

ICON Tutorial



Working with the ICON Model

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

Max-Planck-Institut

für Meteorologie





Deutscher Wetterdienst Wetter und Klima aus einer Hand

Acknowledgments

Many people contributed to this manuscript.

The section on ICON physics was partly provided by S. Schäfer (radiation, Section 3.8.1), J. Helmert (land-soil model TERRA, Section 3.8.11), D. Klocke (convection parameterization, Section 3.8.4), M. Köhler (cloud-cover parameterization and turbulence, Sections 3.8.5, 3.8.6), D. Mironov (summary of sea-ice and lake model, Sections 3.8.9, 3.8.10), M. Raschendorfer (turbulence, Section 3.8.6), and A. Seifert (grid-scale microphysics parameterization, Section 3.8.3), DWD Physical Processes Division.

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

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

S. Rast (MPI-M) provided useful specifics on the grid construction, internal representation of fields and other details, see Rast (2017), in particular in Ch. 9.

The Section 10.4.2 (Post-Processing using Fieldextra) has been provided by P. Baumann and J.-M. Bettems, MeteoSwiss.

Section 10.3.4 (Visualization with R) has been contributed by J. Förstner, DWD Physical Processes Division.

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

The Appendix A on the basic usage of DWD's HPC platform is based on T. Steinert's and U. Schättler's (DWD) documentation of the NEC SX-Aurora.

DOI: 10.5676/DWD_pub/nwv/icon_tutorial2023

This work may not be translated or copied in whole or in part without the written permission of the publisher.

Publisher

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

Editors

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

Contents

0.	Pref	ace	1
	0.1.	How T	his Document Is Organized 1
	0.2.	How to	Obtain a Copy of the ICON Model Code 2
	0.3.	Furthe	r Documentation 3
1.	Insta	allation	of the ICON Model Package 5
	1.1.	The IC	ON Model Package
		1.1.1.	Directory Lavout
		1.1.2.	Libraries Needed for Data Input and Output
		1.1.3.	Namelist Input for the ICON Model
	1.2.	Config	uring and Compiling the Model Code
		1.2.1.	Computer Platforms
		1.2.2.	Configuring and Compiling 12
	1.3.	The D	WD ICON Tools
		1.3.1.	General Overview
		1.3.2.	Configuring and Compiling the DWD ICON Tools $\hdots \ldots \ldots \ldots \hdots 16$
2	Nece	essary I	nnut Data 19
2.	2 1	Horizo	ntal Grids 19
	2.1.	2 1 1	ICON Grid Files 21
		2.1.1. 2.1.2	ICON "Nests" 23
		2.1.2. 2.1.3	Mapping of Geodesic Coordinates to the Sphere 24
		2.1.0.	Download of Predefined Grids
		2.1.5.	Grid Generator: Invocation from the Command Line
		2.1.6.	Grid Generator: Invocation via the Web Interface
		2.1.7.	Offline ExtPar Subgrid Extraction
		2.1.8.	Which Grid File is Related to My Simulation Data?
		2.1.9.	Planar Torus Grids
	2.2.	Initial	Conditions
		2.2.1.	Obtaining DWD Initial Data
		2.2.2.	Obtaining ECMWF IFS Initial Data
		2.2.3.	Remapping Initial Data to Your Target Grid
	2.3.	Bound	ary Data Preparation for ICON-LAM
	2.4.	Extern	al Parameter Files
		2.4.1.	ExtPar Software
		2.4.2.	Topography Information
		2.4.3.	Additional Information for Surface Tiles
		2.4.4.	Parameter Files for Radiation
3.	Mod	lel Desc	cription 59
	3.1.	Govern	ing Equations

	3.2.	The M	odel Reference State	. 63
	3.3.	Simplif	fying Assumptions in the Recent Model Version	. 64
	3.4.	Vertica	l Coordinates	. 66
		3.4.1.	Terrain-following Hybrid Gal-Chen Coordinate	. 67
		3.4.2.	SLEVE Coordinate	. 68
	3.5.	Tempo	ral Discretization	. 70
		3.5.1.	Basic Idea	. 70
		3.5.2.	Implementation Details	. 71
	3.6.	Tracer	Transport	. 77
		3.6.1.	Directional Splitting	. 78
		3.6.2.	Horizontal Transport	. 79
		3.6.3.	Vertical Transport	. 81
		3.6.4.	Reduced Calling Frequency	. 84
		3.6.5.	Some Practical Advice	. 85
	3.7.	Physics	s-Dynamics Coupling	. 89
		3.7.1.	ICON Time-Stepping	. 89
		3.7.2.	Fast and Slow Processes	. 90
		3.7.3.	Isobaric vs. Isochoric Coupling Strategies	. 91
	3.8.	ICON	NWP-Physics in a Nutshell	. 93
		3.8.1.	Radiation	. 93
		3.8.2.	Saturation Adjustment	. 96
		3.8.3.	Cloud Microphysics	. 98
		3.8.4.	Cumulus Convection	. 99
		3.8.5.	Cloud Cover	. 100
		3.8.6.	Turbulent Diffusion	. 101
		3.8.7.	Sub-grid scale orographic drag	. 105
		3.8.8.	Non-orographic gravity wave drag	. 108
		3.8.9.	Lake Parameterization Scheme FLake	. 108
		3.8.10.	Sea-Ice Parameterization Scheme	. 110
		3.8.11.	Land-Soil Model TERRA	. 112
		3.8.12.	Reduced Model Top for Moist Physics	. 115
	3.9.	Variab	le Resolution Modeling	. 116
		3.9.1.	Parent-Child Coupling	. 118
		3.9.2.	Processing Sequence	. 126
		3.9.3.	Technical and Performance Aspects	. 128
	3.10.	Reduce	ed Radiation Grid	. 129
	_			101
4.	Run	ning Ide	ealized Test Cases	131
	4.1.	Main S	witches for Idealized Test Cases	. 131
		4.1.1.	Activating/De-activating Main Model Components	. 131
		4.1.2.	Specifying the Computational Domain(s)	. 132
	1.0	4.1.3.	Integration Time Step and Simulation Length	. 133
	4.2.	Jablon	owski-Williamson Baroclinic Wave Test	. 134
		4.2.1.	Recommended Namenst Settings	. 134
		4.2.2.	Enabling Passive Tracers	. 135
	4.9	4.2.3.	Activation of Nested Domains	. 137
	4.3.	Straka	Density Current Test	. 138
		4.3.1.	Relevant Namelist Switches in nn_testcase_nml:	. 140

5.	Run	ning Real Data Test Cases 143
	5.1.	Model Initialization
		5.1.1. Basic Settings for Running Real Data Runs
		5.1.2. Starting from Uninitialized DWD Analysis
		5.1.3. Starting from Uninitialized DWD Analysis with IAU
		5.1.4. Starting from Initialized DWD Analysis
		5.1.5. Starting from IFS Analysis
	5.2	Starting or Terminating Nested Domains at Runtime 152
	0.2.	
6.	Run	ning ICON-LAM 153
	6.1.	Limited Area Mode vs. Nested Setups
	6.2	Nudging in the Boundary Region 154
	6.3	Model Initialization 157
	6.4	Reading Lateral Boundary Data 150
	0.1.	6.4.1 Naming Scheme for Lateral Boundary Data 161
		6.4.2 Dro Establing of Doundary Data (Mandatory)
	GE	Tranical Seture
	0.0.	
7	Mod	lel Output 167
	7 1	Settings for the Model Output 167
	1.1.	7.1.1 Output on Pagular Crids and Vertical Interpolation 160
		7.1.1. Output on Regular Grids and Vertical Interpolation
		7.1.2. Remarks on the Horizontal Interpolation
		7.1.3. Interpolation onto Rotated Lat-Lon Grids
		7.1.4. Output Rank Assignment
	7.2.	Checkpointing and Restart
	7.3.	Meteogram Output
Q	Dar	Illelization and Performance Accests
0.		Modes of Depullel Execution 170
	0.1.	Setting of Faranel Execution
	0.2.	
	8.3.	Best Practice for Parallel Setups
		8.3.1. MPI Tasks and OpenMP Threads
		8.3.2. Blocking (nproma) \ldots 182
		8.3.3. Mixed Single/ Double Precision in ICON
		$8.3.4. Bit-Reproducibility \dots \dots$
	8.4.	Basic Performance Measurement
	8.5.	ICON on Accelerator Devices (GPUs)
		8.5.1. Configuring and Compiling ICON-OpenACC
		8.5.2. Special Namelist Options for ICON-OpenACC
		8.5.3. Implementation Details
9.	Pro	gramming ICON 189
	9.1.	Representation of 2D and 3D Fields
	9.2.	Data Structures
		9.2.1. Description of the Model Domain
		9.2.2. Date and Time Variables
		9.2.3. Data Structures for Physics and Dynamics Variables
		9.2.4. Parallel Communication

9.3.	Implementing Own Diagnostics	. 198
9.4.	NWP Call Tree	. 202
10. Post	-Processing and Visualization	205
10.1	Retrieving Data Set Information	. 205
	10.1.1. The ncdump Tool	. 205
	10.1.2. CDO – Climate Data Operators	. 206
10.2	Plotting Data Sets on Regular Grids: ncview	. 207
10.3	Plotting Data Sets on the Triangular Grid	. 208
	10.3.1. Visualization with Python	. 208
	10.3.2. Visualization with PyNGL	. 209
	10.3.3. NCL – NCAR Command Language	. 214
	10.3.4. Visualization with R \ldots	. 217
	10.3.5. GMT – Generic Mapping Tools	. 220
10.4	Post-Processing of Data Sets	. 223
	10.4.1. Post-Processing using the CDO	. 223
	10.4.2. Post-Processing using Fieldextra	. 224
11.ICO	N's Data Assimilation System and Analysis Products	227
11.1	Data Assimilation	. 227
	11.1.1. Variational Data Assimilation	. 228
	11.1.2. Ensemble Kalman Filter	. 229
	11.1.3. Hybrid Data Assimilation	. 229
	11.1.4. Surface Analysis	. 230
11.2	Assimilation Cycle at DWD	. 230
11.3	Analysis Products	. 232
	11.3.1. Uninitialized Analysis for IAU	. 232
	11.3.2. Uninitialized Analysis	. 235
	11.3.3. Initialized Analysis	. 237
_		
Append	lix A. The Computer System at DWD	241
Append	ix B. Table of ICON Output Variables	245
Bibliog	raphy	259
Index o	f Namelist Parameters	269

0. Preface

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

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

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

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

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

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

0.1. How This Document Is Organized

Not all topics in this manuscript are covered during the workshop. Therefore, the manuscript can be used as a textbook, similar to a user manual for the ICON model.

Readers are assumed to have a basic knowledge of the design and usage of numerical weather prediction models.

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

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

To some extent this document can also be used as a reference manual. We refer to the index on page 269 for a quick look-up of namelist parameters.

Paragraphs describing common pitfalls and containing details for advanced users are marked by the symbol $\sqrt[n]{2}$.

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

To institutions, the ICON model is distributed under an **institutional license** issued by DWD. To obtain a grant of license that must be signed and returned to the DWD, please contact icon@dwd.de or follow the information on the public ICON web site

https://code.mpimet.mpg.de/projects/iconpublic

To individuals, the ICON model is distributed under a **personal non-commercial re**search license distributed by MPI-M, see also the instructions on the public ICON web site. Access to the source code management system git is limited to the development partners of the ICON project.

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

Data Services

On the ICON web page under the mentioned URL you will also find the access to the **grid generator web service** (see Section 2.1.6), and the web links to ICON's official grid download site and GRIB2 definitions.

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

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

0.3. Further Documentation

The ICON model is accompanied by various other manuals and documentation. An extensive list is available and constantly kept up-to-date in the documentation section of the public ICON web site https://code.mpimet.mpg.de/projects/iconpublic.

We restrict ourselves to a small subset in the following.

Scientific Documentation

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

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

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

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

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

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

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

Technical Documentation

For model users who intend to process data products of DWD's operational runs, the DWD database documentation may be a valuable resource, see Reinert et al. (2020). It can be found (in English language) on the DWD web site

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

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

icon/doc/Namelist_overview.pdf

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

which is deployed together with the code.

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

Finally, please note the FAQ section on the ICON web site which covers a variety of common pitfalls.

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)

For this tutorial the release v2.6.6 of the ICON code is used (state *February 2023*). It is close to DWD's currently operational *icon-nwp* version (see below for explanation). The code also contains the ocean model developed at MPI-M which is, however, not covered by this tutorial.

• ICON-ART for aerosols and reactive trace gases

The ART module, where ART stands for Aerosols and Reactive Trace gases, is an extension of the ICON model to enable the simulation of gases, aerosol particles and related feedback processes in the atmosphere. The module is provided by the Karlsruhe Institute of Technology (KIT) and requires a separate license.

• 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.

ICON-NWP code version: The versioning of ICON is a bit complex and reflects the parallel development in several "flavors" like "atmosphere", "ocean", and several more. There are five important branches tagging versions that reached certain milestones: *icon-mpim* (common repository for MPI-M, mainly AES physics in the atmosphere, land, ocean and infrastructure), *icon-nwp* (numerical weather prediction, mainly dynamics and physics of the LEM and NWP configurations of the atmospheric model), *icon-dkrz* (development and integration of new infrastructure components), *icon-cscs* (focus on parallelization, like porting ICON to GPUs), *icon-kit* (development and integration of the latest ART features). The common release version integrates the stable components of all branches.

Each tag contains all model components, but the latest tag of, say, *icon-mpim* may not contain the most recent developments of the NWP physics although these are already included into the latest *icon-nwp* tag.

1.1.1. Directory Layout

Figure 1.1 shows a brief description of the directory structure of the ICON model, located in the directory icon.



Figure 1.1.: Directory structure of the ICON model.

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. When you open this newly created folder you find a **bin** subdirectory containing the ICON binary **icon** and several other subdirectories containing the compiled module files.

Subdirectory config

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

Subdirectory src

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

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

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

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

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

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

The libraries play a particularly important role in the execution of I/O tasks. 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 machineindependent data formats that support the creation, access, and sharing of arrayoriented scientific data. NetCDF files contain the complete information about the dependent variables, the history, and the fields themselves. The NetCDF file format is also used for the definition of the computational mesh (grid topology). For more information on NetCDF see http://www.unidata.ucar.edu.

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

The Climate Data Interfaces (CDI) - externals/cdi

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

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

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

The NetCDF library - libnetcdf.a

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

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

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

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

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

The ECMWF ecCodes package¹ - libeccodes.a, libeccodes_f90.a

The European Centre for Medium-Range Weather Forecasts (ECMWF) has developed an application programmers interface (API) to pack and unpack GRIB1 as well as GRIB2 formatted data. For reading and setting meta-data, the *ecCodes* package uses the so-called key/value approach, which means that all the information contained in the GRIB message is retrieved through alphanumeric names. Indirect use of this ecCodes library in the ICON model is implemented through the CDI.

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

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

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

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

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

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

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

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

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

GRIB Definition Files

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

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

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

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

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

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

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

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

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

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



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

1.1.3. 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. The run scripts in this tutorial create a file NAMELIST_NWP which contains all user-defined namelist parameters, together with some substituted shell script variables. Discussing all available namelist switches is definitely beyond the scope of this tutorial. We will merely focus on the particular subset of namelist switches that is necessary to setup an idealized model run as well as real case runs using the NWP physics package. A complete list of namelist switches can be found in the namelist documentation

icon/doc/Namelist_overview.pdf

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 ecCodes version are documented in the log output of each model run.

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

1.2.1. Computer Platforms

For a small number of HPC platforms settings are provided with the code. The two most important examples of ICON-NWP are the NEC system in Offenbach/Ludwigshafen and "Levante" at the DKRZ in Hamburg. As of September 2022, the technical specifications of these platforms are as follows:

NEC SX-Aurora cluster (Research Cluster Ludwigshafen "rcl.dwd.de")

 $3520~\mathrm{NEC}$ SX-Autora Tsubasa vector engine CPUs

- 1.584 GHz, double precision peak performance ca. 2.15 TFLOPS,
- 48 GiB HBM2 3D-stacked memory
- 440 x86 Vector Hosts with 8 Vector Engines attached to each host AMD EPYC "Rome" (24 cores, 2.8GHz, 512 GiB memory)

MPI:	NEC MPI 2.22.0
NetCDF:	Version 4.7.3
Compiler:	NEC Fortran $4.0.0 / \text{gcc v}9.2.0$

HLRE-4 cluster "Levante" (DKRZ Hamburg)

CPU-Partition

2520 compute nodes AMD 7763 (2 CPUs/node, 64 cores/CPU, 256 Gb)

- 294 compute nodes AMD 7763 (2 CPUs/node, 64 cores/CPU, 512 Gb)
- 18 compute nodes AMD 7763 (2 CPUs/node, 64 cores/CPU, 1024 Gb) GPU-Partition
- 60 compute nodes AMD 7713 (2 CPUs/node, 64 cores/CPU, 512 Gb) compute nodes Nvidia A100 (4 GPUs/node, 80/40 Gb)

MPI:	OpenMPI 4.1.2	
NetCDF:	Version 4.5.3	
Compiler:	Intel Fortran compiler ifort	2021.5.0

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



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

ICON Model Tutorial

Fortran Compiler	Working Version(s)
GNU	gcc v11.2.0 gcc v12.1.0 gcc v12.2.0
Cray	ftn v12.0.3
Intel	ifort v2021.5.0 ifort v2021.6.0
NAG	nagfor v7.1.7101 nagfor v7.1.7114
NEC	nfort v4.0.0
NVIDIA (CPU)	nvfortran v21.3.0 nvfortran v22.7.0
NVIDIA (GPU)	nvfortran v21.3.0 nvfortran v22.7.0

 Table 1.1.: Compiler versions which are tested regularly and which are known to successfully build the ICON code (state February 2023).

1.2.2. Configuring and Compiling

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

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

The configure command scans the build environment and generates an appropriate Makefile. The list of arguments enabling a successful configuration might be quite long and difficult to compose, therefore, instead of running the generic configure script directly, users are recommended to execute a corresponding platform- or machine-specific configuration wrapper that sets the required compiler and linker flags as well as the recommended set of configure options.

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

./configure --help

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

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

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

make distclean

Example: Configuration and build process for the NEC SX-Aurora. The NEC platform represents a special case in the sense that two separate ICON binaries are created, which will be simultaneously executed on the x86 hosts nodes and the vector engines.

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

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

```
mkdir -p build/VE,VH
cd build/VE
../../config/dwd/rcl.VE.nfort
make -j6
cd ../../build/VH
```

```
../../config/dwd/rcl.VH.gcc
make -j6
```



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

Users of the ECMWF system may find recent pre-compiled binaries in the following subdirectory:

/sc1/home/zde/routfox/abs

ICON Model Tutorial

1.3. The DWD ICON Tools

The DWD ICON Tools provide a number of utilities for the pre- and post-processing of ICON model runs. The DWD ICON Tools are not generally available ICON software but were developed by the DWD as an add-on. Note that the software is not distributed under the personal license that is provided by the Max-Planck Institute for Meteorology.

