`, **string**)

As it has been explained in Section 2.3.4, the lateral boundary data is defined on an auxiliary grid, which contains only the cells of the boundary zone.

The filename of this grid file is specified with this namelist parameter.

5.1.5. 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 process 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 prefetching processors, can be zero or one.

Enabling the prefetching mode is **mandatory** for the described LAM setup.

5.2. ICON Physics in a Nutshell**5.2.1. Overview**

Table 5.1 contains a summary of physical parameterizations available in ICON (NWP-mode).

5.2.2. 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 4.1.

Fast-physics processes are calculated at every physics time step and are treated with time splitting (also known as sequential 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 → surface transfer scheme → land-surface scheme → boundary-layer / turbulent vertical diffusion scheme → 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'

Process	Scheme	Settings
Radiation	RRTM (<u>R</u> apid <u>R</u> adiative <u>T</u> ransfer <u>M</u> odel) Mlawer et al. (1997), Barker et al. (2003)	<code>inwp_radiation=1</code>
	PSRAD Pincus and Stevens (2013)	<code>inwp_radiation=3</code>
Non-orographic gravity wave drag	Wave dissipation at critical level Orr et al. (2010)	<code>inwp_gwd=1</code>
Sub-grid scale orographic drag	Lott and Miller scheme Lott and Miller (1997)	<code>inwp_sso=1</code>
Cloud cover	Diagnostic PDF <i>M. Köhler et al. (DWD)</i>	<code>inwp_cldcover=1</code>
	All-or-nothing scheme (grid-scale clouds)	<code>inwp_cldcover=5</code>
Microphysics	Single-moment scheme Doms et al. (2011), Seifert (2008)	<code>inwp_gscp=1</code>
	Double-moment scheme Seifert and Beheng (2006)	<code>inwp_gscp=4</code>
Convection	Mass-flux shallow and deep Tiedtke (1989), Bechtold et al. (2008)	<code>inwp_convection=1</code>
Turbulent transfer	Prognostic TKE (COSMO) Raschendorfer (2001)	<code>inwp_turb=1</code>
	EDMF-DUALM (Eddy-Diffusivity/Mass-Flux) Neggens et al. (2009)	<code>inwp_turb=3</code>
	3D Smagorinsky diffusion (for LES)	<code>inwp_turb=5</code>
Land	Tiled TERRA Schrodin and Heise (2002)	<code>inwp_surface=1</code>
	Flake: Mironov (2008)	<code>llake=.TRUE.</code>
	Sea-ice: Mironov et al. (2012)	<code>lseaiice=.TRUE.</code>

Table 5.1.: Summary of ICON’s physics parameterizations, together with the related namelist settings (namelist `nwp_phy_nml`). Note: Since the JSBACH component is not available in NWP mode, it has been excluded from this list.

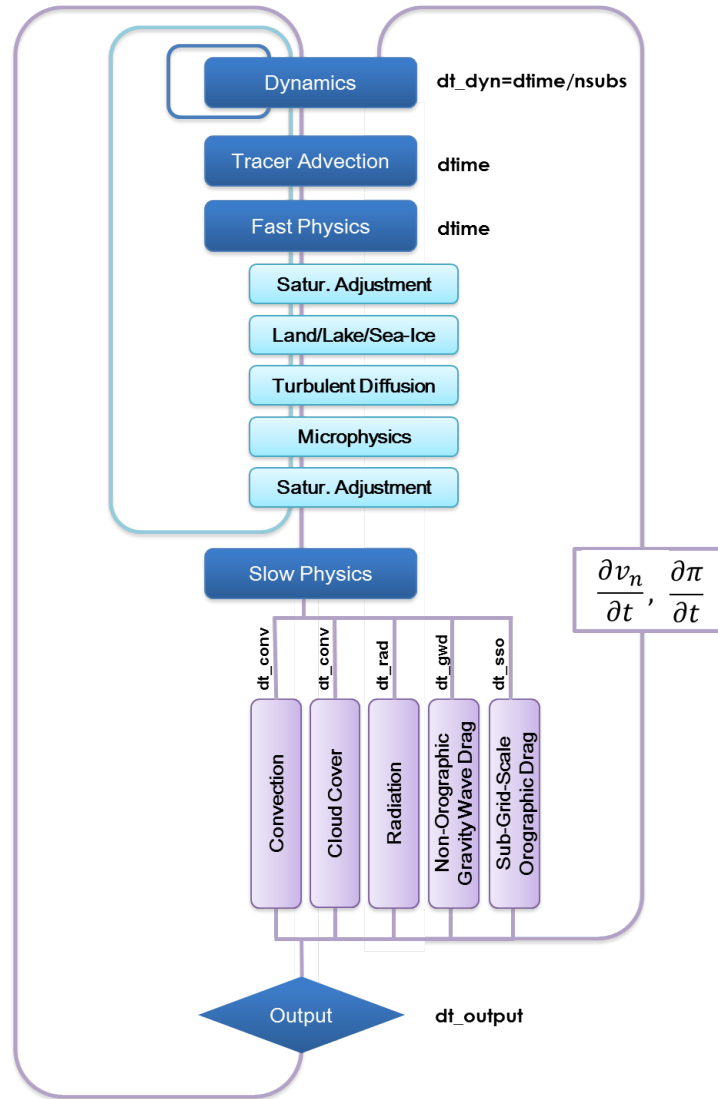


Figure 5.1.: Coupling of the dynamical core and the NWP physics package. Processes declared as fast (slow) are treated in a time-split (process-split) manner.

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 5.1.

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, 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 5.1). Since ICON solves a

prognostic equation for π rather than T , the temperature tendencies are converted into tendencies of the Exner function, beforehand.

5.2.3. Isobaric vs. Isochoric Coupling Strategies

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

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

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

5.2.4. Cloud Microphysics

In the exercises at the end of this chapter, 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*.

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

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 one 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 from the mass content. Such schemes are called *double-moment schemes*, because both, mass content and number concentration, are statistical moments of the particles size distribution.

ICON does also provide a double-moment microphysics scheme (`inwp_gscp=4`), 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.

5.3. 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. Moreover, its scope would not be limited to LAM applications. Here, we try to keep things as simple and short as possible with a view to the subsequent exercises.