1.3.1. General Overview

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



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

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

ICONGRIDGEN

- Used in Section 2.1.5

The icongridgen tool is a simple grid generator. It creates icosahedral grids from scratch, which can be fed into the ICON model. Alternatively, an existing global or local grid file is taken as input and parts of this input grid (or the whole grid) are refined via bisection. No storage of global grids is necessary and the tool also provides an HTML plot of the grid. The grid generator provides the basis for the DWD grid generator web tool under https://webservice.dwd.de/cgi-bin/spp1167/webservice.cgi.

ICONREMAP

- Used in Sections 2.2.3, 2.3

The iconremap utility is especially important for pre-processing the initial data for the basic test setups in this manuscript. iconremap (*ICO*sahedral Nonhydrostatic model *REMAP*ping) is a utility program for horizontally interpolating ICON data onto regular grids and vice versa. Besides, it offers the possibility to interpolate between triangular grids of different grid spacing.

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

Several interpolation algorithms are available: Nearest-neighbor remapping, radial basis function (RBF) approximation of scalar fields, area-weighted formula for scalar fields, RBF interpolation for wind fields from cell-centered zonal, meridional wind components u, v to normal and tangential wind components at edge midpoints of ICON triangular grids (and reverse), and barycentric interpolation. For more remarks on the available interpolation methods, see Section 7.1.2.

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

ICONSUB

- Used in Section 2.3

The iconsub tool (*ICO*sahedral Nonhydrostatic model SUBgrid extraction) allows "cutting" sub-areas out of ICON data sets.

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

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

ICON Model Tutorial

ICONGPI

The icongpi tool (*ICO*sahedral *N*onhydrostatic model *G*rid *P*oint *I*nformation) 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.

ICONDELAUNAY

- see Section 7.1.2

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

1.3.2. Configuring and Compiling the DWD ICON Tools

To compile the DWD ICON Tools binaries, log into the Linux cluster node rclh³ for NEC SX-Aurora cross-compilation and change into the base directory:

cd dwd_icon_tools

This directory contains a GNU Autotools configure script which, when run, scans the build environment and generates a Makefile appropriate for that build environment.

The configure expects numerous platform-dependent options which can be listed by the command configure -help.

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

make

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

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

 $^{^3 \}rm Note that \, {\tt rclh}$ serves as an alias for one of the Linux cluster nodes ${\tt rcnl*}.$

List of pre-configured platforms: Pre-set options for a short list of available compile targets are contained in the do_configure.sh auxiliary script.

Hostname	Target description
oflws*, omlws*	Generic Linux platform without MPI support; compiled with the gcc compiler.
<pre>rcnl*, target=="rcl"</pre>	DWD Linux cluster.
rcnl*, target == "VH"	cross compilation for DWD NEC SX-Aurora vector host with MPI support.
breeze4*	DKRZ "breeze" Linux cluster.
levante*	DKRZ BullSequana XH2000 supercomputer "Levante".
a*6-1*.bullx	ECMWF Atos Sequana XH2000 system.
daint*	compilation for CSCS Daint cluster.

The o[fm]lw* target in the do_configure.sh helper script is related to a more or less generic Linux workstation but probably requires adaptation.

In the following, we list the invocation of the do_configure.sh helper script in detail:

DWD, rcnl: Cross compilation for NEC vector hosts

DWD, rcnl: Compilation for Linux cluster with GNU compiler (without MPI)

DKRZ, levante: Compilation for "Levante"

```
module purge && \
module load gcc/11.2.0-gcc-11.2.0 openmpi/4.1.2-gcc-11.2.0
./do_configure.sh
```

2. Necessary Input Data

Before anything else, preparation is the key to success.

Alexander Graham Bell

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

2.1. Horizontal Grids

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

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

The effective mesh size can be estimated as

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

This formula is motivated as follows:

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

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

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



Figure 2.1.: Left: Illustration of the grid construction procedure. The original spherical icosahedron is shown in red, denoted as R_{1B00} following the nomenclature described in the text. In this example, the initial division (n=2; black dotted), followed by one edge bisection (k=1) yields an R_{2B01} grid (solid lines). Right: Graphical representation of the 12 pentagon points. Here, the base icosahedron is rotated exactly like the DWD operational grids.

Pentagon Points

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

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

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

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

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

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

Cell Neighborhoods

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

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

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

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

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

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

Dual Hexagonal Grid

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

Spring Dynamics Optimization

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

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

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

2.1.1. ICON Grid Files

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

The most important data entries of the main grid file are

• cell (INTEGER dimension) number of (triangular) cells

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

Note that each facet corresponds to a triangle edge and that the dual normal matches the direction of the primal tangent vector but signs can be different.

• uuidOfHGrid (global attribute) Grid fingerprint (see Section 2.1.8).



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

2.1.2. ICON "Nests"

ICON has the capability for running

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

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

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

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

- uuidOfParHGrid (global attribute) Fingerprint of the parent grid (see Section 2.1.8). If your grid does not contain the uuidOfParHGrid global attribute, then you'll need the namelist parameter dynamics_parent_grid_id, see below.
- parent_cell_index (INTEGER array, dimension: #triangles) Global index of coarser parent triangle in the parent grid.
- parent_edge_index (INTEGER array, dimension: #triangle edges) Global index of parent edge (with double length) in the parent grid.
- parent_vertex_index (INTEGER array, dimension: #triangle vertices) Global index of parent vertex in the parent grid.

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

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

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

dynamics_parent_grid_id (namelist grid_nml, list of INTEGER values)

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

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

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

2.1.3. Mapping of Geodesic Coordinates to the Sphere

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

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

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

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

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

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

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



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

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

2.1.4. Download of Predefined Grids

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

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

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

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

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



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

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

2.1.5. Grid Generator: Invocation from the Command Line

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

In this section we will discuss the grid generator **icongridgen** that is contained in the DWD ICON Tools, because this utility also acts as the backend for the publicly available web tool. The latter is shortly described in Section 2.1.6. It is important to note, however, that this grid generator is not capable of generating non-spherical geometries like torus grids, see Section 2.1.9.

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

Grid Generator Namelist Settings

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

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

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

icongridgen --help

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



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

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

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

Settings for ICON-LAM

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

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

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

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

ICON Model Tutorial

ICON Grid Generator + ExtPar Web Frontend		Version Info Web service (Python): ICON Tools: v2.4.7 Web page: 1-8-2022
Particular department		icongridgen: Tag Version 2.4.7: July 2020 Extpar:
		May 2020
Orun grid generator + ExtPar tool generate script for ExtPar tool generate script for ExtPar usobet extraction re-run ExtPar tool for input file		F. Prill (DWD)
Your e-mail address (required for download):		News / Notes
		The extpar data set will be provided as NetCDF4 instead of NetCDF3 from Dezember 1st, 2020 forward.
input file:	BASE DOMAIN	 The base grids for the grid generation process can be downloaded from <u>here</u>. This download page also contains some details on the grid characteristics.
create new base grid RxxByy (no input file) v) = Domain #0 Grid resolution RxxByy:	domain "domain 1"	Some information on ICON's grid structure is available in the <u>ICON</u> database documentation.
xx=3 , yy=7 approx. 13.2 km		
rotation pole (degrees): longitude: 0.0 , latitude: 90.0		Quick Help
	3 0.0. 3.0.E 0.0.E 8.0.E 15.0.E 18.0.E	
(Sub-) Domain #1: • • rectangular • circular • refine all cells of parent domain		
Note: Coordinate data is given with respect to the ellpsoidal reference system WGS84. ExtPar re-interprets coordinates without transformation.	54/ ON Condi	
domain name: domain1	Birmingham BITHERLANDS Berling POLAND	
Enable for grid data.	London GERMANY	
option for nests.	51' 0'N Brusels Cologne 51' 0'N Brusels Frankfurt Promo	
center: longitude (degrees) 8,747375295 latitude (degrees) 50,1031947	Nuremberg CZECHIA	
half width: 5.0	Stiller oP att	
half height: s.o	All' Municho Bratislava Budan	
north pole: longitude (degrees) 0.0 latitude (degrees) 90.0	FRANCE Zuich AUSTRIA	
Estimated resolution: 6 576 km	Lyon A C Lyothan Control	
Estimated no. of triangle cells: 18358 (R3B8)	Bordeaux	
(estimate, does not take into account the cut-out by other domains;	BOSNIA AND Belg REZECOVINA	
note the limit of 3 000 000 cells for the web interface)	Salevo" Salevo" Sa	
note the limit of 3 000 000 cells for the web interface)	Leaflet Tiles @ Esri Esri, DeLorme, NAVTEQ, TomTom, Internap, IPC, USGS, FAO, NPS, NRCAN,	
nde the limit of 3 000 000 cells for the web interface)	Leaffel Tifes ID Earl — Earl, DeLcome, NAVTEQ, TomTorn, Infermap, IPC, USGS, FAO, NPS, INRCAN, GeoBase, Kadaster NL, Ordnance Surviey, Earl Japan, METI, Earl China (Hong Kong), and the GIS User Community	



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

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

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

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

2.1.6. Grid Generator: Invocation via the Web Interface

A web service has been made available to help users with the generation of custom grid files. After entering grid coordinates through an online form, this web service creates a corresponding ICON grid file together with the necessary external parameter file (NetCDF 4 format).

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

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

The web form is mostly self-explanatory. The settings reflect the namelist parameters of the **icongridgen** grid generator tool that runs as the first stage of the web service. The second stage, the ExtPar tool, does require (almost) no further settings.

The tool is capable of generating multiple grid files at once. Please note that the webbased grid generation submits a batch job to DWD's HPC system and takes some time for processing! Due to limited computing resources a threshold is imposed: the maximum grid size which can be generated is 3000000 cells. Of course, larger grid sizes are possible when invoking the grid generator from the command line.

Finally all results (and log files) are packed together into a *.zip archive and the user is informed via e-mail about its FTP download site. Additionally, a web browser visualization of the grids based on *OpenStreetMap* is provided, see Fig. 2.5.

Step 1: Choosing the Base Grid

The web-based generation of ICON grids and their corresponding external parameter (ExtPar) data sets starts from an "input file", which can be chosen from a pull-down menu with a pre-defined list of grids. These grids are identical to those of the download list described in Section 2.1.4.

Alternatively, it is also possible to start from a "synthetic" base grid RnBk by specifying n and k, following the algorithm by Sadourny et al. (1968), see page 19. Synthetic grids are always global domains. Optionally, the location of the grid's pentagon points may be adjusted by a rotation of the base icosahedron, see also Section 2.1.

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

"ASTER" orography. In ExtPar the ICON orography will be generated from the "GLOBE" or the "ASTER" raw data set. The selected data set depends on the horizontal resolution of the target grid. For grid spacings below 3 km the high-resolution, non-global orography "ASTER" will be used, while for spacings greater than 3 km the the coarser and global data set "GLOBE" is selected. The MERIT orography cannot (yet) be requested via the webinterface.

Please kindly note that "ASTER" orography data are only available in the latitude range $60^{\circ}S - 60^{\circ}N$. Requests with spacings below 3 km and with parts or all of the domain located outside this range will fail. To avoid such problems please switch off high-resolution "ASTER" orography for poleward regions by enabling the corresponding checkbox.

Step 2: Specify Global Options

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

• "centre", "subcentre"

These numbers are stored in the output meta-data section for the identification of the originating / generating (sub)-center. The values are defined by the WMO, e.g. DWD: 78/0 (see WMO's Common Code Table C-11 for additional values).

• "spring dynamics optimization"

The ICON grids are based on the spherical icosahedron, but they are post-processed by an iterative algorithm inspired by elastostatics, see the explanation on "spring dynamics optimization" in Section 2.1.

- "max. number of iterations"

maximum number of pseudo-time stepping steps of the elastostatics model.

– "beta factor"

stiffness coefficient of the elastostatics model.

- "fixed lateral boundary"

If this checkbox is set, then the boundary vertices of regional grids are not moved during the optimization. Thus they will still coincide with vertices of the base grid.

Recommendation: For nested domains, do not alter the default settings unless you know the details of the underlying algorithm. The "optimized" grid points of the refined computational grids can deviate substantially from those of the coarser parent grid. Of course, if one considers only a single regional grid, i.e. no nesting hierarchy, then the settings for "spring dynamics" are less critical. We advise a visual inspection of the resulting grids.

• "initial refinement"

Sometimes the grid region "Domain 1" (or the first level of a grid hierarchy) shall not be refined, because only a cut-out subset of the base grid is needed. Disable this checkbox in order to simply extract the first domain from the base grid rather than refining.

• "include base grid"

This setting is disabled by default but it becomes useful when a global base grid has been created from scratch: In this situation the base grid domain # 0 may not only serve the purpose of being the starting point for the subsequent refinement hierarchy, but it can be used in the ICON simulation itself. Selecting "include base grid" will run the ExtPar process and include this dataset into the resulting zip file.

Step 3: Sub-domain Name and Parent Grid ID

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

Numbering: In the web form the base grid is denoted by "#0" while the created domains are denoted by numbers "#1", "#2" and so on.
In the simplest case the domains specify multiple nests on the same refined level. In this case, the "parent grid ID" is always set to "0" (base grid). Besides, domains can be nested, for example

Germany
$$\rightarrow$$
 Europe \rightarrow Global.

Then, the grid for Germany has "parent grid ID=1", and the Europe grid has "parent grid ID=0". The chosen hierarchy of grids is graphically depicted in the top right corner of the web form.

• "domain name"

Each domain requires a name (string) which will be used as a name prefix for the resulting files. Here, it is a good idea to choose (or later rename) file names matching ICON's popular nomenclature iconR<nroot>B<jlev>_DOM<idom>.nc, see Section 5.1.2.

• "number of grid used"

This setting will be written into the grid meta-data. It is part of a GRIB2 mechanism to link data files to their underlying grid files – see also the explanation in Section 2.1.4. For regional domains which are not used operationally, we suggest to choose arbitrary but distinguishable integer numbers.

• "write parent grid"

Enable this checkbox when your grid file is to be used with ICON-LAM and a reduced radiation grid. Note that in this case the grid cannot be used as a nest in standard (non-LAM) mode – see the corresponding remark in Section 2.1.5. Also note that writing a parent grid requires at least one refinement step, i.e. this option cannot be combined with deactivating "initial refinement".

The effect of the switches "include base grid" and "write parent grid" is different when creating ICON-LAM domains: For example, assume that a regional computational grid (domain 1) for ICON-LAM is to be created from the refinement of a global grid. In this case, the "base grid" refers to the global grid, while the "parent grid" denotes a regional grid that includes only the "parent cells" of domain 1.

Step 4: Specify the Grid Type/Shape

• "rectangular"

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



- "rotate" / "north-pole"

By default, the latitude-longitude coordinates for the rectangular refinement area are based on the standard North Pole 90N 0E. You may use, however, a



rotated pole similar to the grids of the COSMO model. This pole rotation value should not be confused with the rotation of the base icosahedron which specifies the location of the grid's pentagon points.

• "circular"

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

• "refine all cells of parent domain"

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

Finally, having filled out all necessary fields of the web form, click on the "Proceed" button – but do not forget to provide your e-mail address! The grid generation job is inserted into DWD's processing queue and you will be informed via e-mail about its completion.

2.1.7. Offline ExtPar Subgrid Extraction

There is a global setting generate script for ExtPar subset extraction available in the web form which leads to a fundamentally different grid generation mode: If you prefer to use your own installation of the DWD ICON Tools grid generator, then you can select this option to prepare the grid generation in an "offline mode".

The option requires a local installation of the ICON Tools version > 2.4.0, which (besides other changes) contain a utility script

icontools/extpar_subset_extraction.py

This Python script loads a "config" script (similar to a Fortran namelist) where the user specifies the desired grid parameters:

```
config = {
    "centre" : 1,
    "subcentre" : 2,
    "initial_refinement" : ".FALSE.",
    "filename" : ["icon_grid_0026_R03B07_G.nc"],
    "dom(1)%outfile" : ["mydomain"],
    "dom(1)%number_of_grid_used" : 99,
    ...
```

Here is where the web grid generator comes in handy: The configuration is automatically created by the web form option generate script for ExtPar subset extraction.

After the Python configuration config ... has been generated, the user can execute the extpar_subset_extraction.py locally on his own machine:

```
./extpar_subset_extraction.py --config=config.py
```

The necessary base grid and ExtPar data are then automatically determined and downloaded from ICON's public web site. The required grid region is cut out from this data set and no separate run of the ExtPar tool is necessary. If there exists a parent grid (reduced radiation grid), then the corresponding part is extracted, too. The whole operation greatly reduces the computational effort to generate the grid data for a custom ICON run.

As a final remark, the extraction script accepts two optional command-line arguments (which are listed with extpar_subset_extraction.py -help):

- The option -icontoolsdir allows to specify a directory where the ICON Tools grid generator binary is located.
- The option **-localcopy** greatly accelerates the script execution for large data sets by loading the base grid data from a local directory.

Also note that the extpar_subset_extraction.py is not fully "offline", since it requires access to the XML metadata and the base grids available under http://icon-downloads.mpimet.mpg.de.

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

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

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

• numberOfGridUsed

This is simply an integer number, as explained in the previous sections. The numberOfGridUsed helps to identify the grid file in the public download list. If the numberOfGridUsed differs between two given data files, then these are not based on the same grid file.

• uuidOfHGrid

This acronym stands for *universally unique identifier* and corresponds to a binary data tag with a length of 128 bits. The UUID can be viewed as a fingerprint of the underlying grid. Even though this is usually displayed as a hexadecimal number string, the UUID identifier is not human-readable. Nevertheless, two different UUIDs can be tested for equality or inequality.

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

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



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

2.1.9. Planar Torus Grids

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

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

The lon-lat parameterization of the torus is

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

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

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

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

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

2.2. Initial Conditions

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

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

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

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

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



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

2.2.1. Obtaining DWD Initial Data

The most straightforward way to initialize ICON is to make use of DWD's analysis products, which are generated operationally every 3 hours and stored in GRIB2 format on the native ICON grid. Deterministic as well as ensemble analysis products are available. Deterministic products are generated by a hybrid Ensemble Variational Data assimilation (En-Var), which combines variational and ensemble methods. Ensemble products are based on a Localized Ensemble Transform Kalman Filter (LETKF) approach. See Chapter 11 for more information on DWD's data assimilation system.