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 4.3:

```
&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 explained. For reasons of cache efficiency 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`.

Note: Three-dimensional fields 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 neighbouring sub-domains (so-called *halo cells*). Often it is not necessary to loop over these points twice.

An 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.

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 5.1: 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.

5.4. Exercises

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. 4.5. The preprocessing of this data is explained in Ex. 2.4 (see p. 33).

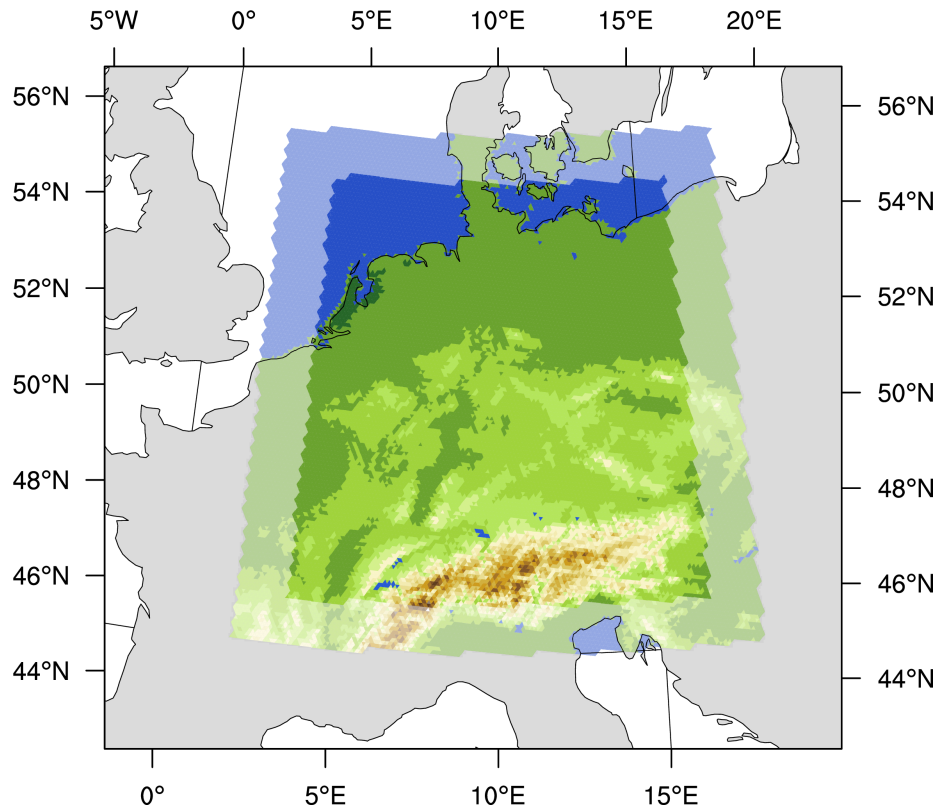


Figure 5.2.: Illustration of the local grid used in Exercises 5.1–5.3. The horizontal resolution is ≈ 6.5 km (which corresponds to R3B8 in ICON nomenclature). The boundary region is highlighted, where nudging towards the driving data is performed. The driving data for this test case has been created in Ex. 2.4.

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. 2.4.

Open the run script `case3/run_ICON_R3B08_1am` and prepare it for running a **48 hour forecast** on a **limited area grid over Germany** (see Figure 5.2). As for the global run, the start date is

2017-01-12T00:00:00 , i.e. January 12, 2017.



EX 5.1

The run script is geared up insofar as all softlinks 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 5.1.3).

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 5.1.3).

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 5.1.4).
- 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.

- For the boundary data, set the number of vertical levels in `nlev_latbc`. 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 `case3/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 5.1.3 for the namelist parameter. Write down your chosen value:

– Answer:

- 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 `case3/plot_cross_section.ncl` and/or make use of `ncview`.



EX 5.2

Implementing New Diagnostics

In this exercise we will implement two new diagnostic fields (2D variables):

RHI: Saturation 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 5.3.

Step-by-step checklist:

- Open the module `mo_nonhydro_types` (subdirectory `src/atm_dyn_iconam`).

Insert two new 2D variable pointers in `TYPE(t_nh_diag): RHI, QI_MAX`

Got it? Did it!

- Open the module `mo_nonhydro_state` (subdirectory `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!

- Open the module `mo_nh_stepping` (subdirectory `src/atm_dyn_iconam`).



EX 5.3

Create an (empty) subroutine `calculate_diagnostics` and place a call to this subroutine immediately before the call to the output routine.

Fill your subroutine with a 2D loop over all grid points, calculate the two new quantities.

Two hints:

- The module `mo_util_phys` (subdirectory `src/atm_phy_nwp`) may offer some help with respect to RHI.
- The 3D tracer field QI for domain “jg” can be accessed via `p_nh_state%prog(nnow_rcf(jg))%tracer(:, :, :, iqi)`.
- Open the namelist of the previous test case 5.1 and insert the new fields in the output specification.

Compile and run the model. Visualize the results (`ncview`).

Got it?
 Did it!

Got it?
 Did it!

6. ICON NWP Mode: Further Features

This chapter provides details on some advanced technical aspects of the ICON model. The following topics are covered: First, we briefly describe the settings for a reduced moist physics computation and explain an option for coarse-resolution radiation grids. Then we discuss the possibilities to write model output on regular grids and different sets of vertical levels. Finally, before concluding this chapter with some exercises, we give a short overview of the “defensive I/O”, that is the write-out and read-in of the model state in order to resume the model run, say, after a previous crash.

6.1. Reduced Model Top for Moist Physics

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

Another means for improving the efficiency of ICON is depicted in Figure 6.1. The switch `htop_moist_proc` allows to switch off moist physics completely above a certain height. Moist physics include saturation adjustment, grid scale microphysics, convection, cloud cover diagnostic, as well as the transport of all water species but moisture q_v . Of course, moist processes should only be switched off well above the tropopause. The default setting is `htop_moist_proc=22500 m`.

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

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 $80 \frac{\text{m}}{\text{s}}$. In real case applications, this mostly happens in the stratosphere and mesosphere. The recommendation for Δt given in Section 4.1 then exceeds the numerical stability range of the horizontal transport scheme. To stabilize the integration without the need to reduce the time step globally, transport schemes with and without internal substepping can be combined. The switch `hbot_qvsubstep` 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).

If moist physics are switched off above 22.5 km (default for NWP applications), internal substepping only needs to be applied to specific humidity q_v , since the advection of all

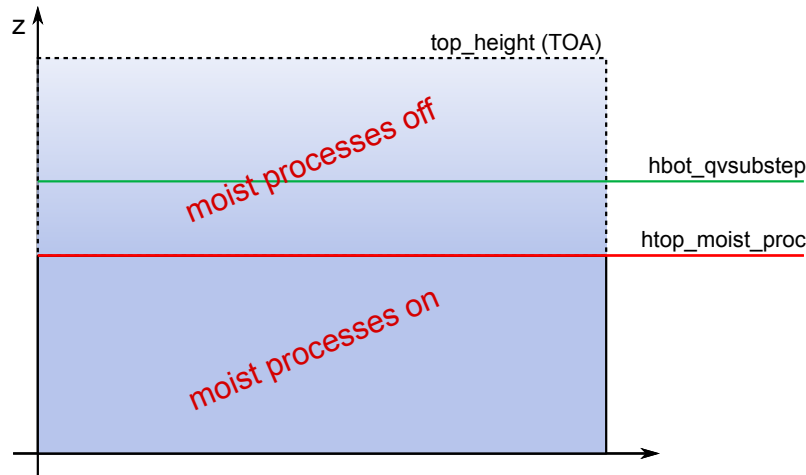


Figure 6.1.: Moist physics are switched off above `htop_moist_proc`, while tracer substepping is switched on above `hbot_qvsubstep`. (Remark: `hbot_qvsubstep` is allowed to be lower than `htop_moist_proc`)

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 8).

6.2. Reduced Radiation Grid

In real case simulations, radiation is one of the most time consuming physical processes. It is therefore very desirable to reduce the computational burden without degrading the results significantly. One possibility is to use a coarser grid for radiation than for dynamics.

The implementation is schematically depicted in Figure 6.2:

- Step 1.* Radiative transfer computations are usually performed every 30 minutes. Before doing so, all input fields required by the radiation scheme are upscaled to the next coarser grid level.
- Step 2.* Then the radiative transfer computations are performed and the resulting short wave transmissivities τ^{SW} and longwave fluxes F^{LW} are scaled down to the full grid.
- Step 3.* In a last step we apply empirical corrections to those fields 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)

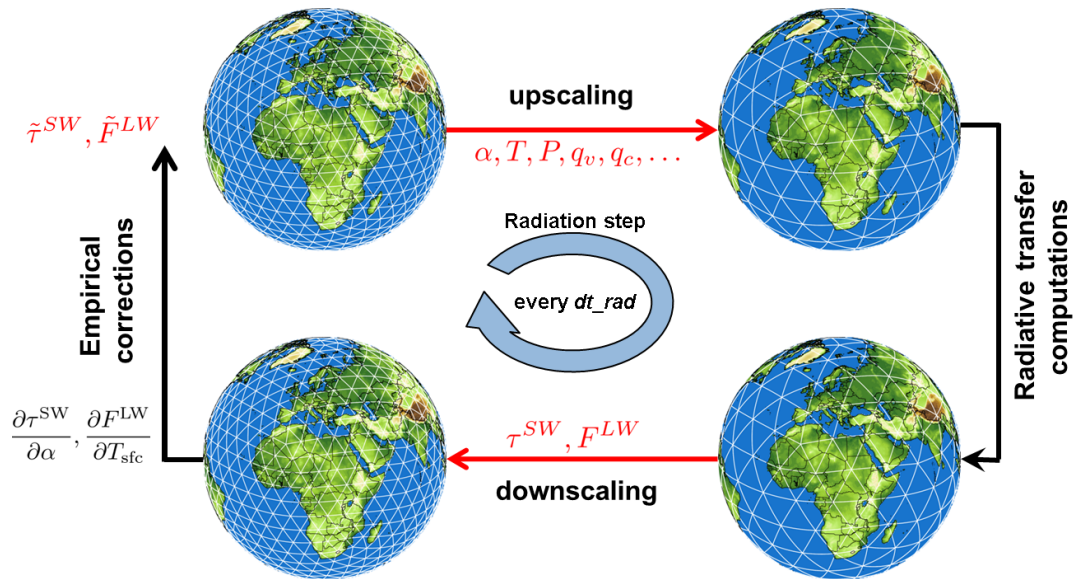


Figure 6.2.: Schematic showing how radiation is computed on a reduced (coarser) grid.

`radiation_grid_filename` (**namelist** `grid_nml`, **string** `parameter`)

Filename of the grid to be used for the radiation model. An empty string is required, if radiation is computed on the full (non-reduced) grid.

Note that running radiation on a reduced grid is the standard setting for operational runs at DWD. Using the reduced radiation grid is also possible for the limited area mode ICON-LAM. In this case, both the computational grid and the reduced radiation grid are regional grids. Make sure to create the latter during the grid generation process by setting `dom(:)%lwrite_parent = .TRUE.`, see Section 2.1.2.

6.3. Internal Post-Processing

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. 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 4.3, 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 `h_levels`, `pl_varlist` defines variables on `p_levels` and `il_varlist` specifies output on `i_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.

6.4. 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.

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 4.4) 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.

6.5. Exercises

The necessary grids, external parameters and IFS input data on a regular grid are already available in the directory `case4/input`. **The missing interpolation step, which maps the IFS data onto the triangular ICON grid is explained in Ex. 2.3.**

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`!**

Starting from IFS analysis

In this lesson you will learn how to start a real-case simulation from IFS data.

Open the ICON run script `case4/run_ICON_R02B06_ifsini` and prepare the script for running a global 48 hour forecast with 40 km horizontal resolution without nest. The start date is

2017-01-12T00:00:00 , i.e. January 12, 2017.

- Fill in the name of the ICON model binary (including the path) and several missing namelist parameters. I.e. set `ini_datetime_string`, `end_datetime_string`, `ltestcase`, `ldynamics`, `ltransport`, `iforcing`, `init_mode`, `itopo` as well as the filenames for the initial data and external parameters (`ifs2icon_filename`, `extpar_filename`). See Section 4.2.1 and 4.2.4 for specific settings.
- For better runtime performance, switch on *asynchronous* output by setting the number of dedicated I/O processors (`num_io_procs`) to the number of active output namelists. See Section 4.4.1 for more details on the asynchronous output module.
- Submit the job to the Cray XC 40.
- After the job has finished, inspect the model output:



EX 6.1

- Take a look at the output files in `case4/output`. You should find two files named `NWP_...` Both should contain 6-hourly model level output up to 48 hours. However, 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.
- Visualize the 2 m temperature, integrated water vapour, and total precipitation using the `ncview` utility. If you like, you can look into other fields as well.

Running ICON with Different Output Products

In this lesson you will learn how to adjust the model output according to your needs.



EX 6.2

ICON forecast starting from IFS analysis.

- Create a copy of your run script `run_ICON_R02B06_ifsini` from Ex. 6.1 and name it `run_ICON_R02B06_ifsini_output`. In order to avoid overwriting your old results, replace the output directory name (`EXPDIR`) by `exp02_ifsini_output`.
- The run script contains a tentative commented-out output namelist. Activate the name list and fill in the missing parameters. See Section 4.3 for additional details regarding output namelists. Deactivate (comment out) all other output namelists. Output should be written
 - in NetCDF format
 - 6-hourly from the start until simulation end (48 hours)
 - with all output steps in one file
 - on the native (triangular) grid
 - including grid information
 - using the filename prefix `NWP_DWD`
 - using a relative time axis (forecast mode)
 - containing model level output for `qv`, `qc`, `qi`, `qr`, `qs`, 2m temperature, total precipitation, and mean sea level pressure. See Appendix C for variable names and description.
- Add a second output namelist which is identical to the previous one, except for the following changes:
 - 6-hourly output should become active after 24 hours until the end of the model run.

- Output should be created on a regular lat-lon grid with a resolution of 0.75° in both zonal and meridional direction.
- Set the filename prefix to `NWP_DWD_1onlat`.
- Output of qv , qc , qi , qr , qs should be on height levels instead of model levels. In addition, write out total precipitation, 2 m temperature, and mean sea level pressure, again.
- Define 13 height levels ranging from 1 km up to 25 km with 2 km intervals. See Section 6.3 for help.
- Let the model write a restart file every 36 h. For this you have to adapt `dt_checkpoint`. See Section 6.4 for additional details.
- Submit the job to the Cray XC 40.
- Have a short break and relax :-)
- Check correctness of your output files:
 - You should find three output files in your output directory. Apply the command `cdo sinfov data-file.nc > data-file.sinfov` to each of your output files. Compare with the reference output in Tables 6.1–6.3, to see whether your output namelists are correct.
- Visualization:
 - Apply the NCL script `case4/zonal_mean.ncl` to your lat-lon output field. The script plots vertical cross sections of zonally averaged qv as well as $qc + qi + qr + qs$ after 48 h and contour plots of accumulated precipitation, sea-level pressure and 2 m temperature. Compare it to Figure 6.3.
- The following fact should become visible from Figure 6.3:

Moist quantities except qv are approximately 0 above the tropopause. Since the NWP microphysics are not capable of dealing with polar stratospheric clouds (PSCs) anyway, one can save computing time by completely switching off moist physics in the upper stratosphere and mesosphere. The height above which moist physics are switched off can be set with `htop_moist_proc`. See Section 6.1 for more information. In your run, this switch was set to a height of 22500 m, while for Figure 6.3 the parameter `htop_moist_proc` was set to a value larger than the model top height. Based on your comparison, would you say that a moist physics maximum height `htop_moist_proc=22500 m` is a meaningful choice?

Table 6.1.: Structure and content of file NWP_DWD_DOM01_ML_0001.nc

```

File format : netCDF2
-1 : Institut Source   Ttype   Levels Num   Points Num Dtype : Parameter name
  1 : unknown  git@git.mpimet instant    90   1   327680  1  F32 : qv
  2 : unknown  git@git.mpimet instant    90   1   327680  1  F32 : qc
  3 : unknown  git@git.mpimet instant    90   1   327680  1  F32 : qi
  4 : unknown  git@git.mpimet instant    90   1   327680  1  F32 : qr
  5 : unknown  git@git.mpimet instant    90   1   327680  1  F32 : qs
  6 : unknown  git@git.mpimet instant     1   2   327680  1  F32 : t_2m
  7 : unknown  git@git.mpimet instant     1   3   327680  1  F32 : tot_prec
  8 : unknown  git@git.mpimet instant     1   3   327680  1  F32 : pres_msl

Grid coordinates :
  1 : unstructured      : points=327680  nvertex=3
                        grid : number=24  position=1
                        clon  : -3.14159 to 3.14159 radian
                        clat  : -1.56615 to 1.56615 radian
                        available : cellbounds
                        uuid   : 9b0b03ca-18c4-11e4-9318-776f158edd08

Vertical coordinates :
  1 : generalized_height : levels=90
                        height : 1 to 90 by 1
                        bounds : 1-2 to 90-91 by 1
                        uuid   : acbc56ad-2658-bd10-7ce8-cb2a4afd2c80
  2 : height             : levels=1
                        height_2 : 2 m
  3 : surface            : levels=1