Choosing the Right Product

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

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

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

Uninitialized analysis for IAU

This product consist of a first guess file and an analysis file. The latter contains analysis *increments*, which is the difference between the analysis and the first guess. The validity dates of both files differ. The validity date of the first guess is shifted ahead of the analysis date by 90 min (5 min for ICON-D2). This product is meant for starting the model in *Incremental Analysis Update* (IAU) mode. IAU is a model internal filtering technique for reducing spurious noise introduced by the analysis (see Section 11.3.1).

While this initialization method performs best in terms of noise reduction, it bears the disadvantage that the corresponding analysis product cannot be interpolated horizontally in a straightforward manner. This prevents its use on custom target grids. The underlying reason is that the analysis product contains tiled surface data. Remapping of tiled data sets makes no sense, since the tile-characteristics can differ significantly between individual source and target grid cells. Only *aggregated* surface fields can safely be remapped (see Section 3.8.11 for more details on the surface tile approach).

A list of included fields can be found in Section 11.3.1.

Plain uninitialized analysis

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

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

A list of included fields can be found in Section 11.3.2.



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

Initialized analysis

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

A list of included fields can be found in Section 11.3.3.

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

$$\begin{split} \left\langle \left| \frac{\mathrm{d}p_s}{\mathrm{d}t} \right| \right\rangle &= \frac{1}{A} \sum_i \left| \frac{\mathrm{d}p_s}{\mathrm{d}t} \right| \Delta a_i \\ &= \frac{1}{A} \sum_i \left(\sum_k \left| -g \nabla_h \cdot \left(\overline{\rho} \hat{\boldsymbol{v}}_h \right) \Delta z_k \right| \right) \Delta a_i \,, \end{split}$$

with A denoting the earth's surface and Δa_i denoting the area of the *i*th cell. The mathematical steps to obtain the pressure tendency equation are discussed at the end of Section 3.3. It can be seen that for the uninitialized analysis (red line) the noise level at simulation start is significantly increased when compared to the other two products. It takes about two days of model forecast for the noise levels to align. The uninitialized analysis for IAU (blue) performs best in terms of noise-level, but keep in mind that some data

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

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

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

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

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

Downloading Initial Conditions

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

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

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

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

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

Registration form for PAMORE web service - https://www.dwd.de/DE/leistungen/pamore/pamore_registrierung.html

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

Further note that the set of data fields of the early months is likely to be incomplete with regard to the initialization procedure that is explained in this tutorial. For example, the surface tile approach (see Section 3.8.11) has been activated no earlier than December 2015.

Data retrieval with PAMORE via command-line. PAMORE (*PA*rallel *MO*del data *RE*trieve from Oracle databases) is a high-level tool for the retrieval of (model) data from DWD's meteorological data management system SKY.

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

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

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

Uninitialized analysis for IAU (deterministic)

Global domain with $13 \,\mathrm{km}$ grid spacing

pamore -d 2019020100 -lt m -iglo_startdata -iau

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

pamore -d 2019020100 -lt m -iglo_eu_startdata -iau

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

Uninitialized analysis (deterministic)

Global domain with $13 \,\mathrm{km}$ grid spacing

pamore -d date -lt m -iglo_startdata

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

pamore -d date -lt m -iglo_eu_startdata

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

Initialized analysis (deterministic)

Global domain with 13 km grid spacing

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

Nested EU domain (6.5 km):

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

ICON-D2 limited area domain (2.1 km):

pamore -d date -model ilam -iles_startdata_0

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

Important note for ensemble products: With the additional options

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

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

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

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

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





After submitting your database request, the data will be extracted from the database and stored on an FTP server for download. Once your request has been processed, you will receive an e-mail with information about the FTP server address and the path to your data.

2.2.2. Obtaining ECMWF IFS Initial Data

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

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

https://software.ecmwf.int/wiki/display/UDOC/MARS+user+documentation

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

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

icon/scripts/preprocessing/mars4icon_smi

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

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

Further options are shown by typing ./mars4icon_smi -h

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

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

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

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

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

When initializing from "external" analysis files, ICON requires the soil moisture index SMI and not the volumetric soil moisture content SWV as input. The conversion from SWV to SMI

is performed as part of the MARS request (mars4icon_smi). However, this conversion is not reflected in the variable short names. The fields containing SMI are still named SWVLx, with x denoting the surface layer index. The ICON model, however, expects them to be named SMIx. Therefore, the proper output name SMIx must be specified explicitly in the namelist input_field_nml of iconremap (see the example namelist on p. 45).

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

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

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

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

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

2.2.3. Remapping Initial Data to Your Target Grid

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

We shortly describe the basic steps of remapping: After the successful download, the analysis data must be interpolated from a regular or triangular grid onto the ICON target grid. To this end, the **iconremap** utility from the DWD ICON Tools will be used in batch mode.

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

```
&remap_nml
in_grid_filename = INPUT_GRID_FILENAME
in filename = INPUT FILENAME GRB
                 = 1
                                            ! 1: regular grid, 2: ICON grid
in_type
out_grid_filename = ICON_GRIDFILE
out filename = OUTPUT FILENAME NC
out_type
                  = 2
                                            ! ICON grid
                  = 4
                                            ! NetCDF format
out_filetype
! DEFINITIONS FOR INPUT DATA
ļ
&input_field_nml ! temperature
 inputname = "T"
               = "T"
outputname
&input_field_nml ! horiz. wind comp. U
inputname = "U"
               = "U"
outputname
&input_field_nml ! horiz. wind comp. V
             = "V"
 inputname
               = "V"
outputname
/
. .
```

For each of the variables to be remapped, the script must contain a namelist $input_field_nml$ which specifies details of the interpolation methods and the output name. In this example, the 3D temperature field T and the horizontal wind components U, V are remapped from a regular grid onto a triangular ICON grid³. Variables are usually accessed through their name (character string), but note that for GRIB1 input data the correct field parameter must be provided with the namelist parameter code.

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

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

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

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

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

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

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

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

DWD initial data:

dwd_icon_tools/example/runscripts/create_ic_dwd2icon

IFS initial data:

dwd_icon_tools/example/runscripts/create_ic_ifs2icon

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

Some comments are in order for particular data fields:

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

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

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

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

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

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

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

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

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

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

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

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

2.3. Boundary Data Preparation for ICON-LAM

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

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

Boundary Data Retrieval

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



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

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

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

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

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

Set I (e.g. ICON) or THETA V. DEN, QV, HHL W. QC. QI, QR. QS. VN Set II (e.g. COSMO) U, V, W, Τ, Ρ, QV. QS. HHL QC. QI, QR. Set III (e.g. IFS) U. V. OMEGA, Τ. LNSP. QC, FΙ QV, QI, QR, QS.

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

Lateral boundary conditions from deterministic ICON forecasts

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

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

Use -model iglo, for retrieving global 13 km forecast data, -model ieu for 6.5 km forecast data on the ICON-EU domain, and -model ilam for 2.1 km forecast data on the ICON-D2 domain⁴. The meaning of the remaining command line arguments is as follows:

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

Example: The following example retrieves lateral boundary conditions from the ICON-EU domain for a time range of 36 hours and a temporal resolution of 2 hours, starting at 2020092212:

pamore -d 2020092212 -hstart 0 -hstop 36 -hinc 2 -model ieu -ilam_boundary

⁴If you run into problems, the command pamore -d date -model ilam -iles_boundary might serve as a fallback option for ICON-D2 boundary data.



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

Lateral boundary conditions from ICON's assimilation cycle

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

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

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

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

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

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

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

ICON-LAM Pre-Processing Script

Here we describe the individual steps of the interpolation onto the boundary zone of a limited area grid. This sequence is carried out, for example, in the practical exercises of the ICON Workshop. However, it can also be found summarized in an example script of the DWD ICON tools, namely the run script create_lbc_dwd2icon in the directory dwd_icon_tools/icontools, which processes a whole directory of raw data files (hereafter referred to as the script variable "DATADIR"). The create_lbc_dwd2icon script can be submitted to the PBS batch system of DWD's NEC SX-Aurora. In order to run it on other machines, the batch system header and mpirun command must be adapted accordingly.

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

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

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

Step 1: Extract Boundary Region from the Local Grid File

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



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

We use the **iconsub** program from the collection of ICON Tools, see Section 1.3, with the following namelist:

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

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

The width of the extracted boundary strip in terms of cell rows is specified by the namelist parameters min_refin_c_ctrl and max_refin_c_ctrl. The maximum allowed value for max_refin_c_ctrl is given by max(refin_c_ctrl) == bdy_indexing_depth, i.e. the boundary indexing depth that has been chosen when generating the limited area grid (see Section 2.1.5). The width of the extracted boundary strip must be equal or larger than the width of the lateral nudging zone in the limited-area run for which it is foreseen. A safe setting would be max_refin_c_ctrl = max(refin_c_ctrl), as used in this example.



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

Reference to the original grid: For a given boundary region file grid_file_lbc.nc the question may arise, which original grid was used for its creation. This is analogous to the explanation in Section 2.1.8: The boundary region files contain a link to the original grid file in the form of a unique fingerprint uuidOfOriginalHGrid.

Step 2: Creating Boundary Data

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

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

```
&remap_nml
    in_grid_filename = "${INGRID}"
```

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

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

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

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

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

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

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

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

2.4. External Parameter Files

External parameter fields describe properties of the Earth's surface and atmosphere, which can be assumed to be invariant during the course of a typical NWP forecast (i.e. a couple of days). Examples are the topography, the land-sea mask, the soil type and atmospheric _

aerosols. Most of the fields are constant in time while some are available on a monthly basis in order to represent the seasonal cycle. They are read by the model during startup. The full list of external parameter fields is given in Table 2.3.

Table 2.3.: External parameter fields which are requested by ICON during startup (in
alphabetical order). Fields marked in blue are not read by ICON in opera-
tional NWP runs. In general they are only requested, if the respective depicted
namelist parameter is set.

shortName	Description
AER_SS12	Sea salt aerosol climatology (monthly mean)
	$\texttt{irad_aero=6,9} $ $(\texttt{namelist radiation_nml})$
AER_DUST12	Total soil dust aerosol climatology (monthly mean)
	$irad_aero=6,9 \ (namelist radiation_nml)$
AER_ORG12	Organic aerosol climatology (monthly mean)
	$irad_aero=6,9 \ (namelist radiation_nml)$
AER_SO412	Total sulfate aerosol climatology (monthly mean) irad aero=6.9 (namelist radiation nml)
AER BC12	Black carbon aerosol climatology (monthly mean)
	irad aero=6.9 (namelist radiation nml)
ALB DIF12	Shortwave $(0.3 - 5.0 \mu\text{m})$ albedo for diffuse radiation (monthly mean)
	albedo type=2 (namelist radiation nml)
ALB UV12	$UV_{visible} (0.3 - 0.7 \mu m)$ albedo for diffuse radiation (monthly mean)
$\text{ALD}_{0} \text{V12}$	$0.5 - 0.1 \mu \text{m}$ abed for unuse radiation (monthly mean)
ALP NI19	Near infrared $(0.7 - 5.0 \mu\text{m})$ albedo for diffuse radiation (monthly mean)
ALD_N112	Near initiated $(0.7 - 5.0 \mu\text{m})$ arbedo for diffuse radiation (monting mean)
DEDTU IV	Lake depth
DEI III_LK	Lake depth
EMIC DAD	Surface longways (thermal) emissivity
LMIS_RAD	itura lasmian (namalist suture mil)
TEMICO	Surface longroups (thermal) emissivity derived from establite measure
EMIDO	Surface longwave (therman) emissivity derived from satellite measure-
	ments (monthly mean)
	itype_Iwemiss=2 (namelist extpar_nml)
FOR_D	Fraction of deciduous forest
	ntiles=1 (namelist lnd_nml)
FOR_E	Fraction of evergreen forest
	ntiles=1 (namelist lnd_nml)
FR_LAKE	Lake fraction (fresh water)
FR_LAND	Land fraction (excluding lake fraction but including glacier fraction)
FR_LUC	Land-use class fraction
TATE	
HSURF	Topographic height at cell centers
LAI_MX	Leaf area index in the vegetation phase
	ntiles=1 (namelist lnd_nml)
NDVI_MAX	Normalized differential vegetation index
NDVI_MRAT	Proportion of monthly mean NDVI to yearly maximum (monthly mean)