Time coordinate : 9 steps
  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
  2017-01-12 00:00:00  2017-01-12 06:00:00  2017-01-12 12:00:00  2017-01-12 18:00:00
  2017-01-13 00:00:00  2017-01-13 06:00:00  2017-01-13 12:00:00  2017-01-13 18:00:00
  2017-01-14 00:00:00

```

Table 6.2.: Structure and content of file NWP_DWD_lonlat_DOM01_ML_0001.nc

```

File format : netCDF2
-1 : Institut Source   Ttype   Levels Num   Points Num Dtype : Parameter name
  1 : unknown  git@git.mpimet instant     1   1   115680  1  F32 : t_2m
  2 : unknown  git@git.mpimet instant     1   2   115680  1  F32 : tot_prec
  3 : unknown  git@git.mpimet instant     1   2   115680  1  F32 : pres_msl

Grid coordinates :
  1 : lonlat            : points=115680 (480x241)
                        lon   : 0 to 359.25 by 0.75 degrees_east  circular
                        lat   : -90 to 90 by 0.75 degrees_north

Vertical coordinates :
  1 : height            : levels=1
                        height : 2 m
  2 : surface            : levels=1

Time coordinate : 5 steps
  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
  2017-01-13 00:00:00  2017-01-13 06:00:00  2017-01-13 12:00:00  2017-01-13 18:00:00
  2017-01-14 00:00:00

```

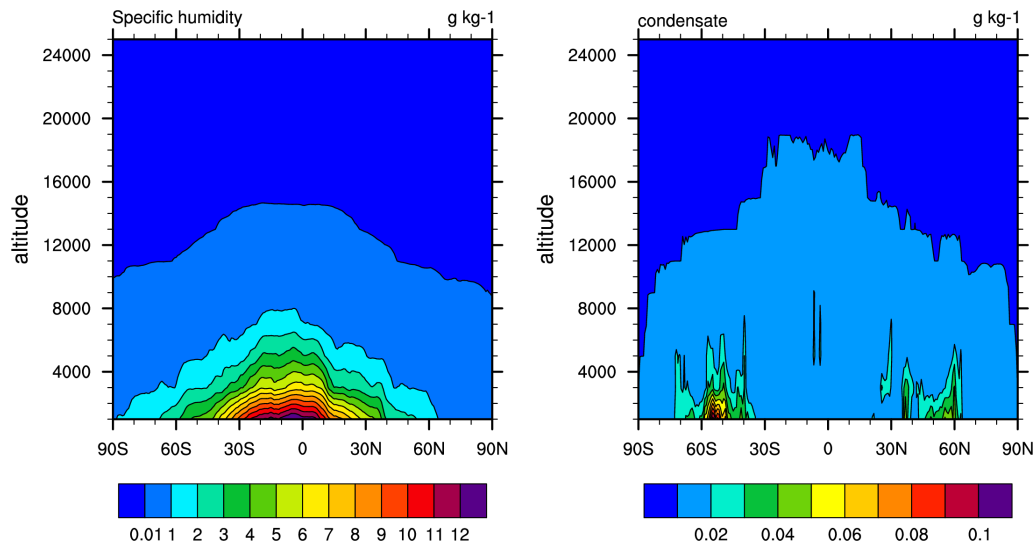


Figure 6.3.: Zonal average of specific humidity q_v and total condensate $q_c + q_i + q_r + q_s$ at 2017-01-14T00:00:00 (i.e. 48 h forecast). Note that the Y-axis starts at a height of 1000 m. For this reference run, `htop_moist_proc=100 km` was chosen, i.e. moist physics were computed up to the model top (75 km).

Restarting a Simulation

This exercise practices the restart of an ICON simulation.

Exercise 6.2 did not contain the sea-ice height `h_ice`. Fortunately, the exercise has produced a checkpoint file from which we can restart the simulation.

- Restart the ICON model by setting the namelist parameter `lrestart` as explained in Section 6.4 and do the following changes:
 - The simulation should end after 48h in total (2017-01-14T00:00:00). This way we can compare the output directly to the results from Exercise 6.2. If you have specified the simulation length via the parameter `nsteps` in namelist `run_nml`, instead of `end_datetime_string`, you need to adapt it accordingly.
 - For the resumed run, specify the sea-ice height `h_ice` as an additional output variable on the native ICON grid.
- After the restarted run has completed, use the `cdo infov` command (see Section 7.1.2 for details) to get minimum, maximum and average value of the sea-ice height `h_ice` after 48 h in the output data.

H_ICE: **MAX** **MIN** **AVG**
 m m m

- Execute the NCL script `case4/sea_ice_plot.ncl` to create a polar plot.



EX 6.3

The following fact should become visible from the polar plot: In contrast to the DWD analysis, the IFS analysis used in this chapter does not offer the sea-ice height as an initialization field. Therefore, when starting from IFS analysis, the sea ice height is always set to a constant value of 1 m. Figure 6.4 shows the sea ice height after 48 h for both the IFS- and DWD-based initialization for reference.

- If time permits, compare the model output for 2017-01-14T00:00:00 (after 48 hours) that you have created in Exercise 6.2 with the corresponding fields of the restarted simulation. Again, you can use the `cdo infov` command for this purpose. There should not be any notable differences between the output fields, meaning that a restarted run will produce bit-identical results.

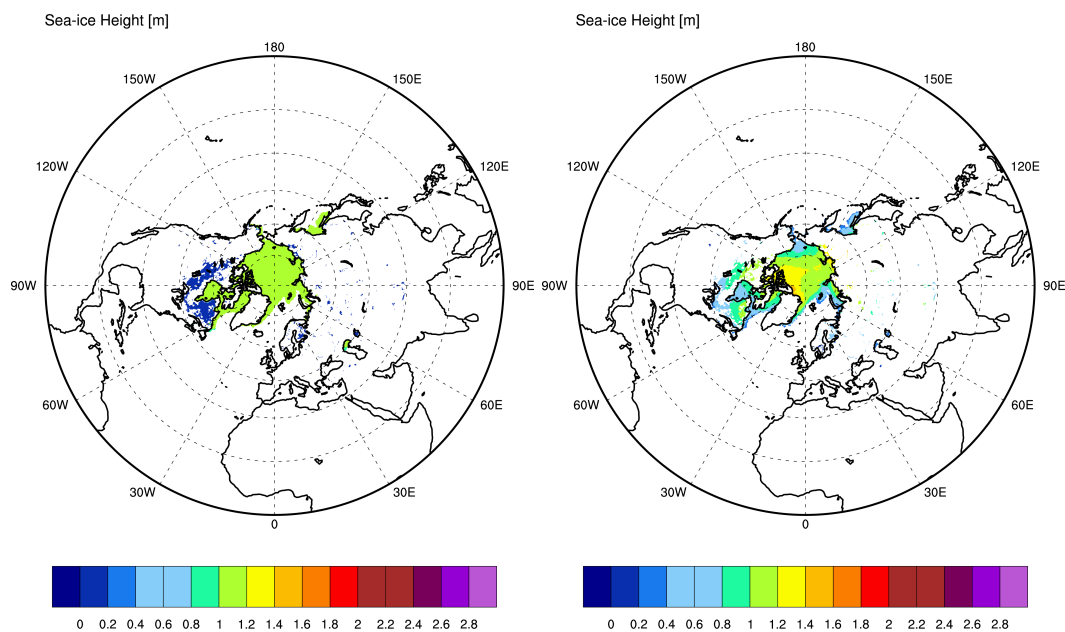


Figure 6.4.: Sea-ice height on the northern hemisphere after 48h (simulation time 2017-01-14T00:00:00), *Left*: Simulation started from IFS analysis as in Exercise 6.3; *Right*: Simulation started from DWD analysis. Sea-ice height is not offered by the IFS analysis. Thus, on the left, the sea-ice was initialized with a constant height of 1 m.

Reduced Grid for Radiation

In this exercise you will learn how to control the reduced (coarser) grid for radiation.

Switching off the reduced grid:

- Create another copy from your run script `run_ICON_R02B06_ifsini` from Ex. 6.1 and name it `run_ICON_R02B06_ifsini_rg`.



EX 6.4

- Open the file `run_ICON_R02B06_ifsini_rg`.
 - Change the name of the output directory (`EXPDIR`) to `exp02_ifsini_rg`. This avoids overwriting your old results.
 - Switch off the reduced radiation grid. This is controlled by the namelist switches `lredgrid_phys` and `radiation_grid_filename`. See Section 6.2 for help.
 - Submit the job script.
 - Have a short break and relax :-)
- After the job has finished, compare the timers for radiation of your previous run (`run_ICON_R02B06_ifsini`) and the new run (`run_ICON_R02B06_ifsini_rg`). What is the speedup of the radiation module? Which speedup would you expect from “theory”?
 - *Answer:* Speedup achieved = $\frac{T_{\text{nonrg}}}{T_{\text{rg}}}$
 - *Answer:* Speedup expected =
- Comparing the quality of results with and without reduced radiation grid is beyond the scope of this tutorial. To give you a rough impression that running forecasts with a reduced radiation grid is a valid choice, we have added Figure 6.5. It shows verification results (BIAS and RMSE) for 850 hPa temperature on the northern hemisphere in January with and without reduced radiation grid.

Table 6.3.: Structure and content of file NWP_DWD_lonlat_DOM01_HL_0001.nc