Continued on next page

PLCOV_MX	Plant covering degree in the vegetation phase
	<pre>ntiles=1 (namelist lnd_nml)</pre>
ROOTDP	Root depth
	<pre>ntiles=1 (namelist lnd_nml)</pre>
RSMIN	Minimum stomatal resistance
	<pre>ntiles=1 (namelist lnd_nml)</pre>
SOILTYP	Soil type
~~~~~~~	
SSO_STDH	Standard deviation of sub-grid scale orographic height
SSO_THETA	Principal axis-angle of sub-grid scale orography
SSO GAMMA	Horizontal anisotropy of sub-grid scale orography
	Torizonal anisotropy of sub-grid scale orography
SSO SIGMA	Average slope of sub-grid scale orography
—	
$T_2M_CL$	Climatological 2m temperature (serves as lower boundary condition for
	soil model)
$T_2M_CLIM$	Climatological 2m temperature (monthly mean)
	$\texttt{itype_vegetation_cycle>1} (namelist \texttt{extpar_nml})$
TOPO_CLIM	Interpolated topographic height for T_2M_CLIM data
	<pre>itype_vegetation_cycle&gt;1 (namelist extpar_nml)</pre>
T_SEA	Sea surface temperature climatology (monthly mean)
	<pre>sstice_mode=2 (namelist lnd_nml)</pre>
ZO	Surface roughness length (over land), containing a contribution from
	subgrid-scale orography
	<pre>itype_z0=1 (namelist nwp_phy_nml)</pre>

 Table 2.3.: Continued from previous page

# 2.4.1. ExtPar Software

The ExtPar software (ExtPar – External Parameters for numerical weather prediction and climate application) is able to generate external parameters for the different models GME, COSMO, HRM and ICON. Experienced users can run ExtPar on UNIX or Linux systems to transform raw data from various sources into domain-specific data files. For ICON, ExtPar will output the fields given in Table 2.3 in the NetCDF file format and GRIB2 on the native triangular grid. For a more detailed overview of ExtPar, the reader is referred to the *User and Implementation Guide* of ExtPar, Asensio and Messmer (2014), and, additionally Smiatek et al. (2008, 2016).

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

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

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

#### 2.4.2. Topography Information

Among various other fields, the external parameter files provide topography information, see Table 2.3. The HSURF dataset contains the geometric height of the earths surface above sea level (unit: m), where the raw data of the terrain model used (GLOBE, ASTER) is aggregated over the grid box/triangle and the aggregated value is assigned to the triangle center point. Therefore, HSURF is *not* identical to the specific mean sea level height of the point lying under the center of the triangle or the maximum altitude of the area lying under the triangle (as e.g. in aeronautical charts).

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

#### 2.4.3. Additional Information for Surface Tiles

ExtPar data files are available for download with and without additional information for surface tiles. See Section 3.8.11 for details on the tile approach.

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

The data files do not differ in the number or type of fields, but rather in the way some fields are defined near coastal regions. Without the _tiles suffix, various surface parameters (e.g. SOILTYP, NDVI_MAX) are only defined at so-called dominant land points, i.e. at grid elements where the land fraction exceeds 50%. With the _tiles suffix, however, these parameters are additionally defined at cells where the land fraction is below 50%. By this, we allow for mixed water-land points. The same holds for the lake depth (DEPTH_LK) which is required by the lake parameterization scheme FLake. For files without the _tiles suffix, DEPTH_LK is only defined at dominant lake points.

# 2.4.4. Parameter Files for Radiation

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

## ecRad

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

## RRTM

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

#### rrtmg_lw.nc

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

#### ECHAM6_CldOptProps.nc

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

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

# 3. Model Description

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

Enrico Fermi

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

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

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



Figure 3.1.: ICON's model structure. This flow chart will be revisited in detail in Fig. 3.8.

# 3.1. Governing Equations

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

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

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

$$oldsymbol{v}_b = rac{\sum_k 
ho_k oldsymbol{v}_k}{\sum_k 
ho_k}\,,$$

with the partial density  $\rho_k$  of constituent k and its advective velocity  $v_k$ . For simplicity,  $v_b$  will be denoted as v in the following.

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

 $\phi = \widehat{\phi} + \phi'' \,,$ 

with

$$\widehat{\phi} = \frac{\overline{\rho \phi}}{\overline{\rho}}$$

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

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

$$\frac{\partial \widehat{v}_n}{\partial t} + \frac{\partial \widehat{K}_h}{\partial n} + (\widehat{\zeta} + f)\widehat{v}_t + \widehat{w}\frac{\partial \widehat{v}_n}{\partial z} = -c_{pd}\widehat{\theta}_v\frac{\partial\overline{\pi}}{\partial n} - F(v_n)$$
(3.1)

$$\frac{\partial \widehat{w}}{\partial t} + \widehat{v}_h \cdot \nabla \widehat{w} + \widehat{w} \frac{\partial \widehat{w}}{\partial z} = -c_{pd} \widehat{\theta}_v \frac{\partial \overline{\pi}}{\partial z} - g \tag{3.2}$$

$$\frac{c_{vd}c_{pd}}{R_d}\overline{\rho}\widehat{\theta}_v\frac{\partial\overline{\pi}}{\partial t} = c_{pd}\overline{\pi}\frac{\partial\overline{\rho}\theta_v}{\partial t} = -c_{pd}\overline{\pi}\nabla\cdot(\overline{\rho}\widehat{\boldsymbol{v}}\widehat{\theta}_v) + \overline{Q}$$
(3.3)

$$\frac{\partial \overline{\rho}}{\partial t} + \nabla \cdot (\overline{\rho} \widehat{\boldsymbol{v}}) = \sum_{k} \overline{\sigma}_{k}^{conv}$$
(3.4)

$$\frac{\partial \overline{\rho} \widehat{q}_k}{\partial t} + \nabla \cdot \left( \overline{\rho} \widehat{q}_k \widehat{\boldsymbol{v}} \right) = -\nabla \cdot \left( \overline{J}_k^z \boldsymbol{k} + \overline{\rho} \overline{q}_k'' \boldsymbol{v''} \right) + \overline{\sigma}_k \qquad (3.5)$$

Prognostic equations are solved for the horizontal velocity component normal to the triangle edges  $\hat{v}_n$  (3.1), the vertical wind component  $\hat{w}$  (3.2), virtual potential temperature  $\hat{\theta}_v$  (3.3), the total density of the air mixture  $\bar{\rho}$  (3.4), with

$$\overline{\rho} = \sum_{k} \overline{\rho}_{k} \,, \tag{3.6}$$

and mass fractions (3.5)

$$\widehat{q}_k = \widehat{\rho}_k / \overline{\rho} \,, \tag{3.7}$$

with the exception of the dry air mass fraction  $q_d$ , which is diagnostic. The index  $k \in \{d, v, c, i, r, s, g\}$  represents a specific constituent of the mixture. We use

k = d	for dry air,
k = v	for water vapor,
k = c	for cloud water,
k = i	for cloud ice,
k = r	for rain,
k = s	for snow, and
k = g	for graupel.

Further explanation of symbols and variables is given in Table 3.1. Note that the corresponding data structures containing the physics and dynamics variables are outlined in Section 9.2.

The equation system (3.1)-(3.5) is supplemented by the lower boundary conditions

$$\widehat{w}|_{s} = \frac{\overline{E}_{v} - \sum_{k_{prec}} \overline{S}_{k}|_{s}}{\overline{\rho}|_{s} - \sum_{k_{prec}} \overline{\rho}_{k}|_{s}}$$
(3.8)  
$$\overline{J}_{k}^{z}|_{s} = \begin{cases} \overline{E}_{k} - \overline{\rho}_{k}\widehat{w}|_{s}, & \text{if } k \equiv v \\ -\overline{\rho}_{k}\widehat{w}|_{s}, & \text{if } k \equiv \text{non-prec. constituent} \\ -\overline{S}_{k}|_{s}, & \text{if } k \equiv \text{prec. constituent} \end{cases}$$

and the equation of state

$$\overline{p} = R_d \,\overline{\rho} \,\widehat{T} \,(1+\alpha) \;,$$

with

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

When expressed in terms of the prognostic variables, the equation of state reads

$$\overline{\pi} = \left(\frac{R_d \overline{\rho} \widehat{\theta}_v}{p_{00}}\right)^{\frac{R_d}{c_{vd}}}$$
(3.9)

In contrast to the original formulation by Gassmann and Herzog (2008), we make use of the two-dimensional rather than the three-dimensional Lamb transformation to convert

	Symbol	Description
$\frac{\partial}{\partial n}$		horizontal derivative in edge-normal direction
$\widehat{K}_h$	$= 0.5 \left( \hat{v}_n^2 + \hat{v}_t^2 \right)$	horizontal component of the kinetic energy
$\widehat{\zeta}$	$= ( abla  imes \widehat{oldsymbol{v}}) \cdot oldsymbol{k}$	vertical component of relative vorticity
f	$= 2\Omega\sin\phi$	Coriolis parameter
$\pi$		Exner function
$c_{pd}, c_{vd}$		specific heat capacity for dry air at constant pressure/volume
$F(v_n)$		turbulent momentum fluxes
g		acceleration of gravity
$\overline{Q}$		diabatic heat source
$\overline{J}_k^z$	$=\overline{\rho}_{k}\left(\widehat{w}_{k}-\widehat{w}\right)$	vertical diffusion flux for constituent $\boldsymbol{k}$
$\overline{\sigma}_k$		internal conversion rate for $k$ th constituent (i.e. conversion among different phases or particle forms)
$\overline{\sigma}_k^{conv}$		internal conversion rate for $k$ th constituent due to convection only
$\overline{ ho q_k'' oldsymbol{v''}}$		turbulent flux of $k$ th partial mass fraction
$\overline{E}_v$	$=\overline{ ho q_v''m v''} _s$	surface evaporation flux
$\overline{S}_k$	$=\overline{ ho}\widehat{q}_k\widehat{v}_k^T$	sedimentation flux of $k$ th constituent
$\widehat{v}^T$		terminal fall velocity of $k$ th constituent

Table 3.1.: Explanation of symbols in the model equations

the nonlinear momentum advection term into a vector-invariant form. Vector-invariant means that no gradients of vectors appear in this equation, which avoids derivatives of the coordinate basis that would otherwise arise in an arbitrary coordinate frame from the nonlinear momentum advection term.

Note that we do not explicitly solve a prognostic equation for the density of dry air. From Equation (3.6) it becomes clear that the partial density of one constituent (here  $\overline{\rho}_d$ ) can be diagnosed, given that a prognostic equation for the total density and all but one partial densities is solved. The reconstructed tangential velocity component is denoted as  $\hat{v}_t$ , and in accordance with the model code, it is assumed here that  $(\hat{v}_t, \hat{v}_n, \hat{w})$  form a right-handed system.

From the Equations (3.4), (3.5) for total density and partial densities some important constraints can be derived which must hold in the discretized analogue in order to achieve mass conservation. First of all, the total density is defined as the sum of all partial densities, as shown by Eq. (3.6). From Eq. (3.7) it follows that

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

Likewise, the prognostic equation for total density (3.4) should be obtained by summing the budget equations (3.5) of all constituents. As a consequence the following constraints hold:

$$\sum_{k} \overline{J}_{k} = 0, \qquad \sum_{k} \overline{\rho} q_{k}^{\prime \prime} v^{\prime \prime} = 0.$$

# 3.2. The Model Reference State

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

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

$$\psi(\lambda, \phi, z, t) = \psi_0(z) + \psi'(\lambda, \phi, z, t)$$

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

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

The background state components  $\rho_0$ ,  $\pi_0$ , and  $\theta_{v0}$  are related by the equation of state (3.9) and are hydrostatically balanced, i.e.

$$\pi_0 = \left(\frac{R_d \rho_0 \theta_{v0}}{p_{00}}\right)^{\frac{R_d}{c_{vd}}}$$
$$\frac{\mathrm{d}\pi_0}{\mathrm{d}z} = -\frac{g}{c_{pd}\theta_{v0}} \tag{3.10}$$

The actual vertical reference profiles can be obtained by integration of Eq. (3.10) given that suitable boundary values are provided. The reference state in ICON is identical to the state used by the COSMO model.

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

$$c_{pd} heta_v rac{\partial \overline{\pi}}{\partial n} = c_{pd} heta_v rac{\partial \pi'}{\partial n}$$
 .

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

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

With the base state at hand, the vertical acceleration due to the pressure gradient and gravity in Eq. (3.2) is rewritten as

$$-c_{pd}\widehat{\theta}_{v}\frac{\partial\overline{\pi}}{\partial z} - g = -c_{pd}\left(\theta_{v0} + \theta'_{v}\right)\frac{\partial\left(\pi_{0} + \pi'\right)}{\partial z}$$
$$= -c_{pd}\left(\theta_{v}\frac{\partial\pi'}{\partial z} + \theta'_{v}\frac{d\pi_{0}}{dz}\right) - c_{pd}\theta_{v0}\frac{d\pi_{0}}{dz} - g$$
$$\stackrel{(3.10)}{=} -c_{pd}\left(\theta_{v}\frac{\partial\pi'}{\partial z} + \theta'_{v}\frac{d\pi_{0}}{dz}\right)$$

The vertical momentum equation finally reads

$$\frac{\partial \widehat{w}}{\partial t} + \widehat{v}_h \cdot \nabla \widehat{w} + \widehat{w} \frac{\partial \widehat{w}}{\partial z} = -c_{pd} \left( \theta_v \frac{\partial \pi'}{\partial z} + \theta'_v \frac{\mathrm{d}\pi_0}{\mathrm{d}z} \right) \,.$$

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

# 3.3. Simplifying Assumptions in the Recent Model Version

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

The term  $\sum_k \overline{\sigma}_k^{conv}$  on the r.h.s. of Eq. (3.4), which describes the net mass tendency of the convection parameterization, is neglected. Thus the model neither accounts for mass loss due to convective precipitation, nor mass re-distribution due to convective motions. The mass continuity equation takes the form

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

At the surface, the boundary condition for  $\hat{w}$ , Eq. (3.8), is approximated as

$$\overline{\rho}\widehat{w}|_{s} = \sum_{k} \overline{\rho}_{k}\widehat{w}_{k}|_{s} = 0, \qquad (3.12)$$

which in terrain-following coordinates translates to  $\overline{\rho}\hat{w}|_s = \overline{\rho}\hat{v} \cdot \nabla h$ . *h* denotes the topography height. Thus, the simulated atmospheric system is assumed to be closed w.r.t. total

mass. The sedimentation fluxes at the surface  $\overline{S_k}|_s$  as well as the surface evaporation flux  $\overline{E_v}|_s$ , which appear in the partial mass continuity equations, are retained. In order for Eq. (3.12) to hold, the (implicit) assumption is that the corresponding mass loss/gain due to sedimentation/evaporation is compensated by a fictitious flux of dry air across the surface.

Away from the surface, the diffusion fluxes of all airborne constituents (except for dry air) are neglected. The diffusive fluxes of all precipitating constituents, however, are taken into account. The continuity equation for the total mass is used in the form (3.11). Since Eq. (3.11) only holds if the constraint  $\sum_k \overline{J}_k^z = 0$  holds, again, the (implicit) assumption made is that a fictitious diffusion flux of dry air counteracts the sedimentation fluxes such that the total mass in a volume moving with the barycentric velocity is conserved.

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

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

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

$$\begin{split} \frac{\partial}{\partial t} \int_{\overline{p}_s}^{0} \mathrm{d}p &= -g \frac{\partial}{\partial t} \int_{z_s}^{\infty} \overline{\rho} \, \mathrm{d}z \\ &\frac{\partial \overline{p}_s}{\partial t} = g \int_{z_s}^{\infty} \frac{\partial \overline{\rho}}{\partial t} \, \mathrm{d}z \\ &\frac{\partial \overline{p}_s}{\partial t} = -g \int_{z_s}^{\infty} \nabla \cdot (\overline{\rho} \widehat{v}) - \sum_k \overline{\sigma}_k^{conv} \, \mathrm{d}z \\ &\frac{\partial \overline{p}_s}{\partial t} = -g \int_{z_s}^{\infty} \nabla_h \cdot (\overline{\rho} \widehat{v}_h) \, \mathrm{d}z + g \, (\overline{\rho} w) \, |_s + g \overline{P}^{conv}|_s \\ &\frac{\partial \overline{p}_s}{\partial t} = -g \int_{z_s}^{\infty} \nabla_h \cdot (\overline{\rho} \widehat{v}_h) \, \mathrm{d}z + g \frac{\overline{\rho}|_s}{\overline{\rho}|_s - \sum_{k \overline{prec}} \overline{\rho}_k|_s} \left( \overline{E_v} - \sum_{k \overline{prec}} \overline{S}_k|_s \right) + g \overline{P}^{conv}|_s \, , \end{split}$$

with the convective surface precipitation flux

$$\overline{P}^{conv}|_s = \int_{z_s}^{\infty} \sum_k \overline{\sigma}_k^{conv} \, \mathrm{d}z \, .$$

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



Figure 3.2.: Illustration of ICON's vertical levels. With num_lev layers, there are num_lev + 1 so-called *half levels*. The half levels k - 1/2, k + 1/2 enclose layer k at the centers of which are the corresponding full levels k, for  $k = 1, ..., num_lev$ . Layer 1 is at the top of the atmosphere and layer n at the bottom of the atmosphere. Half level num_lev + 1 coincides with the Earth's surface.

# 3.4. Vertical Coordinates

In a nonhydrostatic model, it cannot be taken for granted that the pressure is monotonously decreasing with increasing altitude. Moreover, the pressure at a certain point does not necessarily represent the mass of the air column above, as it is the case for a hydrostatic model. For that reason, a vertical coordinate in terms of geometric altitude is preferred over a pressure-based coordinate in many nonhydrostatic models.

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

When setting up an ICON simulation, the total number of vertical levels has to be specified for each domain via
#### num_lev (namelist run_nml, list of integer value)

Comma-separated list of integer values giving the number of vertical full levels for each domain.

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

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

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

This indirect *reference grid* approach raises the same questions that played a role in the handling of the horizontal grid, see Section 2.1.8: In order to find out if identical vertical coordinate options were used for two given data sets, the GRIB2 data records contain special meta-data items, namely numberOfVGridUsed and uuidOfVGrid.

# 3.4.1. Terrain-following Hybrid Gal-Chen Coordinate

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

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

where z represents the height of the coordinate surfaces defined by  $\eta$ , 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 terrainfollowing 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

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

such that coordinate surfaces become constant height surfaces above  $z = z_{flat}$ . Oftentimes, Equation (3.13) is also written in the discretized form

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

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

#### Configuring the Hybrid Gal-Chen Coordinate

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

ivctype = 1 (namelist nonhydrostatic_nml, integer value)

The user has to provide the vertical coordinate table (vct) as an input file, using the namelist variable vct_filename in the namelist grid_nml. The table consists of the A and B values (see Equation (3.14)) from which the half level heights  $z_h(x, y, k)$  can be deduced. A(k)[m] contains fixed height values, with A(1) defining the model top height H and  $A(\text{num_lev} + 1) = 0$  m. The dimensionless values B(k) control the vertical decay of the topography signal, with B(1) = 0 and  $B(\text{num_lev} + 1) = 1$ . Thus,  $z_h(x, y, 1)$  is equivalent to the model top height, while  $z_h(x, y, \text{num_lev} + 1)$  is the surface height.

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

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

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

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

# 3.4.2. SLEVE Coordinate

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

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

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

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

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

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

#### **Configuring the SLEVE Coordinate**

The main switch for selecting the SLEVE vertical coordinate is

ivctype = 2 (namelist nonhydrostatic_nml, integer value)

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

#### Namelist sleve nml:

#### 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  $\leq 0$ . The layer thickness is then determined as top_height/num_lev. Control output of the vertical layer distribution is written to stderr.

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.

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

# 3.5. Temporal Discretization

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

#### 3.5.1. Basic Idea

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

$$\frac{\mathrm{d}\psi(\vec{x},t)}{\mathrm{d}t} = F(\psi(\vec{x},t),t)\,. \tag{3.15}$$

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

Let  $t_{n-m}$  denote some time in the past and  $t_{n+1}$  some time in the future. Now we integrate Eq. (3.15) from time  $t_{n-m}$  to  $t_{n+1}$ , which gives

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

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

$$\frac{\psi(\vec{x}, t_{n+1}) - \psi(\vec{x}, t_n)}{\Delta t} = \alpha F_{n+1} + \beta F_n \,. \tag{3.16}$$

The coefficients  $\alpha$  and  $\beta$  must satisfy the so called *consistency condition* 

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

such that the r.h.s. of Eq. (3.16) represents some averaged F. Equation (3.16) represents a whole family of simple two-step schemes. For example, by choosing  $\alpha = 0$ ,  $\beta = 1$  we arrive at the simple Euler forward scheme, while for  $\alpha = 1$ ,  $\beta = 0$  we get the (implicit) Euler backward scheme, both of which are first order accurate. Choosing  $\alpha = \beta = 0.5$  leads to the implicit *trapezoidal* scheme, which is of second order accuracy.

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

predictor: 
$$\psi^{*}(\vec{x}, t_{n+1}) = \psi(\vec{x}, t_{n}) + \Delta t F_{n}$$
  
corrector:  $\psi(\vec{x}, t_{n+1}) = \psi(\vec{x}, t_{n}) + \Delta t \{F_{n+1}^{*}, F_{n}\}_{\alpha}$  (3.17)

Here we have introduced the notation

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

Note that for  $\alpha = 1$ , Equation (3.17) is an imitation of the Euler backward scheme (termed Matsuno scheme, (Matsuno, 1966)), while for  $\alpha = 0.5$ , it is an imitation of the *trapezoidal* scheme (termed *Heun's method*). The Matsuno scheme has first-order accuracy, and Heun's method has second-order accuracy.

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

#### 3.5.2. Implementation Details

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

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

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

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

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

#### **Predictor step:**

$$\frac{v_n^{n+1^*} - v_n^n}{\Delta t} = -\operatorname{adv}(v_n^n) - c_{pd}\theta_v^n \frac{\partial \pi'^{n}}{\partial n} + F(v_n^n)$$
(3.18)

$$\frac{w^{n+1^*} - w^n}{\Delta t} = -\operatorname{adv}(w^n) - c_{pd}\theta''_v \frac{\mathrm{d}\pi_0}{\mathrm{d}z} - c_{pd}\theta^n_v \left\{\frac{\partial \pi'^{n+1^*}}{\partial z}, \frac{\partial \pi'^{n}}{\partial z}\right\}_\eta$$
(3.19)

$$\frac{\rho^{n+1^*} - \rho^n}{\Delta t} = -\nabla_h \cdot \left( v_n^{n+1^*} \rho^n \right) - \frac{\partial}{\partial z} \left[ \left\{ w^{n+1^*}, w^n \right\}_{\eta} \rho^n \right] \\
\frac{\pi^{n+1^*} - \pi^n}{\Delta t} = -\frac{R_d}{c_{vd}} \left( \frac{\pi^n}{\rho^n \theta_v^n} \right) \left[ \nabla_h \cdot \left( v_n^{n+1^*} \rho^n \theta_v^n \right) + \frac{\partial}{\partial z} \left[ \left\{ w^{n+1^*}, w^n \right\}_{\eta} \rho^n \theta_v^n \right] \right] + Q^n \quad (3.20)$$

with

$$\operatorname{adv}(v_n^n) = \frac{\partial K_h^n}{\partial n} + \left(\zeta^n + f\right) v_t^n + w^n \frac{\partial v_n^n}{\partial z}$$
$$\operatorname{adv}(w^n) = \boldsymbol{v}_h^n \cdot \nabla w^n + w^n \frac{\partial w^n}{\partial z} \,.$$

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

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

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

and the default value is  $vwind_offctr = 0.15$ .

**Corrector step:** 

$$\frac{v_n^{n+1} - v_n^n}{\Delta t} = -\left\{ \operatorname{adv}(v_n^{n+1^*}), \operatorname{adv}(v_n^n) \right\}_{\alpha} \\
- c_{pd} \left\{ \theta_v^{n+1^*}, \theta_v^n \right\}_{\delta} \left\{ \frac{\partial \pi'^{n+1^*}}{\partial n}, \frac{\partial \pi'^{n}}{\partial n} \right\}_{\delta} \\
- F_d(v_n^{n+1^*}) + F(v_n^n) \tag{3.21}$$

$$\frac{w^{n+1} - w^n}{\Delta t} = -\left\{ \operatorname{adv}(w^{n+1^*}), \operatorname{adv}(w^n) \right\}_{\alpha} \\
- c_{pd} \left\{ \theta_v^{(n+1^*)}, \theta_v^{(n)} \right\}_{\delta} \left\{ \frac{\partial \pi'^{(n+1)}}{\partial z}, \frac{\partial \pi'^{(n)}}{\partial z} \right\}_{\eta} \\
- c_{pd} \left\{ \theta_v^{n+1^*}, \theta_v^n \right\}_{\delta} \left\{ \frac{\partial \pi'^{(n+1)}}{\partial z}, \frac{\partial \pi'^{(n)}}{\partial z} \right\}_{\eta} \\
\frac{\rho^{n+1} - \rho^n}{\Delta t} = -\nabla_h \cdot (v_n^{n+1}\rho^n) - \left[ \frac{\partial}{\partial z} \left[ \left\{ w^{n+1}, w^n \right\}_{\eta} \left\{ \rho^{n+1^*}, \rho^n \right\}_{\delta} \right] \\
\frac{\pi^{n+1} - \pi^n}{\Delta t} = -\frac{R_d}{c_{vd}} \left( \frac{\pi^n}{\rho^n \theta_v^n} \right) \left[ \nabla_h \cdot (v_n^{n+1}\rho^n \theta_v^n) \\
+ \left[ \frac{\partial}{\partial z} \left[ \left\{ w^{n+1}, w^n \right\}_{\eta} \left\{ \rho^{n+1^*}, \rho^n \right\}_{\delta} \left\{ \theta_v^{n+1^*}, \theta_v^n \right\}_{\delta} \right] \right] \\
+ Q^n$$

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

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

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

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

Once the updated state vector

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

is known, the virtual potential temperature  $\theta_v^{\text{new}}$  is diagnosed from the linearized equation of state (3.9)

$$\frac{\pi^{\text{new}} - \pi^n}{\pi^n} = \frac{R_d}{c_{vd}} \frac{(\rho\theta_v)^{\text{new}} - (\rho\theta_v)^n}{(\rho\theta_v)^n} \,. \tag{3.22}$$

The rationale behind that is as follows: Multiplying (3.22) by  $\Delta t^{-1}$  and rearranging yields

$$\frac{(\rho\theta_v)^{\text{new}} - (\rho\theta_v)^n}{\Delta t} = \frac{c_{vd}}{R_d} \left(\frac{\rho^n \theta_v^n}{\pi^n}\right) \frac{\pi^{\text{new}} - \pi^n}{\Delta t} \,. \tag{3.23}$$

Thus, diagnosing  $\theta_v^{\text{new}}$  from (3.22) can be interpreted as solving two prognostic equations for the two thermodynamic variables  $\pi$  and  $\theta_v$ , instead of one prognostic equation for  $\pi$  and the equation of state. By doing so,  $\rho\theta_v$  is exactly conserved (in the absence of diabatic terms). The two prognostic equations are constrained by (3.23), i.e. the same flux divergence is used. As a side effect, however, the solution at a particular point in time is not constrained by the equation of state (3.9) – only its time evolution is. This approach was first described by Gassmann (2013).

#### **Pragmatic Simplifications**

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

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

Equation $(3.18)$ :	$\operatorname{adv}(v_n^n) \simeq \operatorname{adv}(v_n^{n^*})$
Equation $(3.19)$ :	$\operatorname{adv}(w^n) \simeq \operatorname{adv}(w^{n^*})$

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

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

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

Equation (3.18): 
$$c_{pd}\theta_v^n \frac{\partial \pi'^{,n}}{\partial n} \simeq c_{pd}\theta_v^n \frac{\partial \pi'^{,\tilde{n}}}{\partial n}$$
  
Equation (3.21):  $c_{pd} \left\{ \theta_v^{n+1^*}, \theta_v^n \right\}_{\delta} \left\{ \frac{\partial \pi'^{,n+1^*}}{\partial n}, \frac{\partial \pi'^{,n}}{\partial n} \right\}_{\delta} \simeq c_{pd}\theta_v^n \frac{\partial \pi'^{,\tilde{n}}}{\partial n}$ 

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

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

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

#### Vertically Implicit Solver

The solution to the predictor-corrector scheme described above is mostly straightforward, as the majority of terms are treated in an explicit manner. This, however does not hold for the prognostic equation for vertical wind, since the solution for  $w^{n+1^*}$  depends on  $\pi^{n+1^*}$ , which itself depends on  $w^{n+1^*}$  (see Equations (3.19) and (3.20)).

The overall solution strategy is as follows: First,  $\pi$  is eliminated from Eq. (3.19) by inserting the prognostic equation (3.20). This results in a linear system of equations for the unknown w's in the vertical direction. Once  $w^{n+1^*}$  is known, the Exner equation (3.20) can be solved. The derivation for the corrector step is basically identical and differs only w.r.t. the time levels that are used for  $\rho$  and  $\theta_v$ .

We start with the vertical discretization of (3.19) and (3.20). When using basic centered differences, and noting that  $\partial \pi / \partial t = \partial \pi' / \partial t$  this leads to

$$w_{k+1/2}^{n+1^*} = Z_{k+1/2}^{w \, \text{expl}} - \Delta t c_{pd} \theta_{v,k+1/2}^n \eta \frac{\pi_k^{\prime,n+1^*} - \pi_{k+1}^{\prime,n+1^*}}{\Delta z_{k+1/2}}$$
(3.24)

$$\pi_k^{\prime,n+1^*} = Z_k^{\pi \operatorname{expl}} - \Delta t \frac{R_d}{c_{vd}} \left( \frac{\pi_k^n}{\rho_k^n \theta_{v,k}^n} \right) \, \eta \frac{(w^{n+1^*} \rho^n \theta_v^n)_{k-1/2} - (w^{n+1^*} \rho^n \theta_v^n)_{k+1/2}}{\Delta z_k} \,, \quad (3.25)$$

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

$$Z_{k+1/2}^{w \exp l} = w_{k+1/2}^n - \Delta t \left[ \operatorname{adv}(w_{n^*})_{k+1/2} + c_{pd} \theta_{v,k+1/2}^{\prime,n} \frac{\mathrm{d}\pi_0}{\mathrm{d}z} \right|_{k+1/2} + c_{pd} \theta_{v,k+1/2}^n (1-\eta) \frac{\pi_k^{\prime,n} - \pi_{k+1}^{\prime,n}}{\Delta z_{k+1/2}} \right]$$

and

$$Z_k^{\pi \operatorname{expl}} = \pi_k^{\prime,n} - \Delta t \frac{R_d}{c_{vd}} \left( \frac{\pi_k^n}{\rho_k^n \theta_{v,k}^n} \right) \left[ \nabla_h \cdot \left( v^{n+1^*} \rho^n \theta_v^n \right)_k + (1-\eta) \frac{(w^n \rho^n \theta_v^n)_{k-1/2} - (w^n \rho^n \theta_v^n)_{k+1/2}}{\Delta z_k} \right] + \Delta t \, Q_n^k$$

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

As an intermediate step, we compute  $\pi_k^{\prime,n+1^*} - \pi_{k+1}^{\prime,n+1^*}$  from Eq. (3.25):

$$\pi_{k}^{\prime,n+1^{*}} - \pi_{k+1}^{\prime,n+1^{*}} = Z_{k}^{\pi} \exp \left[ -Z_{k+1}^{\pi} - \frac{\Delta t R_{d}}{c_{vd}} \eta \left[ (w^{n+1^{*}} \rho^{n} \theta_{v}^{n})_{k-1/2} \frac{\Gamma_{k}^{n}}{\Delta z_{k}} - (w^{n+1^{*}} \rho^{n} \theta_{v}^{n})_{k+1/2} \left( \frac{\Gamma_{k}^{n}}{\Delta z_{k}} + \frac{\Gamma_{k+1}^{n}}{\Delta z_{k+1}} \right) + (w^{n+1^{*}} \rho^{n} \theta_{v}^{n})_{k+3/2} \frac{\Gamma_{k+1}^{n}}{\Delta z_{k+1}} \right].$$
(3.26)

Here we have introduced  $\Gamma_k^n$  as an abbreviation for

$$\Gamma_k^n = rac{\pi_k^n}{
ho_k^n heta_{v,k}^n} \, .$$

Inserting Eq. (3.26) into Eq. (3.24) and collecting terms proportional to  $w_{k-1/2}^{n+1*}$ ,  $w_{k+1/2}^{n+1*}$ ,  $w_{k+3/2}^{n+1*}$  on the left hand side leads to:

$$-w_{k-1/2}^{n+1^{*}} \underbrace{\left[ \Delta t \eta \frac{c_{pd}\theta_{v,k+1/2}^{n}}{\Delta z_{k+1/2}} \Delta t \frac{R_{d}}{c_{vd}} \frac{\Gamma_{k}^{n}}{\Delta z_{k}} \rho_{k-1/2}^{n} \theta_{v,k-1/2}^{n} \eta \right]}_{\text{sub-diagonal}} + w_{k+1/2}^{n+1^{*}} \underbrace{\left[ 1 + \Delta t \eta \frac{c_{pd}\theta_{v,k+1/2}^{n}}{\Delta z_{k+1/2}} \Delta t \frac{R_{d}}{c_{vd}} \left( \frac{\Gamma_{k}^{n}}{\Delta z_{k}} + \frac{\Gamma_{k+1}^{n}}{\Delta z_{k+1}} \right) \rho_{k+1/2}^{n} \theta_{v,k+1/2}^{n} \eta \right]}_{\text{main diagonal}} - w_{k+3/2}^{n+1^{*}} \underbrace{\left[ \Delta t \eta \frac{c_{pd}\theta_{v,k+1/2}^{n}}{\Delta z_{k+1/2}} \Delta t \frac{R_{d}}{c_{vd}} \frac{\Gamma_{k+1}^{n}}{\Delta z_{k+1}} \rho_{k+3/2}^{n} \theta_{v,k+3/2}^{n} \eta \right]}_{\text{sup-diagonal}} = Z_{k+1/2}^{w \exp l} - \Delta t \eta \frac{c_{pd}\theta_{v,k+1/2}^{n}}{\Delta z_{k+1/2}} \left( Z_{k}^{\pi} \exp l - Z_{k+1}^{\pi} \right)$$
(3.27)

Equation (3.27) defines a linear system of equations from which the unknown vertical velocities  $w_{k-1/2}^{n+1*}$  can be computed. The system is tridiagonal and can be solved with the Thomas algorithm (Press et al., 2007, p. 57), given that suitable boundary conditions are provided. So far it is assumed that the upper and lower boundary are impermeable w.r.t. to mass, i.e. we set  $w_{1/2}^{n+1*} = w_{nlev+1/2}^{n+1*} = 0$ . In case of vertical nesting, w at the upper boundary is usually nonzero and is interpolated from the parent domain (see Section 3.9.1 for details).

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

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

Note that  $\alpha$  and  $\gamma$  are defined on half levels, while  $\beta$  is defined on full levels. The tridiagonal system (3.27) can now be written in the form  $(a_{1/2} = 0, c_{\text{nlev}+1/2} = 0)$ 

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

or in matrix form

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

with the coefficients

$$a_{k+1/2} = -\gamma_{k+1/2} \,\beta_k \,\alpha_{k-1/2}$$
  

$$b_{k+1/2} = 1 + \gamma_{k+1/2} \,\left(\beta_k + \beta_{k+1}\right) \,\alpha_{k+1/2}$$
  

$$c_{k+1/2} = -\gamma_{k+1/2} \,\beta_{k+1} \,\alpha_{k+3/2}$$
  

$$d_{k+1/2} = Z_{k+1/2}^{w \, \text{expl}} - \gamma_{k+1/2} \left(Z_k^{\pi \, \text{expl}} - Z_{k+1}^{\pi \, \text{expl}}\right)$$

for k = 1, ..., nlev - 1, and the Dirichlet boundary conditions

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

# 3.6. Tracer Transport

The transport module is an important building block of any numerical weather prediction (NWP) or climate model, as it predicts the large-scale redistribution of water substances, chemical constituents or aerosols in the atmosphere due to air motion. Mathematically, it solves one of the fundamental laws of physics, namely the equation of tracer mass continuity (3.5). The transport module itself does not take into account tracer sources or sinks. It only predicts its large scale redistribution. Hence, for each tracer the transport module solves the simplified continuity equation

$$\frac{\partial \overline{\rho} \hat{q}_k}{\partial t} + \nabla \cdot (\overline{\rho} \hat{q}_k \hat{\boldsymbol{v}}) = 0$$
(3.28)

(compare with Eq. (3.5)). Additional sources and sinks as well as turbulent diffusion are accounted for in the physics interface with a fractional step approach (see Section 3.7).

The numerical solution of Eq. (3.28) is based on so-called space-time finite volume methods. By space-time methods we refer to methods where the temporal and spacial discretizations are combined rather than separated. Space-time methods are also known as *cell-integrated*  semi-Lagrangian schemes. As will become clear, such schemes are neither purely semi-Lagrangian, nor Eulerian in the classical sense. They are Eulerian in the sense that the flux of mass through the stationary walls of grid cells is considered. They are, however, semi-Lagrangian in the sense that trajectory calculations are needed for flux computation. In the literature such schemes are sometimes termed *Flux Form Semi-Lagrangian (FFSL)*. The specific implementation in ICON partly builds upon work by Lauritzen et al. (2010, 2011a), Harris and Lauritzen (2010), Skamarock and Menchaca (2010), Miura (2007) for the horizontal and Colella and Woodward (1984), Zerroukat et al. (2006) for the vertical.

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

$$\overline{\psi}_i^n = \frac{1}{\Delta V_i} \iiint_{V_i} \psi(x, y, z, t^n) \,\mathrm{d}V$$

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

# 3.6.1. Directional Splitting

By integrating the continuity equation (3.28) in space over a prismatic grid cell and in time over the time step  $\Delta t$ , a solution to (3.28) can formally be written as

$$\overline{\rho q}_{i,k}^{n+1} = \overline{\rho q}_{i,k}^n + \Delta t \left[ \mathcal{H}(\overline{q}^n) + \mathcal{V}(\overline{q}^n) \right] , \qquad (3.29)$$

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

Instead of solving this somewhat unwieldy equation in one sweep, a fractional step approach is taken in ICON such that separate equations for horizontal and vertical transport are solved consecutively. Of course, replacing equation (3.29) by some approximation involving the two subproblems

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

will inevitably result in a residual error. This error is known as the *splitting error*. On default, the following approximation to (3.29) is used:

$$\overline{\rho q}_{i,k}^* = \overline{\rho q}_{i,k}^n + \Delta t \, \mathcal{V}\left(\overline{q}^n\right) \tag{3.30}$$

$$\overline{\rho q}_{i,k}^{n+1} = \overline{\rho q}_{i,k}^* + \Delta t \,\mathcal{H}\left(\overline{q}^*\right) \tag{3.31}$$

In order to maintain  $\mathcal{O}\left[\Delta t^2\right]$  accuracy, the order of the operators is reversed on alternate time steps. This might be regarded as a poor man's *Strang splitting* (Strang, 1968). Full Strang-splitting of the form  $[\mathcal{V}(\Delta t/2)][\mathcal{H}(\Delta t)][\mathcal{V}(\Delta t/2)]$  has also been tested during the

implementation phase. Except for being more expensive (the vertical operator is called twice per time step) no significant impact on the model results has been noted.

A shortcoming of the splitting (3.30), (3.31) is that it does not preserve an initially uniform tracer field (e.g.  $q(x, y, z, t_0) = 1$ ) in a deformational flow since there is not enough information available to correctly do the conversion  $\overline{\rho q}^* \longrightarrow \overline{q}^*$ . Tempting candidates for this conversion might be  $\overline{\rho}^n$  or  $\overline{\rho}^{n+1}$  as they are readily available from ICON's dynamical core. Any such attempt, however, will not preserve an initially uniform tracer field. In order to do so, it is necessary to keep track of the changes in partial density  $\rho q$  that are solely a result of mass convergence/divergence in the directions of splitting. Therefore we follow the method of Easter (1993) wherein the air mass continuity equation (3.5) is simultaneously reintegrated in the same split manner as the continuity equation for tracer mass.

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

$$\overline{\rho}_{i,k}^{*} = \overline{\rho}_{i,k}^{n} + \Delta t \,\mathcal{V}\left(1\right)$$

$$\overline{q}_{i,k}^{*} = \frac{\overline{\rho}\overline{q}_{i,k}^{*}}{\overline{\rho}_{i,k}^{*}}$$

$$(3.32)$$

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

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

$$\overline{q}_{i,k}^{n+1} = \frac{\overline{\rho}\overline{q}_{i,k}^{n+1}}{\overline{\rho}_{i,k}^{n+1}}$$
(3.33)

Changes in partial density solely due to mass convergence/divergence are corrected for in equations (3.32) and (3.33). The key point here is that the intermediate density  $\bar{\rho}^*$  rather than  $\bar{\rho}^{n+1}$  or  $\bar{\rho}^n$  is used to recover the mass fraction  $\bar{q}^*$  in (3.32). The re-integration of (3.4) (second and fifth equation above) is rather straightforward, as it relies on pre-computed mass fluxes provided by the dynamical core.

### 3.6.2. Horizontal Transport

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

#### **Graphical Interpretation**

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

In order to set the stage, let us first take the Lagrangian viewpoint: Assume that the time dependent velocity field is known analytically such that the trajectories for all the air parcels are known, which terminate at the walls of the Eulerian CV at the new time  $t^{n+1}$ . As an example, trajectories for air parcels terminating at the CV vertices at  $t^{n+1}$  are depicted as gray lines. Accordingly we know the position of these air parcels at time  $t^n$  which we will denote as the starting points. By connecting the starting points we can construct the gray shaded area known as the Lagrangian CV. The latter encompasses all air parcels that are transported into the Eulerian CV (i.e. the grid cell) during the time interval  $[t^n, t^{n+1}]$ . In a standard semi-Lagrangian scheme the key task is to compute an estimate of the Lagrangian CV (gray shaded), followed by a computation of the total tracer mass contained. Then, the solution  $\overline{\rho q_i}^{n+1}$  can easily be deduced from the Lagrangian finite-volume form of the continuity equation (3.28)

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

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

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

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

Mathematically the scheme can be cast into the following flux form:

$$\overline{\rho}\overline{q}_{i}^{n+1} = \overline{\rho}\overline{q}_{i}^{n} - \frac{1}{\Delta A_{i}} \sum_{e=1}^{N_{e}} s_{ie} F_{ie} \quad \text{,with} \quad F_{ie} = \langle \overline{\rho}_{i}^{e} \rangle \iint_{a_{i}^{e}} q^{n}(x,y) \,\mathrm{d}a \,, \qquad (3.34)$$

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

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



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

#### **Basic Algorithm**

The numerical algorithm which solves Eq. (3.34) for a single Eulerian CV proceeds in 4 major stages:

- 1. The flux area  $a_i^e$  for each cell wall is reconstructed by means of backward trajectories.
- 2. For each Eulerian CV the unknown tracer subgrid distribution  $q(x, y, t_0)$  is estimated from the known cell averages  $\overline{q}_i^n$  of the CV itself and surrounding cells. Several polynomial reconstructions, from linear to cubic, are available.
- 3. The total mass  $F_{ie}$  crossing the  $e^{th}$  wall is estimated by numerically evaluating the integral in Eq. (3.34). I.e. the estimated subgrid distribution  $q(x, y, t_0)$  is integrated over the approximated flux area  $a_i^e$  by means of Gauss quadrature.
- 4. The sum on the r.h.s. of Eq. (3.34) is evaluated which leads to the solution  $\overline{\rho q}_i^{n+1}$ .

#### 3.6.3. Vertical Transport

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

#### Piecewise Parabolic Method (PPM)

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



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

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

$$q(\zeta, k) = a_0 + a_1\zeta + a_2\zeta^2$$
, with  $\zeta = \frac{z - z_{k+1/2}}{\Delta z_k}$ . (3.35)

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

$$\int_{0}^{1} q(\zeta, k) d\zeta = \overline{q}_{k}, 
q(\zeta = 1, k) = q_{u} = q_{k-1/2}, 
q(\zeta = 0, k) = q_{l} = q_{k+1/2},$$
(3.36)

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

When applying the constraints (3.36) the parabolic interpolant (3.35) can finally be written as a function of the grid scale variables  $\overline{q}_k$ ,  $q_{k+1/2}$ ,  $q_{k-1/2}$ 