```

File format : netCDF2
-1 : Institut Source   Ttype   Levels Num   Points Num Dtype : Parameter name
  1 : unknown  git@git.mpimet instant    13    1   115680   1  F32  :  qv
  2 : unknown  git@git.mpimet instant    13    1   115680   1  F32  :  qc
  3 : unknown  git@git.mpimet instant    13    1   115680   1  F32  :  qi
  4 : unknown  git@git.mpimet instant    13    1   115680   1  F32  :  qr
  5 : unknown  git@git.mpimet instant    13    1   115680   1  F32  :  qs
Grid coordinates :
  1 : lonlat           : points=115680 (480x241)
                        lon : 0 to 359.25 by 0.75 degrees_east  circular
                        lat : -90 to 90 by 0.75 degrees_north
Vertical coordinates :
  1 : generic          : levels=13
                        alt : 25000 to 1000 m
Time coordinate : 5 steps
  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
  2017-01-13 00:00:00  2017-01-13 06:00:00  2017-01-13 12:00:00  2017-01-13 18:00:00
  2017-01-14 00:00:00

```

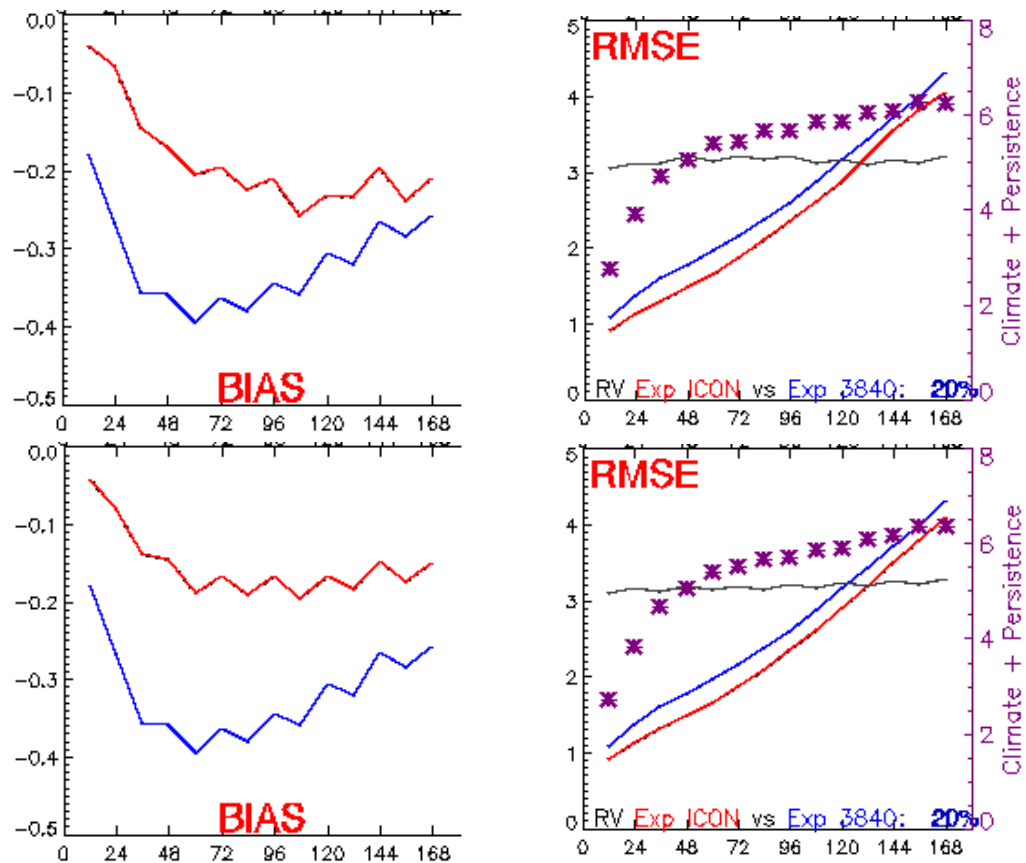


Figure 6.5.: Verification results (BIAS and RMSE) for 850 hPa temperature on the northern hemisphere for January 2012 for the ICON model and DWD's former global model GME. *Upper panel:* ICON with full radiation grid. *Lower panel:* ICON with reduced radiation grid. ICON (R2B6L90) is shown in red, while the GME with 40 km horizontal resolution is shown in blue for reference. In terms of RMSE, ICON results with and without reduced radiation grid are barely indistinguishable.

7. 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 7.3 contains some examples how to visualize NetCDF data sets without the need of an additional regridding.

7.1. 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.

7.1.1. `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)

7.1.2. CDO – Climate Data Operators

The CDO (Climate Data Operators) are a collection of command-line operators to manipulate and analyse 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.zmaw.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.

7.2. Plotting Data Sets on Regular Grids with 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!

7.3. Plotting Data Sets on the Triangular Grid

7.3.1. 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 7.1):

```

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 7.3.2 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 subdirectory `test_cases/casexx` together with the model run scripts.

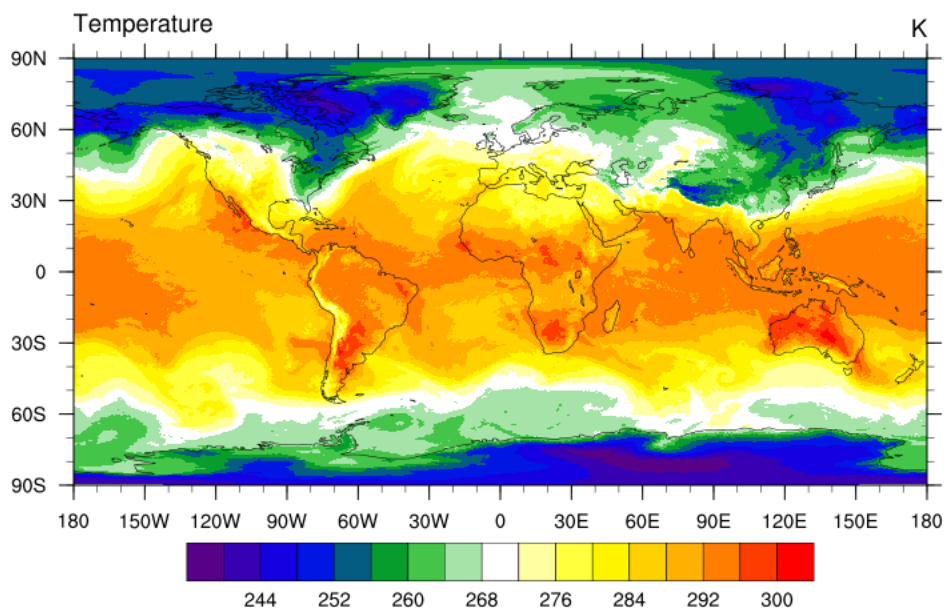


Figure 7.1.: ICON temperature field on a specific model level produced with the above NCL script.