$$q(\zeta,k) = \overline{q}_k - \Delta q_k \left(\frac{1}{2} - \zeta\right) - a_{6,k} \left(\frac{1}{6} - \zeta + \zeta^2\right), \qquad (3.37)$$

with

$$\Delta q_k = q_{k-1/2} - q_{k+1/2}$$
$$a_{6,k} = 6\overline{q}_k - 3q_{k-1/2} - 3q_{k+1/2}$$

The overall accuracy of the parabolic interpolant (3.37) strongly depends on the accuracy to which the edge values  $q_{k\pm 1/2}$  are known. Edge value estimates must be of at least thirdorder in order for the interpolant to exactly reconstruct a parabola. Here we follow Colella and Woodward (1984) and compute fourth-order edge value estimates, as this turned out to be beneficial to the overall accuracy of the scheme (see Lauritzen et al. (2011b, p. 228)).

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

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

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

$$F_{k-1/2} = \frac{1}{\Delta t} \int_{z_{k-1/2}-w_{k-1/2}^{n+1/2}\Delta t}^{z_{k-1/2}} \rho(z)q(z)\mathrm{d}z\,, \quad \text{for} \quad w > 0$$
(3.38)

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

For w > 0 integration of (3.38) finally leads to the time-averaged vertical flux

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

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

In its standard version, the PPM scheme has a Courant number limitation of  $|C| \leq 1$ . It can, however, be easily extended to larger Courant numbers by splitting the computation of the mass fluxes (3.38) into so-called integer and fractional fluxes (Lin and Rood, 1996). In its current form in ICON, the PPM scheme is stable up to |C| = 5.

#### Parabolic Spline Method (PSM)

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

PPM in terms of the edge-value estimate. The edge values are determined by imposing the following additional constraint on the parabola  $q(\zeta, k)$  in (3.35)

$$\frac{1}{\Delta z_{k+1}} \frac{\mathrm{d}q_{k+1}}{\mathrm{d}\zeta} \bigg|_{\zeta=1} = \frac{1}{\Delta z_k} \frac{\mathrm{d}q_k}{\mathrm{d}\zeta} \bigg|_{\zeta=0} , \qquad (3.39)$$

which states that the parabola's first derivative must be continuous at cell edges. Hence, while the PPM parabolas are continuous at cell edges, the PSM parabolas are even continuously differentiable. The latter turns the piecewise parabolic function into a parabolic spline. From the condition (3.39) an implicit equation system for the unknown edge values  $q_{k+1/2}$  can be deduced, which is however beyond the scope of this tutorial (see Zerroukat et al., 2006, Reinert, 2021).

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

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

# 3.6.4. Reduced Calling Frequency

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

Continuity equations for tracers (like water vapour) lack fast wave modes (sound and gravity waves) and, thus, have less restrictive time step limitations. Given the large number of tracers in state of the art climate and NWP models, significant computational cost savings can be obtained by sub-cycling the solution of the density, momentum and thermodynamic equation with respect to the tracer equations. Stated in another way, the tracer equations can be integrated with a much larger time step. In doing so, care has to be taken in order to maintain tracer-mass consistency.

In ICON, the number of times by which the integration of the air mass continuity equation (and the entire dynamical core) is sub-cycled with respect to the tracer mass continuity



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

equations is typically 5 (see Section 3.7.1). In order to maintain tracer-mass consistency, the time-averaged rather than the instantaneous mass flux is passed to the transport module. Thus,  $\langle \overline{\rho}^e_i \rangle$  in Eq. (3.34) can be expressed in terms of the time-averaged horizontal mass flux  $\langle F^m_{ie} \rangle$  as

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

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

#### 3.6.5. Some Practical Advice

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

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

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

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

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

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

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

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

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

- **1**  $1^{st}$  order upwind
- **2** Parabolic Spline Method (PSM)
- **3** Piecewise Parabolic Method (PPM)

#### itype_hlimit (namelist transport_nml, list of Integer values)

Comma separated list of integer values, specifying the type of the horizontal limiter.

The  $i^{th}$  entry corresponds to the  $i^{th}$  tracer in ICON's internal tracer list. Most relevant options are

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

#### itype_vlimit (namelist transport_nml, list of Integer values)

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

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

#### ivlimit_selective (namelist transport_nml, integer value)

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

**0/1** off/on

#### Example Settings for a Standard NWP Run

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

! transport_nml: tracer transport					
&transport_nml					
ihadv_tracer	=	52, 2, 2, 2, 2	! hor. transport selector		
ivadv_tracer	=	3, 3, 3, 3, 3	! vert. transport selector		
itype_hlimit	=	3, 4, 4, 4, 4	! hor. limiter		
itype_vlimit	=	1, 1, 1, 1, 1	! vert. limiter		
<pre>ivlimit_selective /</pre>	=	0	! selective limiting switched off		

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

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

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

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

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

If time to solution is not absolutely critical (e.g. for non-operational, scientific, applications), we recommend to select PSM for vertical transport, together with selective vertical limiting (ivadv_tracer=2, ivlimit_selective=1), as this will slightly improve the accuracy of vertical transport with only marginal computational overhead.

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

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

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

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

# 3.7. Physics-Dynamics Coupling

# 3.7.1. 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:

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

An illustration of the relationship between the time steps can be found in Figure 3.7.



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

ICON solves the fully compressible nonhydrostatic Navier-Stokes equations using a time stepping scheme that is explicit except for the terms describing vertical sound wave propagation (see Section 3.5). Thus, the maximum allowable time step  $\Delta \tau$  for solving the momentum, continuity and thermodynamic equations is determined by the fastest wave

in the system – the sound waves. As a rule of thumb, the maximum dynamics time step can be computed as

$$\Delta \tau = 1.8 \cdot 10^{-3} \ \overline{\Delta x} \frac{\mathrm{s}}{\mathrm{m}} \,, \tag{3.40}$$

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

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

unless ndyn_substeps is set to a non-default value.

Historical remark: Note that historically,  $\Delta \tau$  rather than  $\Delta t$  was used as basic control variable specified in the namelist, as appears logical from the fact that a universal rule for the length of the time step exists for  $\Delta \tau$  only. This was changed shortly before the operational introduction of ICON in 2015 because it turned out that an adaptive reduction of  $\Delta \tau$  is needed in rare cases with very large orographic gravity waves in order to avoid numerical instabilities. To avoid interferences with the output time control, the long time step  $\Delta t$  was then taken to be the basic control variable, which always remains unchanged during a model integration. The adaptive reduction of  $\Delta \tau$  is now accomplished by increasing the time step ratio ndyn_substeps automatically up to a value of 8 if the Courant number for vertical advection grows too large.

Additional time step restrictions for  $\Delta t$  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  $\Delta t$  should not significantly exceed 1000 s, which becomes relevant when  $\overline{\Delta x}$  is larger than about 125 km.

Even longer time steps than  $\Delta t$  can be used for the so-called slow-physics parameterizations, i.e. radiation, convection, non-orographic gravity wave drag, and orographic gravity wave drag. These parameterizations provide tendencies to the dynamical core, allowing them to be called individually at user-specified time steps. The related namelist switches are dt_rad, dt_conv, dt_gwd and dt_sso in nwp_phy_nml. If the slow-physics time step is not a multiple of the advective time step, it is automatically rounded up to the next advective time step. A further recommendation is that dt_rad should be an integer multiple of dt_conv, such that radiation and convection are called at the same time¹. The time-splitting is schematically depicted in Figure 3.7.

#### 3.7.2. Fast and Slow Processes

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

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

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

The calling sequence is saturation adjustment  $\longrightarrow$  surface transfer scheme  $\longrightarrow$  land-surface scheme  $\longrightarrow$  boundary-layer / turbulent vertical diffusion scheme  $\longrightarrow$  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. Both take the input at the 'old' time level because the surface variables are not updated in the dynamical core and the surface transfer coefficients and fluxes would be calculated from inconsistent time levels otherwise. The coupling strategy is schematically depicted in Figure 3.8.

Note that in the actual ICON code, the surface transfer scheme is called at the very end of the sequence of fast physics processes rather than at the beginning, as depicted in Figure 3.8. In compensation, the input to the surface transfer scheme is the 'new' time level rather than the 'old' one claimed in the text. Without proof, we claim that this is an equivalent implementation of the more natural sequence shown in Figure 3.8. The natural sequence would necessitate the allocation of additional memory for storing the 'old' atmospheric state, as this state is no longer available after the dynamics update due to the inherent substepping. By calling the surface transfer scheme at the end of the previous time step, the allocation of additional memory and related data transfer can be avoided.

Slow-physics processes are treated in a parallel-split manner, which means that they are stepped forward in time independently of each other, starting from the model state provided by the latest fast-physics process. In ICON convection, subgrid-scale cloud cover, radiation, non-orographic and orographic gravity wave drag are considered as slow processes. Typically, these processes are integrated with time steps longer than the (fast) physics time step. The slow-physics time steps can be specified by the user. The resulting slow-physics tendencies  $\partial v_n/\partial t$ ,  $\partial T/\partial t$  and  $\partial q_x/\partial t$  with  $x \in [v, c, i]$  are passed to the dynamical core and remain constant between two successive calls of the parameterization (Figure 3.8). Since ICON solves a prognostic equation for  $\pi$  rather than T, the temperature tendencies are converted into tendencies of the Exner function, beforehand. Rather than treating the moisture tendencies as a forcing term during tracer advection, they are treated in a time-split manner and added to the updated moisture variables thereafter.

# 3.7.3. 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  $\rho$  is one of the prognostic variables, whereas



Figure 3.8.: Coupling of the dynamical core and the NWP physics package. Processes declared as fast (slow) are treated in a time-split (process-split) manner.

pressure is only diagnosed for parameterizations needing pressure as input variable. Thus, it is natural to keep  $\rho$  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  $v_n$ .

# 3.8. ICON NWP-Physics in a Nutshell

An in-depth description of the physical parameterization package for NWP is beyond the scope of this document. However, the following section provides a short introduction to the available parameterizations and references for further reading.

Table 3.4 contains a summary of physical parameterizations available in ICON (NWP-mode). In what follows, an outline (executive summary) of the parameterization schemes is given.

#### 3.8.1. Radiation

S. Schäfer, DWD Physical Processes Division

Radiation is a crucial component that drives weather and climate from microscopical to global scales: Heating by absorption of radiation and cooling by emission determine local and global temperature and gradients, which in turn drive dynamics and physical processes. While some other physical effects occur locally, radiation travels throughout the depth of the atmosphere, therefore the entire atmospheric column has to be considered when calculating local radiative fluxes. Both visible or shortwave radiation from the sun and thermal or longwave radiation emitted within the Earth system interact with atmospheric gases, aerosols, clouds and the surface.

Radiative transfer models for the atmosphere consist of multiple components: optical property parameterizations for each atmospheric component and the surface, and a radiation solver that calculates how radiation travels through the optical medium. The new radiation scheme ecRad (Hogan and Bozzo, 2018; implementation in ICON: Rieger et al., 2019) allows choices for each component individually, while the radiation scheme itself is chosen with inwp_radiation=1 for ICON's RRTM radiation scheme

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.

Process		Scheme	Settings
Radiation		RRTM ( <u>R</u> apid <u>R</u> adiative <u>T</u> ransfer <u>M</u> odel)	inwp_radiation=1
		Mlawer et al. (1997), Barker et al. (2003)	
	Ø	ecRad	inwp_radiation=4
		Hogan and Bozzo (2018)	
Non-orographic	Ø	Wave dissipation at critical level	inwp_gwd=1
gravity wave drag		Orr et al. (2010)	
Sub-grid scale		Lott and Miller scheme	inwp_sso=1
orographic drag		Lott and Miller (1997)	
Cloud cover	Ø	Diagnostic PDF	inwp_cldcover=1
		M. Köhler et al. (DWD)	
		All-or-nothing scheme (grid-scale clouds)	inwp_cldcover=5
Microphysics		Single-moment scheme	<pre>inwp_gscp=1, 2</pre>
		Doms et al. (2011), Seifert (2008)	
		Double-moment scheme	inwp_gscp=4
		Seifert and Beheng (2006)	
Convection		Mass-flux shallow and deep	inwp_convection=1
		Tiedtke (1989), Bechtold et al. (2008)	
Turbulent transfer		Prognostic TKE (COSMO)	inwp_turb=1
		Raschendorfer (2001)	
		$EDMF\text{-}DualM\;(\underline{E}ddy\text{-}\underline{D}iffusivity/\underline{M}ass\text{-}\underline{F}lux)$	inwp_turb=3
		Köhler et al. (2011), Neggers et al. (2009)	
		3D Smagorinsky diffusion (for LES)	inwp_turb=5
Land		Tiled TERRA	inwp_surface=1
		Schrodin and Heise (2001), Schulz et al. (2016)	
		Flake: Mironov (2008)	llake=.TRUE.
		Sea-ice: Mironov et al. (2012)	lseaice=.TRUE.

Table 3.4.: Summary of ICON's physics parameterizations for NWP, together with the related namelist settings (namelist nwp_phy_nml). Parameterizations which are used operationally (at 13 km horizontal grid spacing) are indicated by **Z**.

and inwp_radiation=4 for ecRad (Namelist nwp_phy_nml). Using ecRad requires the Compiler-Flag -enable-ecrad in the config command before compiling, and setting the namelist parameter ecrad_data_path (Namelist radiation_nml). The necessary files are contained in '<ICON-directory>/externals/ecrad/data'. Since both the whole column and multiple spectral wavelengths have to be considered, the radiation calculation has to be simplified to be practical. Thus, radiation schemes for global weather and climate models neglect horizontal radiative transfer and treat only the vertical dimension. Clouds have a particularly strong radiative effect, and can vary on scales smaller than the model grid-boxes, therefore radiation is calculated once for the clear-sky part of each grid-box and once for the cloudy part.

In the spectral dimension, the strongest variability is due to atmospheric gases, which can absorb and emit radiation of particular, sharply defined wavelengths, according to their molecular properties. The optical properties of cloud particles, aerosol and the surface also depend on the wavelength, but vary more slowly. ICON's RRTM radiation scheme and ecRad both use the RRTM (Rapid Radiative Transfer Model Mlawer et al., 1997) gas optics scheme. The spectral range is divided into 30 spectral bands (16 bands in the longwave spectrum and 14 bands in the shortwave spectrum), and cloud, surface and aerosol optical properties are described within each band. The bands are subdivided into sub-intervals with similar gas properties, termed g-points. This so-called correlated-k method strongly reduces computational costs with an accuracy comparable to spectrally more detailed line-by-line models. Since we only consider up- and downwelling radiation, the optical properties are integrated for all angles within a hemisphere, reducing the optical parameters that are needed to optical depth, single scattering albedo and asymmetry factor (of scattering).

Cloud particle optical properties depend on the amount of water or ice, on the particle size and particle shape. Since cloud particles often have sizes similar to the visible or thermal wavelength ranges, scattering by particles varies strongly according to scattering angle and to the ratio of particle size to wavelength (Mie scattering). For given particle shape and size, this complex function can be approximated numerically. Cloud optical property parameterizations have to make an assumption on cloud particle shape. While liquid water particles are spherical, real ice particles can have a variety of shapes, so that ice shape assumptions are uncertain, and vary between parameterizations. Using these assumptions, the cloud optics parameterization provides optical properties depending on particle size (which is parameterized within ICON) and wavelength. In ecRad, several cloud water and ice optics parameterizations are available (namelist parameters iliquid_scat and iice_scat in radiation_nml).

For aerosol, optical properties are directly provided as an input to the radiation scheme. In operational settings, we use fixed global aerosol distributions from climatologies. Similarly, surface optical properties are provided to ICON in an external parameter file, based on satellite observations of the surface.

All of these optical properties are provided to the radiation solver, which calculates reflection, transmission and internal radiation sources in each grid-box and model layer, and the resulting amount of up- and downwelling radiation at each height, for cloudy and clear sky. Vertical overlap of cloudy and clear regions between neighboring layers is parameterized according to overlap assumptions (Hogan and Illingworth, 2000, chosen by namelist

parameter icld_overlap in radiation_nml). Operationally, ICON uses the exponentialrandom overlap assumption, meaning clouds with clear layers in-between are uncorrelated, while the overlap in continuous clouds decreases exponentially with vertical distance. Since cloud absorption and reflection depend non-linearly on cloud optical depth, the variability of thick and thin cloud in a grid-box also has an effect. Highly variable clouds interact less strongly with radiation than homogeneous clouds that contain the same amount of water. This effect can be parameterized roughly by reducing cloud optical depth to compensate (ICON's RRTM radiation uses a factor of 0.8). In ecRad, the variability is captured by dividing the cloudy region into two or more sub-regions with different cloud optical depths. One method to do this in a numerically efficient way is the Monte Carlo Independent Column Method (McICA, Pincus et al., 2003), which only calculates the radiative transfer for one spectral band in each sub-column. The distribution of the bands over the sub-columns introduces random noise, but no bias. Since this McICA method is comparatively cheap, it is the default method used, However, ecRad also provides a choice of other solvers without random noise: Tripleclouds, and SPARTACUS, which also approximately accounts for sub-grid horizontal transfer. The solver and further namelist parameters specific to ecRad are set in the module /src/atm_phy_nwp/mo_nwp_ecrad_init and are described in the ecRad documentation (https://confluence.ecmwf.int/display/ECRAD).

From the radiative fluxes, radiative heating and cooling is calculated, which feeds back into dynamics and physics. Despite the simplifying assumptions, radiation is still one of the most expensive parts of the model. Hence, radiation is calculated only on a coarser radiation grid (see Section 3.10) and at a coarse radiation time step  $dt_rad$  (see Section 3.7.1). However, radiative heating rates are updated more frequently, so that they better represent the diurnal cycle of incoming solar radiation.

#### 3.8.2. Saturation Adjustment

Section author A. Seifert, DWD Physical Processes Division

In ICON, the atmospheric state which enters the fast physics parameterizations has already been updated by the dynamical core (see Figure 3.8). As a consequence it is no longer guaranteed that vapor and liquid phase are in equilibrium. Hence supersaturated but cloud-free regions might exist, as well as sub- or supersaturated but cloudy ones.

The aim of a saturation adjustment scheme is to adjust the temperature and water vapor mixing ratio to perfect saturation in supersaturated regions. In subsaturated but cloudy regions, cloud water is evaporated until either saturation is reached or all cloud water is evaporated. From a cloud microphysical point of view the saturation adjustment scheme describes the processes of condensation and evaporation of cloud droplets.

In atmospheric models this adjustment process is usually treated isobarically. In ICON, however, it is treated isochorically, which is a consequence of ICON's physics-dynamics coupling strategy.

We start the description of the saturation adjustment scheme with a short derivation of the temperature equation emphasizing the difference between the enthalpy (constant pressure) and internal energy (constant volume) formulation and including water vapor and liquid

water. For a more complete derivation of the prognostic temperature equation see, e.g., Doms and Baldauf (2018).

To derive the prognostic temperature equation one uses the specific internal energy u or the specific enthalpy h = u + pv as a starting point. While the internal energy is appropriate for processes at constant volume, enthalpy would be chosen for processes at constant pressure. The prognostic temperature equation is derived as an expansion of either h or u as a function of temperature T, mass fractions of dry air  $q_d$ , water vapor  $q_v$  and liquid water  $q_\ell$  and either specific volume v or pressure p.

$$\frac{dh}{dt} = \left. \frac{\partial h}{\partial p} \right|_{T, q_k} \frac{dp}{dt} + \left. \frac{\partial h}{\partial T} \right|_{p, q_k} \frac{dT}{dt} + \sum_{k=d, v, \ell} \left. \frac{\partial h}{\partial q_k} \right|_{p, T} \frac{dq_k}{dt}$$
$$\frac{du}{dt} = \left. \frac{\partial u}{\partial v} \right|_{T, q_k} \frac{dv}{dt} + \left. \frac{\partial u}{\partial T} \right|_{v, q_k} \frac{dT}{dt} + \sum_{k=d, v, \ell} \left. \frac{\partial u}{\partial q_k} \right|_{v, T} \frac{dq_k}{dt}$$

With

$$c_p = \left. \frac{\partial h}{\partial T} \right|_{p, q_h}$$
$$c_v = \left. \frac{\partial u}{\partial T} \right|_{v, q_h}$$

and  $\sum q_k = 1$  and no phase transitions involving dry air, and therefore  $dq_v/dt = -dq_\ell/dt$ , we find:

$$\frac{dh}{dt} = \left. \frac{\partial h}{\partial p} \right|_{T, q_k} \frac{dp}{dt} + c_p \frac{dT}{dt} + (h_\ell - h_v) \frac{dq_\ell}{dt}$$
$$\frac{du}{dt} = \left. \frac{\partial u}{\partial v} \right|_{T, q_k} \frac{dv}{dt} + c_v \frac{dT}{dt} + (u_\ell - u_v) \frac{dq_\ell}{dt}$$

Now we use conservation of energy, i.e. dh = 0 or du = 0, and constant pressure or constant volume respectively, and find:

$$c_p \frac{dT}{dt} = -(h_\ell - h_v) \frac{dq_\ell}{dt} = L_{\ell v} I_\ell, \qquad p = \text{const.}$$
(3.41)

$$c_v \frac{dT}{dt} = -(u_\ell - u_v) \frac{dq_\ell}{dt} = \hat{L}_{\ell v} I_\ell, \qquad V = \text{const.}$$
(3.42)

The  $L_{\ell v}$  or  $\hat{L}_{\ell v}$  are the latent heats of vaporization and  $I_{\ell} = dq_{\ell}/dt$  is the condensation rate and also known as the 'Phasenfluss' (in German). The usual latent heat of vaporization given in most textbooks is the  $L_{\ell v}$  or the *enthalpy of vaporization*. At 0°C it has a value of  $L_{\ell v,0} = 2.501 \times 10^6$  J kg⁻¹. The corresponding *internal energy of vaporization*  $\hat{L}_{\ell v}$  can be derived from  $h_k = u_k + pv_k$  and with  $v_\ell \ll v_d$  and  $pv_v = R_v T$  we find

$$\hat{L}_{\ell v} = L_{\ell v} - pv_v = L_{\ell v} - R_v T$$

Saturation adjustment is a parameterization of condensation which assumes that vapor and liquid phase are in equilibrium, i.e., by saturation adjustment we want to ensure or realize this equilibrium. Therefore we simply assume that the final state with temperature  $T_1$  has a vapor mass fraction of  $q_{v,1} = q_{sat}(T_1)$  inside the cloud, i.e., whenever  $q_{\ell} > 0$ . From the temperature equation at constant volume (3.42) we find

$$c_{vd}(T_1 - T_0) + \hat{L}_{\ell v}(q_{sat}(T_1) - q_{v,0}) = 0$$

Note that we have now replaced  $c_v$  by the specific heat of dry air at constant volume  $c_{vd}$  and, hence, made the usual approximation to neglect the differences in the specific heats between dry air, vapor and liquid water in the temperature term. This is justified because  $q_v$  and  $q_l$  are usually small.

This equation is solved for  $T_1$ , e.g., using a Newton iteration

$$T_1^{n+1} = T_1^n - \frac{F(T_1^n)}{F'(T_1^n)},$$

with

$$F(T_1) = T_1 - T_0 + \hat{L}_{\ell v} \frac{q_{sat}(T_1) - q_{v,0}}{c_{vd}}$$
$$F'(T_1) = 1 + \frac{\hat{L}_{\ell v}}{c_{vd}} \frac{dq_{sat}}{dT} \Big|_{T=T_1}.$$

In ICON the iteration is stopped if either  $|T_1^{n+1} - T_1^n| < 1E - 3K$ , or if the number of iterations exceeds a hard-coded maximum maxiter = 10. Regarding the internal energy of vaporization  $\hat{L}_{\ell v}$  we actually use

$$\tilde{L}_{\ell v}(T) = L_{\ell v,0} + (c_{pv} - c_l)(T - T_0) - R_v T.$$

Hence, in addition to the  $R_v T$  term arising from the Legendre transformation to internal energy as discussed above, we take into account the linear temperature dependency of the latent heat of vaporization according to Kirchhoff's equation³

$$\frac{\partial L_{\ell v}}{\partial T}\Big|_{p, q_k} = \frac{\partial h_v}{\partial T}\Big|_{p, q_k} - \frac{\partial h_l}{\partial T}\Big|_{p, q_k} = c_{pv} - c_l.$$

This is consistent with the assumption of water vapor as an ideal gas with constant specific heat capacity.

#### 3.8.3. Cloud Microphysics

Section author

A. Seifert, DWD Physical Processes Division

Microphysical schemes provide a closed set of equations to calculate the formation and evolution of condensed water in the atmosphere. The most simple schemes predict only the specific mass content of certain hydrometeor categories like cloud water, rain water, cloud ice and snow. This is often adequate, because it is sufficient to describe the hydrological

³see e.g. http://glossary.ametsoc.org/wiki/Kirchhoff's_equation

cycle and the surface rain rate, which is the vertical flux of the mass content. Microphysical schemes of this category are called *single-moment schemes*.

In ICON two single-moment schemes are available, one that predicts the categories cloud water, rain water, cloud ice and snow (inwp_gscp=1 in the namelist nwp_phy_nml), and the other that predicts in addition also a graupel category (inwp_gscp=2). Graupel forms through the collision of ice or snow particles with supercooled liquid drops, a process called *riming*.

Most microphysical processes depend strongly on particle size and although the mean size is usually correlated with mass content this is not always the case. Schemes that predict also the number concentrations have the advantage that they provide a size information, which is independent of the mass content. Such schemes are called *double-moment schemes*, because both, mass content and number concentration, are statistical moments of the particles size distribution.

ICON does also provide a double-moment microphysics scheme (inwp_gscp=4), which predicts the specific mass and number concentrations of cloud water, rain water, cloud ice, snow, graupel and hail. This scheme is most suitable at convection-permitting or convection-resolving scales, i.e., mesh sizes of 3 km and finer. Only on such fine meshes the dynamics is able to resolve the convective updrafts in which graupel and hail form. On coarser grids the use of the double-moment scheme is not recommended.

To predict the evolution of the number concentrations the double-moment scheme includes various parameterizations of nucleation processes and all relevant microphysical interactions between these hydrometeor categories. Currently all choices regarding, e.g., cloud condensation and ice nuclei, particle geometries and fall speeds etc. have to be set in the code itself and can not be chosen via the ICON namelist.

# 3.8.4. Cumulus Convection

D. Klocke, DWD Physical Processes Division

Convection is an important process in the atmosphere by contributing to forming the largescale circulation to local heavy precipitation through thunderstorms. Parameterizations of atmospheric moist convection provide the effect of an ensemble of sub-grid convective clouds on the model column as a function of grid-scale variables. The schemes vertically mix heat, moisture and momentum. They convert available potential energy into kinetic energy and produce precipitation as a result of atmospheric instability.

Three steps are taken. First, it is determined if the grid-scale conditions allow for the occurrence of convection in the column, and a decision is taken if convection is triggered. In the second step, the tendencies of heat, moisture and momentum changes are determined with a *cloud model*, which represents an ascending parcel and its interactions with the environment. Finally the closure decides on the strength of the convection by determining the amount of energy to be converted, which is linked to precipitation amount generated by the convection scheme.

In ICON a bulk mass flux convection scheme is available <code>inwp_convection=1(in the namelist nwp_phy_nml)</code>, which treats three convective cloud types. Only one type - shallow, mid-level or deep convection - can exist at a time in a column, which is decided upon by the trigger function. All three types of convection use a cloud model representing an ascending plume mixing with its environmental air. The cloud base mass flux closure differs between the three convection types, with a CAPE (convective available potential energy) based closure for deep convection, a boundary layer equilibrium closure for shallow convection, and a large-scale omega (vertical velocity in pressure coordinates) based closure for mid-level convection.

The full convection scheme can generally be used for horizontal grids coarser than 5 km, as some resolution dependent adjustments are implemented for grid spacings smaller than 20 km. For convection permitting simulations (1-3 km horizontal grid spacing) the largest convective clouds can be resolved by the model and the parameterization parts treating deep and mid-level convection can be switched off (lshallowconv_only=.TRUE. in the namelist nwp_phy_nml).

The implemented scheme represents a branch of the Tiedtke-Bechtold convection scheme used in the IFS model. For further reading we refer to Bechtold et al. (2008), Tiedtke (1989), Bechtold (2017), ECMWF (2017). Similar to the operational IFS scheme it contains an improved CAPE closure for deep convection (Bechtold et al., 2014) in order to improve the representation of the diurnal cycle of convection over land (icapdcycle>0 in the namelist nwp_phy_nml). Note that in the operational ICON scheme the modified closure is only applied over land and latitudinally restricted to the tropics (icapdcycle=3).

#### 3.8.5. Cloud Cover

Section author M. Köhler, DWD Physical Processes Division

To prepare optical properties of clouds for the radiative transfer it is necessary to determine the best estimate of cloud cover, cloud water and cloud ice as well as the precipitation quantities, such as snow, rain and graupel if those are required by the radiation calculation. Note that there are various assumptions in the ICON model on the subgrid distribution of water, such as the up/down/subsidence regions in convection, a uniform distribution in microphysics, and a Gaussian distribution in turbulence.

The aim of the diagnostic cloud cover scheme is to combine information from the different parameterizations mentioned above: turbulence, convection and microphysics. The turbulence scheme provides the sub-grid variability of water due to turbulent motions, the convection scheme detrains cloud into the anvil and the microphysics scheme describes the supersaturation due to ice, in other words, the distribution between ice and vapor in cold situations.

The turbulent variability of water is at the moment prescribed by using a top-hat total water (the sum of water vapor  $q_v$ , cloud water  $q_c$ , and cloud ice  $q_i$ ) distribution with a fixed width of 10% of total water. Work is in progress to replace this crude assumption with the total water variance from the turbulence scheme.

The split between water vapor and cloud ice as determined from the microphysics scheme is replicated in the diagnostic cloud scheme for the turbulent component of ice clouds.

The convective anvil is calculated by writing an equation for the evolution of cloud cover that depends on the detrainment of volume (from the convection scheme) and a decay term with a fixed decay time-scale (taken as 30 min). The diagnostic assumption means that we can neglect the tendency term on cloud cover so that we arrive simply at the diagnostic anvil cloud cover that is purely a function of detrainment and decay time-scale. The liquid and ice cloud water is taken also from the convective updraft properties.

In the end the turbulent and convective clouds are combined with a simple maximum function.

To emphasize, the cloud cover scheme takes into account the subgrid variability of water and therefore the associated distribution in water vapor, cloud liquid water and cloud ice (optional 3D diagnostic output variables tot_qv_dia, tot_qc_dia, tot_qi_dia and their vertically integrated counterparts tqv_dia, tqc_dia, tqi_dia). They are not equal to their prognostic grid-scale equivalents (standard output variables qv, qc, qi), yet the sum of all three water quantities is kept the same. This cloud information is then passed to the radiation, where additional assumptions are made on the vertical overlap of clouds.

#### 3.8.6. Turbulent Diffusion

Section authors

M. Köhler and M. Raschendorfer, DWD Physical Processes Division

#### TKE Scheme for Turbulence.

The TKE turbulence scheme consists of two components: one describing the free troposphere, and the other for the surface layer.

**TURBDIFF.** The turbulence scheme TURBDIFF developed by Raschendorfer (2001) is based on a  $2^{nd}$ -order closure on level 2.5 according to Mellor and Yamada (1982) (MYscheme). In this scheme, all pressure correlation terms and dissipation terms, being present in the system of  $2^{nd}$ -order equations for all the turbulent (co-)variances that can be built from the dynamically active prognostic model variables, are expressed by the standard closure assumptions according to Rotta (1951a,b) and Kolmogorov (1968) valid for quasiisotropic turbulence. These dynamically active model variables are the horizontal wind components u and v, vertical wind speed w, a variable related to inner energy (like absolute temperature T, potential temperature  $\theta$  or virtual potential temperature  $\theta_v$ ), specific humidity  $q_v$  and at least one cloud water variable  $q_c$  (which is a mass fraction and may be split into liquid  $q_l$  and frozen  $q_i$  cloud water).

Among these  $2^{nd}$ -order moments, only the trace of the turbulent stress tensor, which is twice the Turbulent Kinetic Energy (TKE), is described by a prognostic equation. Each of the remaining  $2^{nd}$ -order equations (for the elements of the remaining trace-less stress tensor and for the other  $2^{nd}$ - order moments) is simplified as diagnostic source term equilibrium being a linear equation in terms of the governing statistical moments.

Further, correlations between any model variable and source terms of scalar model variables are neglected in these equations. However, by choosing quasi-conserved scalar variables (total water content  $q_t = q_v + q_c$  and liquid-water potential temperature  $\theta_l = \theta - \frac{L_c}{c_{p_d}}q_c$ ), these correlations are taken into account implicitly as far as local condensation and evaporation within liquid non-precipitating clouds is concerned. Hence, TURBDIFF is a moist turbulence scheme, which includes the effect of these sub-grid scale phase transitions. The required conversion of turbulent fluxes of these conserved variables into those of absolute temperature, specific humidity and liquid water content is performed by means of turbulent saturation adjustment, assuming a Gaussian distribution-function for local saturation-deficiency according to Sommeria and Deardorff (1977).

Finally, application of the horizontal boundary layer approximation reduces the linear system of diagnostic 2nd-order equilibrium equations to a single column scheme with only two equations for two diffusion coefficients (one for horizontal wind components and another for scalar variables), which both are proportional to the square root of TKE and an integral turbulent length scale. This length-scale rises with height above ground according to Blackadar (1962) with a further limitation related to the horizontal grid scale. The desired vertical turbulent fluxes of any prognostic variable can then be calculated by multiplying the (negative) vertical gradient of the latter with the associated diffusion coefficient.

One main extension of TURBDIFF (compared with a moist MY-scheme) is the formal separation of turbulence from a possible non-turbulent part of the subgrid-scale energy spectrum. This separation is related to additional scale-interaction terms in the prognostic TKE equation, which describe additional shear production of TKE by the action of other non-turbulent sub-grid-scale flow patterns (such as wakes generated by sub-grid-scale orography, convective currents or separated horizontal circulations). Through this formalism, the scheme describes Separated Turbulence Interacting with non-turbulent Circulations (STIC), which allows for a consistent application of turbulence closure assumptions, even though other sub-grid-scale processes may be dominant within a grid cell. Due to this extension, the scheme is applicable also above the boundary layer and for very stable stratification. The Eddy Dissipation Rate (EDR) calculated by TURBDIFF can even be used to forecast the intensity of Clear Air Turbulence (CAT).

**TURBTRAN.** The turbulence scheme TURBDIFF is closely related to the scheme TURBTRAN developed by Raschendorfer (2001) for the surface-to-atmosphere transfer (SAT), which calculates transport resistances for fluxes of prognostic model variables at the surface of the Earth. Figure 3.9 illustrates the corresponding sub layers of the surface layer. In TURBTRAN, a constant flux approximation is applied to the sum of turbulent and laminar vertical fluxes within the transfer layer (between the rigid surface and the lowest atmospheric main level of the model). By application of the turbulence scheme at the top of the lowest atmospheric layer as well as the bottom of this layer (which is the top of the near surface roughness layer being intersected by roughness elements), a vertical interpolation function for the turbulent diffusion coefficient is derived between these two levels and is extrapolated down to the rigid surface. With this preparation, a vertical integration of the flux gradient representation across the transfer layer provides the desired


Figure 3.9.: Surface layer as described in the parameterization TURBTRAN of Raschendorfer (2001).

transport resistances for the final bulk representation of SAT fluxes. With this formulation, scale interaction terms considered through STIC in the turbulence scheme also affect the transfer resistances; and hence, some additional mixing is automatically introduced for very large bulk Richardson numbers, as soon as non-turbulent sub-grid-scale motions are present.

Further, with the described procedure, the determination of a specific roughness length for scalars is substituted by a direct calculation of the partial resistance of scalar transfer through the laminar layer and the roughness layer. This partial resistance is dependent on the near-surface model variables, the aerodynamic roughness length and the Surface Area Index (SAI), which is a measure of the surface area enlargement by land use.

TURBDIFF and TURBTRAN are the default schemes for atmospheric turbulence and SAT, respectively, in ICON, and they correspond to inwp_turb =1 (namelist nwp_phy_nml). While TURBTRAN provides the transfer resistances for scalars and horizontal wind components, the final surface fluxes of water vapor and sensible heat are determined in TERRA as a result of updated surface values for  $q_v$  and (in case of the upcoming implicit treatment of surface temperature) also for T. Based on these surface fluxes, an implicit equation for vertical diffusion is solved as a final part of TURBDIFF. In this procedure also horizontal momentum and TKE is included, while for these 3 quantities the respective surface concentration is used as a lower boundary condition. Currently, a lower zero-concentration condition is applied for cloud-water and -ice, which is always related to a downward flux for these quantities. For vertical diffusion of passive tracers, the diffusion coefficient for scalars is applied, and the lower boundary condition can be specified individually. Although the effect of local condensation and evaporation is considered in the solution of 2nd –order equations, and hence, can amplify the intensity of turbulent vertical mixing, the direct effect of these additional thermodynamic source-terms in the grid-scale budgets of heat, water vapor and liquid water is not yet considered.