7.3.2. 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"

```

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 13). 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"
```

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]`:

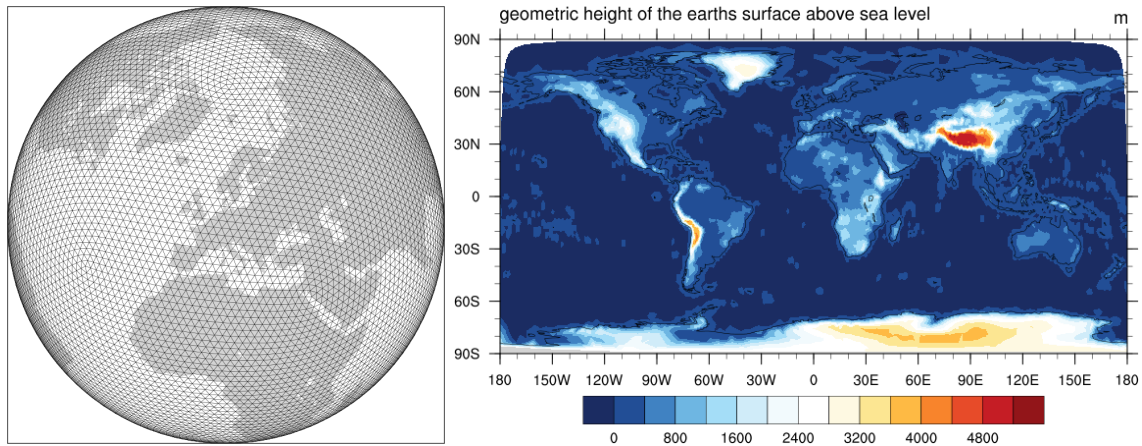


Figure 7.2.: The two plots generated by the NCL example script in Section 7.3.2.

```
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)
```

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. 7.2 (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?

7.3.3. 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 7.3.

```

#!/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

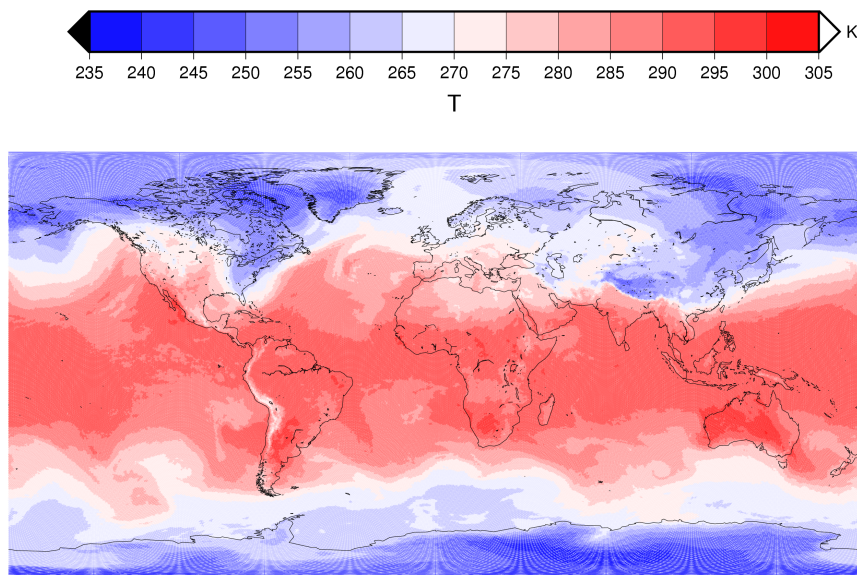
# draw triangle and take fill color from colors.cpt
psxy ${INAME} \
-Rd -Jq0/1:190000000 -Ccolors.cpt -X3.2 -Y4. -K > ${ONAME}

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

# plot colorbar
psscale -D11c/14c/18c/1.0ch -Ccolors.cpt -E -B:"T":/K: -U -O>> $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 7.3.: ICON temperature field on a specific model level produced with the above GMT script.

8. 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).

8.1. 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.

8.2. 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`, `mecicon`, `tixi`).

The Directory `io`

The `io` directory contains input and output routines.

The Directory `mozart_init`

The `mozart_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).

8.3. 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 or Zängl et al. (2014) in order to compile ICON successfully. First, the `ART-v<X>.<YY>.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-v<X>.<YY>` 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
```

8.4. Configuration of an ART Job

8.4.1. 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 8.1.

Table 8.1.: 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.

8.4.2. 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 8.2. Namelist parameters for ART input are listed within Table 8.3. Namelist parameters related to atmospheric chemistry are listed within Table 8.4. Namelist parameters related to aerosol physics are listed within Table 8.5.

Table 8.2.: 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_ntracer</code>	0	Number of transported ART tracers. This number is automatically added to the ICON variable <code>ntracer</code> . It has to be equal to the number of tracers listed in your XML files specified by <code>cart_chemistry.xml</code> , <code>cart_aerosol.xml</code> and <code>cart_passive.xml</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 controled via <code>cart_pntSrc.xml</code> . See also Section 8.4.5.
<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 8.3.: 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 8.4.3.

<code>cart_chemistry_xml</code>	''	Path and file name to the XML file for specifying chemical tracer. See also Section 8.4.3.
<code>cart_aerosol_xml</code>	''	Path and file name to the XML file for specifying aerosol tracer. See also Section 8.4.3.
<code>cart_passive_xml</code>	''	Path and file name to the XML file for specifying passive tracer. See also Section 8.4.3.
<code>cart_modes_xml</code>	''	Path and file name to the XML file for specifying aerosol modes. See also Section 8.4.4.
<code>cart_pntSrc_xml</code>	''	Path and file name to the XML file for specifying point source emissions. See also Section 8.4.5.

Table 8.4.: 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 8.5.: 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.

8.4.3. 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 Zängl et al. (2014)

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**.

8.4.4. 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.

8.4.5. 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 8.4.2).

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.

8.4.6. 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 8.1 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- Eyjafjcell Iceland-S Historical 63.63 N X 19.62 W 1666 Stratovolcano D3 S0
```

Figure 8.1.: 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.

8.5. 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 8.6.

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 8.6.: 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

8.6. Exercises

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 8.3.

You will find the ART code at:

```
/e/uwork/trng024/packages/ART-v2.1.00.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 8.3. 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

In this exercise, you will learn to add your own tracers with emissions from point sources. This exercise makes use of the same setup for ICON-LAM as you have already used in the Ex. 5.1 of Chapter 5. You are free to choose the tracer(s) to transport and the location of the point source(s).

**EX 8.1**

In order to perform the simulation, you have to do the following steps:

- You will find a copy of the runscript used in the previous ICON-LAM test case inside `$ARTDIR/runctrl_examples/run_scripts` called `exp.art.trng17.case1.pntSrc`.
- Edit the run script according to the namelist parameters you find in Section 8.4. 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 8.4.3) and point sources (see section 8.4.5).
- Submit the job.
- Visualize your results using `ncview`.

Very Short-lived Substances

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 (Br_y) budget. Once

**EX 8.2**

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.trng17.case2.vsls`. Edit the run script according to the namelist parameters you find in Section 8.4. 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 8.4.3). 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



EX 8.3

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.trng17.case3.volc`. Edit the run script according to the namelist parameters you find in Section 8.4. 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`.

- Create the XML file that is needed to define tracers (see section 8.4.3). You also have to create an XML file specifying the aerosol modes in the simulation (see section 8.4.4). 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

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.trng17.case4.seas`. Edit the run script according to the namelist parameters you find in section 8.4. 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 8.4.3). You also have to create an XML file specifying the aerosol modes in the simulation (see section 8.4.4). 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`.

**EX 8.4**

9. 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).

9.1. 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 9.1 for a schematic of the basic assimilation process.

9.1.1. Variational Data Assimilation

The basic 3D-VAR step minimizes the functional

$$\mu(x) := \|x - x^{(b)}\|_{B^{-1}}^2 + \|y - H(x)\|_{R^{-1}}^2, \quad (9.1)$$

ICON Basic Cycling Environment

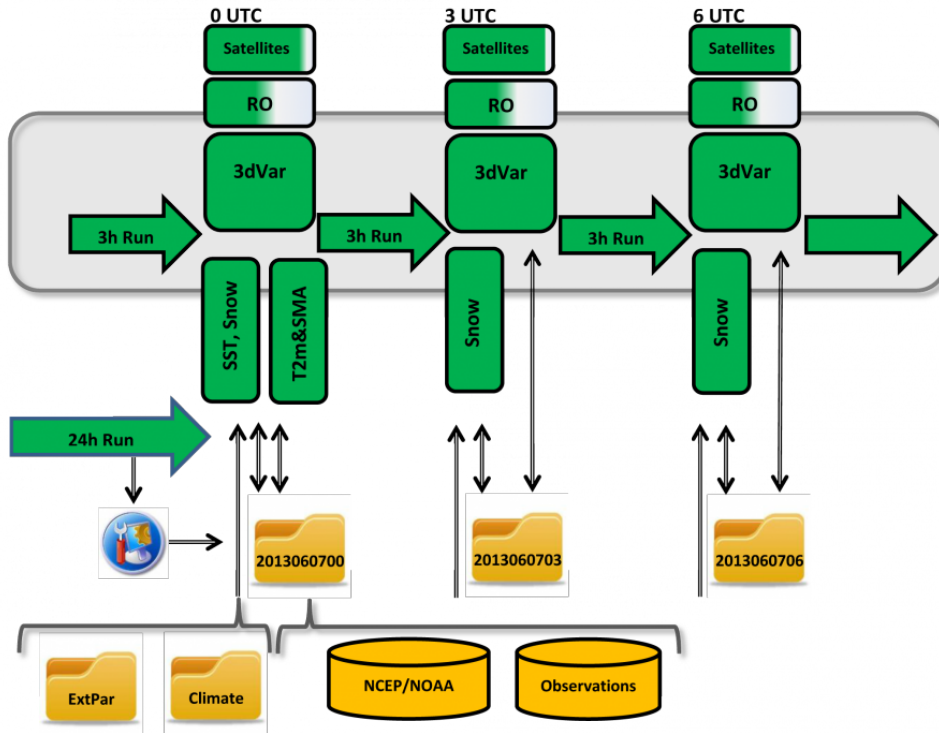


Figure 9.1.: Basic ICON cycling environment using 3DVar. Observations are merged with a background field taken from a 3 h forecast (first guess) of the ICON model. *Courtesy of R. Potthast, DWD.*

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 (9.1) is given by

$$x^{(a)} = x^{(b)} + BH^T(R + HBH^T)^{-1}(y - H(x^{(b)})). \quad (9.2)$$

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 (9.1) is not carried out explicitly by (9.2), 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.

9.1.2. 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

$$B = \frac{1}{L-1} \sum_{\ell=1}^L (x_k^{(\ell)} - \bar{x}_k)(x_k^{(\ell)} - \bar{x}_k)^T, \quad (9.3)$$

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 (9.3). 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 (9.3). 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.

9.1.3. 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 resolution for the ensemble and a 13 km global resolution for the deterministic state. The ensemble B matrix is then carried over to the finer deterministic resolution by interpolation. See Section 9.2 for more details on the operational assimilation system at DWD.

9.1.4. 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.

9.2. Assimilation Cycle at DWD

The *Assimilation cycle* iterates the steps described in Section 9.1: 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 9.1.3. 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 resolution 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 resolution, which is then used to initialize a deterministic main forecast at the same resolution.

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 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) τ , p , u , v , qv .

Grid(s) on which it is performed: global

Carried out at every assimilation time step (3h) 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 2m 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, 2m-temperature analysis (see below) to be used as observations.

Main output: Soil moisture analysis and estimated error.

2m-temperature analysis

Although carried out only at 0 UTC, it is ran 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 (a second node partition with 544 Intel Broadwell nodes will not be used in these exercises). Each node therefore has 24 computational cores. 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. For normal test jobs it should be enough to use 10-15 nodes (depending on the chosen grid resolution).

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> .

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 for the ICON Model

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.

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*. You can find a description of how to use and set up such configuration files in the ICON user manual, cf. Zängl et al. (2014).

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 4.3 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>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
<code>aswflx_par_sfc</code>	Downward PAR flux 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
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_qc_conv	convective 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_turb	turbulence tendency of specific cloud ice
ddt_qv_conv	convective 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_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
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

Continued on next page

Table C.1 – *Continued from previous page*

Variable name	Description
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
grad_topo	gradient of geometric height of the earths surface above sea level
grf_tend_mflx	normal mass flux tendency (grid refinement)
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

Continued on next page

Table C.1 – *Continued from previous page*

Variable name	Description
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_m1_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_fle	horizontal mass flux at edges
mass_fle_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
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
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

Continued on next page

Table C.1 – *Continued from previous page*

Variable name	Description
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	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
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
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

Continued on next page

Table C.1 – *Continued from previous page*

Variable name	Description
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	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_cl	CRU near surface temperature climatology
tcm	turbulent transfer coefficients for momentum
td_2m	dew-point in 2m
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
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 earths 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)

Continued on next page

Table C.1 – *Continued from previous page*

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

Bibliography

- Avissar, R. and R. Pielke, 1989: A parameterization of heterogeneous land surfaces for atmospheric numerical models and its impact on regional meteorology. *Mon. Wea. 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. *Journal of Climate*, **16**(16), 2676–2699.
- 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.
- Bloom, S. C., L. L. Takacs, A. M. D. Silva, and D. Ledvina, 1996: Data assimilation using incremental analysis updates. *Mon. Wea. Rev.*, **124**, 1256–1270.
- 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.
- 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.
- 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. Wea. Rev.*, **139**, 2163–2169.
- Leuenberger, D., M. Koller, and C. Schär, 2010: A generalization of the sleeve vertical coordinate. *Mon. Wea. Rev.*, **138**, 3683–3689.
- 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.
- Mironov, D., 2008: Parameterization of lakes in numerical weather prediction. description of a lake model. Technical report, Consortium for Small-Scale Modelling.
- Mironov, D., B. Ritter, J.-P. Schulz, M. Buchhold, M. Lange, and E. Machulskaya, 2012: Parameterisation of sea and lake ice in numerical weather prediction models of the german weather service. *Tellus A*, **64**(0).
- 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. *Journal of Geophysical Research: Atmospheres*, **102**(D14), 16663–16682.
- Neggers, R. A. J., M. Köhler, and A. C. M. Beljaars, 2009: A dual mass flux framework for boundary layer convection. Part I: Transport. *Journal of the Atmospheric Sciences*, **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. *Journal of Climate*, **23**(22), 5905–5926.
- Pincus, R. and B. Stevens, 2013: Paths to accuracy for radiation parameterizations in atmospheric models. *Journal of Advances in Modeling Earth Systems*, **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. Wea. 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.
- 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. *Geoscientific Model Development*, **8**(6), 1659–1676.
- Schär, C., D. Leuenberger, O. Fuhrer, D. Lüthi, and C. Girard, 2002: A new terrain-following vertical coordinate formulation for atmospheric prediction models. *Mon. Wea. 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.
- 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. *Meteorology and Atmospheric Physics*, **92**(1), 45–66.
- Simmons, A. and D. Burridge, 1981: An energy and angular-momentum conserving finite-difference scheme and hybrid vertical coordinates. *Mon. Wea. Rev.*, **109**, 758–766.
- Tiedtke, M., 1989: A comprehensive mass flux scheme for cumulus parameterization in large-scale models. *Mon. Wea. Rev.*, **117**(8), 1779–1800.
- Zängl, G., D. Reinert, F. Prill, M. Giorgetta, L. Kornbluh, and L. Linardakis, 2014: *ICON User's Guide*. DWD & MPI-M.
- Zängl, G., D. Reinert, P. Ripodas, and M. Baldauf, 2015: The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core. *Q. J. R. Meteorol. Soc.*, **141**, 563–579.

Index of Namelist Parameters

The following index contains only namelist parameters covered by this tutorial. Please take a look at the document

`icon-dev/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), 54
- ana_varnames_map_file, **53**
- art_nml (Namelist), 114
- atmo_dyn_grids, 47
- bdy_indexing_depth, **16**, 16
- cart_volcano_file, 124
- diffusion_nml (Namelist), 43
- dom, **57**, 58
- dt_checkpoint, **88**, 88, 89
- dt_checkpoint (Namelist), 91
- dt_conv, **51**, 51
- dt_gwd, **51**
- dt_iau, **55**, 55, 56, 64
- dt_rad, **51**, 51
- dt_restart, 88, **89**
- dt_shift, **55**, 55, 56, 64
- dt_sso, 51
- dtype, **46**, 65
- dtype_latbc, **73**, 82
- dwdana_filename, **54**, 54, 55, 64
- dwdfg_filename, **54**, 54, 55, 64, 73, 82
- dynamics_grid_filename, **44**, 52, 53, 63, 67
- dynamics_nml (Namelist), 43
- dynamics_parent_grid_id, **45**, 47, 52, 67
- end_datetime_string, **51**, 51, 63, 82, 89, 93
- extpar_filename, **52**, 52, 64, 89
- extpar_nml (Namelist), 43, 52
- filetype, **57**, 57
- flat_height, **40**
- grid_nml (Namelist), 39, 44, 45, 47, 52, 67, 71, 86, 87
- gridgen_nml (Namelist), 16, 17
- h_levels, **88**
- hbot_qvsubstep, **85**, 85, 86
- hl_varlist, **87**, 87
- htop_moist_proc, **85**, 85, 86, 91, 93
- i_levels, **88**
- iart_seasalt, 122
- iart_volcano, 122
- iforcing, **42**, 43, 47, 52, 63, 89
- ifs2icon_filename, **55**, 89
- ihadv_tracer, **85**
- il_varlist, **87**, 88
- in_filename, 33
- in_grid_filename, **31**, 33
- in_type, **31**
- ini_datetime_string, **51**, 51, 54, 55, 63, 82, 89
- init_mode, **53**, 53–55, 64, **73**, 73, 82, 89
- initicon_nml (Namelist), 53–55, 73
- input_field_nml (Namelist), 27, 28, 31
- inwp_cldcover, **75**
- inwp_convection, **75**
- inwp_gscp, **75**, 75, **78**, 78
- inwp_gwd, **75**
- inwp_radiation, **75**
- inwp_sso, **75**
- inwp_surface, **75**
- inwp_turb, **75**
- io_nml (Namelist), 57, 88
- itopo, **43**, 43, 47, 52, 63, 89
- itype_latbc, **72**

- ivctype, **38, 40**
- l_limited_area, **71, 82**
- lart, **114**
- lart_diag_out, **122, 125**
- latbc_boundary_grid, **74, 82**
- latbc_filename, **73, 73, 82**
- latbc_path, **73, 73, 82**
- latbc_varnames_map_file, **73, 73**
- ldynamics, **43, 47, 52, 63, 89**
- limarea_nml (Namelist), **72–74**
- llake, **75**
- lnd_nml (Namelist), **19**
- lread_ana, **73, 82**
- lredgrid_phys, **86, 95**
- lrestart, **88, 93**
- lseice, **75**
- ltestcase, **42, 47, 52, 63, 89**
- ltimer, **61**
- ltransport, **43, 48, 52, 63, 89**
- lwrite_parent, **17, 87**
- m_levels, **57**
- master_nml (Namelist), **54, 88**
- min_lay_thckn, **40, 40**
- ml_varlist, **57, 57, 87**
- model_base_dir, **54**
- ncstorage_file, **31**
- ndyn_substeps, **49, 50, 65**
- nh_test_name, **43, 43, 47**
- nh_testcase_nml (Namelist), **37, 43, 48**
- nlev_latbc, **73, 82**
- nonhydrostatic_nml (Namelist), **38, 40, 43, 49, 85**
- nsteps, **46, 51, 93**
- ntiles, **19**
- num_io_procs, **58, 63, 89**
- num_lev, **38, 40, 47, 67**
- num_prefetch_proc, **74**
- num_restart_procs, **58, 89**
- nwp_phy_nml (Namelist), **43, 51, 75, 78**
- out_grid_filename, **31**
- out_type, **31**
- output, **56**
- output_bounds, **57**
- output_filename, **57**
- output_nml (Namelist), **46, 48, 56–58, 87, 88**
- output_nml_dict, **57**
- p_levels, **88**
- parallel_nml (Namelist), **58, 74, 89**
- pl_varlist, **87, 88**
- radiation_grid_filename, **52, 63, 87, 95**
- reg_lat_def, **88**
- reg_lon_def, **88**
- remap, **57, 88**
- remap_nml (Namelist), **31**
- restart_filename, **88, 88**
- run_nml (Namelist), **38, 42, 43, 46, 47, 51, 52, 56, 61, 67, 88, 93, 114, 137**
- sleve_nml (Namelist), **40**
- steps_per_file, **57**
- time_nml (Namelist), **51, 89**
- timers_level, **61**
- top_height, **40**
- tracer_inidist_list, **48, 48**
- tracer_names, **48, 48**
- transport_nml (Namelist), **43, 48, 52, 85**
- var_in_mask, **28**

*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

List of Exercises

Installation of the ICON Model Package	12
Exercise 1.1	12
Necessary Input Data	32
Exercise 2.1: Download of Global Data	32
Exercise 2.2: Retrieving DWD Analysis Data	32
Exercise 2.3: Retrieving IFS Data	33
Exercise 2.4: Preparation of Limited Area Runs	33
Running Idealized Test Cases	46
Exercise 3.1: Jablonowski-Williamson Test Case	46
Exercise 3.2: Nested Subdomain	47
Exercise 3.3: Tracer Advection	48
Real Data Test Cases	62
Exercise 4.1: Starting a Global ICON Forecast from DWD Analysis	63
Exercise 4.2: Timestepping	65
Exercise 4.3: Run Time Performance	65
Exercise 4.4: Parallelization	65
Exercise 4.5: Writing Output for Driving a Limited Area Simulation	66
Exercise 4.6: A Deeper Look into Spin-Up Effects	69
Exercise 4.7: Checking for Bit-Reproducibility	69
Limited Area Mode	81
Exercise 5.1: Limited Area Run	81
Exercise 5.2: Modifying the Temporal Resolution	83
Exercise 5.3: Implementing New Diagnostics	83
ICON NWP Mode: Further Features	89
Exercise 6.1: Run the ICON Model in Real Data Mode	89
Exercise 6.2: Running ICON with Different Output Products	90
Exercise 6.3: Restarting a Simulation	93
Exercise 6.4: Reduced Grid for Radiation	94
Running ICON-ART	123
Exercise 8.1: Point Sources in ICON-ART LAM	123
Exercise 8.2: Very Short-lived Substances	123
Exercise 8.3: Volcano Eruption	124
Exercise 8.4: Sea Salt Aerosol	125