In the namelist turbdiff_nml several parameters or selectors for optional calculations related to both schemes can be specified. Through this, TURBDIFF can also be configured as a 3D-turbulence scheme, calculating additionally horizontal shear and providing also horizontal diffusion coefficients. This horizontal shear is the sum of related turbulent shear by the mean flow itype_sher>0 and (if ltkeshs=.TRUE.) of additional shear by larger sub-grid scale non-turbulent horizontal circulations (SHS), which are, for their part, generated by shear of the mean flow, but are formally separated from isotropic turbulence in the framework of STIC. Similarly, each horizontal diffusion coefficient is the sum of the isotropic turbulent diffusion coefficient and an optional additional one related to SHS, provided that this STIC term is active (ltkeshs=.TRUE.). However, in ICON, the calculation of horizontal diffusion is not yet connected with these non-isotropic diffusion coefficients from TURBDIFF. Rather, for the time being, isotropic diffusion coefficients calculated by a Smagorinsky formulation are automatically used for 3D diffusion.

#### **DualM EDMF for Turbulence and Shallow Convection**

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

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

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

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

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

#### Smagorinsky-Lilly for LES Application

The 3D sub-grid model of Smagorinsky (1963) with the stability correction of Lilly (1962) is implemented for LES applications. This scheme writes the eddy viscosity  $K_m$  as

$$K_m = (C_s \Delta)^2 |S| C_B,$$

with the Smagorinsky constant  $C_s$ , the filter width  $\Delta$ , the norm of the strain rate tensor |S| and the stability correction factor  $C_B$ .

The filter width is taken to be  $\Delta = (\Delta x \Delta y \Delta z)^{\frac{1}{3}}$ . The strain rate tensor is defined as

$$S_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \,,$$

whose calculation requires special care due to the triangular grid in ICON. A metric correction has been developed that treats the horizontal gradients over a sloped orography given a terrain-following coordinate correctly. The norm of  $S_{ij}$  is

$$|S| = \sqrt{2S_{ij}S_{ij}}$$

The stability correction factor  $C_B$  is given by

$$C_B = (1 - \operatorname{Ri} / \operatorname{Pr}_t)^{\frac{1}{2}}$$

with the gradient Richardson number  $\operatorname{Ri} = \frac{N^2}{|\overline{S}|}$  and the buoyancy frequency  $N^2 = \frac{g}{\theta_0} \frac{\partial \theta_v}{\partial z}$ .

The Smagorinsky constant usually has the value of  $C_s = 0.1 - 0.2$ . In the ICON implementation  $C_s$  is set by default to smag_constant =0.23 (namelist les_nml) and the Prandtl number  $\Pr_t$  to turb_prandtl =0.333 (namelist les_nml).

## 3.8.7. Sub-grid scale orographic drag

Section authors

J.-P. Schulz, DWD Numerical Models DivisionM. Köhler, DWD Physical Processes DivisionS. Borchert, DWD Numerical Models Division

ICON treats the entire sub-grid scale orographic drag with one model based on the work of Lott and Miller (1997). Other NWP models treat scales smaller than  $\sim 5 \,\mathrm{km}$  with different ansatzes, for example with the help of turbulent orographic form drag formulations of the use of an effective roughness length that determines the conductivity of the surface for momentum fluxes between the atmosphere and the Earth.

The motivation for the implementation of a sub-grid scale orographic drag model into ICON traces back to experience with the COSMO model. When the 7-km EU domain of the operational COSMO model at DWD was expanded in order to cover almost all Europe (see Schulz, 2006), it turned out that the surface pressure in the model forecasts became systematically biased. In particular, in wintertime high pressure systems of the model atmosphere tended to develop a positive pressure bias, by 1 to 2 hPa after 48 h, low

pressure systems a negative bias ("highs too high, lows too low"). At the same time the wind speed tended to be overestimated by up to  $1 \text{ m s}^{-1}$  throughout the entire troposphere. The wind direction near the surface showed a positive bias.

The combination of these deficiencies led to the hypothesis that in the model there is too little surface drag, causing an underestimation of the cross-isobar flow in the planetary boundary layer. Consequently, the solution would be to increase the surface drag in the model. This may be accomplished, for instance, by introducing an envelope orography (Wallace et al., 1983, Tibaldi, 1986), but this has unfavorable effects, e.g., for the simulated precipitation. Another option is the inclusion of sub-grid scale orographic (SSO) effects, which were neglected in the COSMO model before. The SSO scheme by Lott and Miller (1997) was selected for this purpose. Its implementation in the COSMO-EU model is described in Schulz (2008), and was later transferred to the ICON model.

The SSO scheme by Lott and Miller (1997) deals explicitly with a low-level flow which is blocked when the sub-grid scale orography is sufficiently high. For this blocked flow separation occurs at the mountain flanks, resulting in a form drag. The upper part of the low-level flow is lead over the orography, while generating gravity waves⁴. In order to describe the low-level flow behavior in the SSO scheme a non-dimensional height  $H_n$  of the sub-grid scale mountain is introduced

$$H_n = \frac{NH}{|U|} = Fr^{-1} \,,$$

where H is the maximum height of the mountain, |U| is the wind speed and N is the Brunt-Väisälä frequency of the incident flow. The latter is defined by

$$N = \sqrt{\frac{g}{\theta} \frac{\partial \theta}{\partial z}} \,,$$

where  $\theta$  is the potential temperature, g the acceleration of gravity and z the height coordinate.  $H_n$  may be also regarded as an inverse Froude number  $Fr^{-1}$  of the system "flow round a mountain".

A small  $H_n$  means that there is an unblocked regime, all the flow goes over the mountain and gravity waves (GWs) are forced by the vertical motion of the fluid. A large  $H_n$  means that there is a blocked regime, the vertical motion of the fluid is limited and part of the low-level flow goes around the mountain. The SSO scheme requires four external parameters, which are the standard deviation SSO_STDH, the anisotropy SSO_GAMMA, the slope SSO_SIGMA and the geographical orientation SSO_THETA of the sub-grid scale orography. Following Baines and Palmer (1990) these are computed by ExtPar (see Section 2.4.1) from the GLOBE data set (GLOBE-Task-Team, 1999), which has a resolution of approximately 1 km.

Four tuning parameters in the namelist nwp_tuning_nml control the SSO scheme (see Figure 3.10). First, the critical Froude number  $Fr_c = \text{tune_gfrcrit}$ , which has a twofold effect. The larger its value the higher the likelihood for low-level blocking to occur, since it is activated only where  $Fr < Fr_c$ . Conversely, where  $Fr > Fr_c$  all flow goes over the mountain and the entire stress is associated with GW radiation. In addition, if blocking is

⁴Propagating, coherent structures consisting of buoyancy and inertial oscillations.



Figure 3.10.: Left: Schematic illustration of the two elements of sub-grid scale orographic drag: First, the low-level blocking, where the horizontal flow (orange) is forced to flow around the sub-grid scale orography (SSO, colored gray). Second, the gravity wave drag on the flow forced to overflow the SSO, and on the flow at higher altitudes, where the radiated gravity waves (white-blue) break. (Following Figure 1 in Lott and Miller (1997).) Right: Schematic illustration of the non-orographic gravity wave drag.

active,  $Fr_c$  controls the thickness of the blocking layer relative to the mountain height (the larger  $Fr_c$  the larger the layer thickness). Its default value is 0.4. Operational simulations with horizontal mesh sizes of 13 km and 40 km use the values 0.333 and 0.425, respectively. Second and third, the magnitudes of the SSO drag and the GW drag are directly proportional to the parameters tune_gkwake and tune_gkdrag, respectively. The default values are 1.5 and 0.075. Different from this, operational simulations with 13 km horizontal mesh size use tune_gkwake = 1.8 and tune_gkdrag = 0.09. Finally, the critical Richardson number  $Ri_c = tune_grcrit$  is a control parameter for the onset of GW breaking. If the Richardson number of the resolved atmospheric state plus the unresolved GWs

$$Ri = \frac{N_{\rm tot}^2}{|\partial \boldsymbol{u}_{\rm tot}/\partial z|^2}\,,$$

where u denotes the horizontal wind vector, falls below  $Ri_c$ , the flow configuration becomes unstable and the GWs break (partly). This process is accompanied by a drag effect on the resolved horizontal flow. Combined with the rule of thumb that Ri decreases with height if GWs are present, this means that the larger the value of  $Ri_c$  the lower the altitude where the radiated GWs tend to break and exert a drag. The default value is 0.25.

The scheme computes tendencies of the horizontal wind and the temperature. A detailed description can be found in ECMWF (2018a).

## 3.8.8. Non-orographic gravity wave drag

Section authors

M. Köhler, DWD Physical Processes Division S. Borchert, DWD Numerical Models Division

All kinds of (synoptic-scale) atmospheric flow structures (e.g., fronts, convection, jet streams) can develop imbalances that force air parcels to oscillate (vertically) and radiate gravity waves (GWs). The interaction of these non-orographically forced GWs with the atmospheric background flow is assumed to be significant in the middle and upper atmosphere, and is consequently of interest for models that cover this region (e.g., the operational global ICON configuration with its model top at  $75 \,\mathrm{km}$ ). If the synoptic scale flow (be it resolved or unresolved by the model) would force GWs, whose horizontal and vertical wave lengths would be smaller than the horizontal and vertical grid mesh sizes, they cannot be resolved by the model. But yet this unresolved part of the GW spectrum could have a significant impact on the resolved flow, and is therefore parameterized (see Figure 3.10).

The parameterization implemented in ICON follows the ansatz of Scinocca (2003) and McLandress and Scinocca (2005), which in turn is based on the work of Warner and McIntyre (1996) to which the simplifying assumption of a hydrostatic, non-rotational atmosphere has been applied. The parameters of this scheme have been optimized following Ern et al. (2006). Details can be found in Orr et al. (2010).

The mechanisms of non-orographic GW forcing can be relatively complex and are not completely understood. For this reason the parameterization assumes a constant source of GWs. The amount of GWs radiated by this idealized source is directly proportional to the namelist parameter tune_gfluxlaun (nwp_tuning_nml). Its default value is 0.0025 Pa.

The scheme computes tendencies of the horizontal wind and the temperature. A detailed description can be found in ECMWF (2018b).

## 3.8.9. Lake Parameterization Scheme FLake

Section author

D. Mironov, DWD Physical Processes Division

Lakes significantly affect the structure and the transport properties of the atmospheric boundary layer. The interaction of the atmosphere with the underlying surface strongly depends on the surface temperature.

In numerical weather prediction (NWP), a simplified approach is often taken that amounts to keeping the water surface temperature constant over the entire forecast period. This approach is to some extent justified for ocean and seas. It is hardly applicable to lakes where diurnal variations of the water surface temperature reach several degrees. The situation is even more grave for frozen lakes as the diurnal variations of the ice surface temperature may exceed ten degrees.

Initialization of the NWP model grid boxes that contain water bodies also presents considerable difficulties. When no observational data for some grid boxes are available, those grid boxes are initialized by means of interpolation between the nearest grid-boxes for which the water surface temperature is known (the interpolation procedure may account for some other variables, e.g., two-meter temperature over land). Such procedure is acceptable for sea points where large horizontal gradients of the water surface temperature are comparatively rare, but it is hardly suitable for lakes. Lakes are enclosed water bodies of a comparatively small horizontal extent. The lake surface temperature has little to do with the surface temperature obtained by means of interpolation between the alien water bodies.

In NWP, the lake surface temperature (i.e., the surface temperature of lake water or lake ice) is a major concern. It is this variable that communicates information between the lake and the atmosphere, whereas details of the vertical temperature distribution (e.g., the temperature near the lake bottom) are of minor importance. Therefore, simplified lake models (parameterization schemes), whose major task is to predict the lake surface temperature, the lake freezing and the ice break-up, should be sufficient for NWP and related applications.

The NWP model ICON (as well as COSMO) utilizes the lake parameterization scheme FLake (Mironov, 2008, Mironov et al., 2010, 2012). FLake is based on a two-layer parametric representation of the evolving temperature profile. The structure of the stratified layer between the upper mixed layer and the basin bottom, the lake thermocline, is described using the concept of self-similarity (assumed shape) of the temperature-depth curve. The same concept is used to describe the temperature structure of the thermally active upper layer of bottom sediments and of the ice and snow cover. In this way, the problem of solving partial differential equations (in depth and time) for the temperature and turbulence quantities is reduced to solving ordinary differential equations (in time only) for the time-dependent parameters that specify the temperature profile.

The approach is based on what is actually "verifiable empiricism". However, it still incorporates much of the essential physics and offers a very good compromise between physical realism and computational economy. FLake incorporates the heat budget equations for the four layers in question, viz., snow, ice, water and bottom sediments, developed with due regard for the volumetric character of the solar radiation heating. An entrainment equation is used to compute the depth of a convectively-mixed layer, and a relaxation-type equation is used to compute the wind-mixed layer depth in stable and neutral stratification. Simple thermodynamic arguments are invoked to develop the evolution equations for the ice and snow depths.

Empirical constants and parameters of FLake are estimated, using independent empirical and numerical data. They should not be re-evaluated when the scheme is applied to a particular lake. In this way, the scheme does not require re-tuning, a procedure that may improve an agreement with a limited amount of data but should generally be avoided. Further information about FLake can be found at http://lakemodel.net.

FLake is activated within ICON if the namelist parameter llake (lnd_nml) is set .TRUE., which is the default operational setting at DWD.

FLake requires two external parameter fields. These are the fields of lake fraction (area fraction of a given numerical-model grid box covered by the lake water) and of lake depth. These external parameter fields are generated with the ExtPar software (see Section 2.4)

using the Global Lake Database (Kourzeneva, 2010, Kourzeneva et al., 2012, Choulga et al., 2014).

ICON makes use of a tile approach to compute the grid-box mean values of temperature and humidity (and of other scalars) and the grid-box mean fluxes of momentum and scalars. FLake is applied to the ICON grid boxes whose lake fraction exceeds a threshold value; otherwise the effect of sub-grid scale lakes is ignored. Currently, the value of 0.05 is used (see the namelist variable frlake_thrhld (lnd_nml)).

In the current ICON configuration, the lake depth is limited to 50 m. For deep lakes, the abyssal layer is ignored, a "false bottom" is set at a depth of 50 m, and the bottom heat flux is set to zero. The bottom sediment module is switched off, and the heat flux at the water-bottom sediment interface (or at false bottom) is set to zero. The setting lflk_botsed_use=.FALSE. is hard-coded in mo_data_flake.f90.

Snow above the lake ice is not considered explicitly. The effect of snow is accounted for in an implicit manner through the temperature dependence of the ice surface albedo with respect to solar radiation Mironov et al. (2012). There is no logical switch to deactivate the snow module of FLake. It is sufficient to set the rate of snow accumulation to zero (hard-coded in mo_flake.f90). Without explicit snow layer of the lake ice, the snow depth over lakes is set to zero and the snow surface temperature is set equal to the ice surface temperature.

The attenuation coefficient of lake water with respect to solar radiation is currently set to a default "reference" value for all lakes handled by ICON. It would be advantageous to specify the attenuation coefficient as a global external parameter field. This can be done in the future as the information about the optical properties of lakes becomes available (not the case at the time being).

Generally, no observational data are assimilated into FLake, i.e., the evolution of the lake temperature, the lake freeze-up, and break-up of ice occur freely during the ICON runs. An exception are the Laurentian Great Lakes of North America. Over the Laurentian Great Lakes, the observation data on the ice fraction (provided by the ICON surface analysis scheme) are used to adjust the ice thickness, the ice surface temperature, and (as needed) the water temperature. See the subroutine flake_init in mo_flake.f90 for details. The use of the ice-fraction data over Great Lakes is controlled by the namelist parameter use_lakeiceana (initicon_nml).

Finally, a word of caution is in order. Running ICON with the lake parameterization scheme switched off (llake=.FALSE.) is not recommended as this configuration has never been comprehensively tested at DWD.

## 3.8.10. Sea-Ice Parameterization Scheme

Section author

D. Mironov, DWD Physical Processes Division

A major task of the sea-ice parameterization scheme for NWP is to predict the existence of ice within a given atmospheric-model grid box and the ice surface temperature. The sea-ice scheme used within ICON NWP accounts for thermodynamic processes only, i.e., no ice rheology is considered (cf. the sea-ice scheme for climate modeling). The horizontal distribution of the ice cover, i.e., the fractional area coverage of sea ice within a given grid box, is governed by the data assimilation scheme. A detailed description of the sea-ice scheme for ICON NWP is given in Mironov et al. (2012), where a systematic derivation of governing equations, an extensive discussion of various parameterization assumptions and of the scheme disposable parameters, and references to relevant publications can be found. Further comments can be found directly in the code, see the module src/lnd_phy_schemes/mo_seaice_nwp.f90.

A distinguishing feature of the ICON NWP sea-ice scheme is the treatment of the heat transfer through the ice. As different from many other sea-ice schemes that solve the heat transfer equation on a finite difference grid, the present scheme uses the integral, or bulk, approach (cf. the lake parameterization scheme FLake, Section 3.8.9). It is based on a parametric representation (assumed shape) of the evolving temperature profile within the ice and on the integral heat budget of the ice slab. Using the integral approach, the problem of solving partial differential equations (in depth and time) is reduced to solving ordinary differential equations (in time only) for the quantities that specify the evolving temperature profile. These quantities are the ice surface temperature and the ice thickness.

In the full-fledged scheme outlined in Mironov et al. (2012), provision is made to account for the snow layer above the ice. Both snow and ice are modeled using the same basic concept, that is a parametric representation of the evolving temperature profile and the integral energy budgets of the ice and snow layers (see Mironov (2008) for a detailed discussion of the concept). In the current ICON configuration, snow over sea ice is not considered explicitly. The effect of snow is accounted for implicitly (parametrically) through the ice surface albedo with respect to solar radiation.

A prognostic sea-ice albedo parameterization is used. The sea-ice surface albedo is computed from a relaxation-type rate equation, where the equilibrium albedo and the relaxation (e-folding) time scale are computed as functions of the ice surface temperature. In order to account for the increase of the sea-ice albedo after snowfall events, the ice albedo is relaxed to the equilibrium "snow-over-ice" albedo. The equilibrium snow-over-ice albedo is computed as function of the ice surface temperature, and the relaxation time scale is related to the snow precipitation rate.

The horizontal distribution of the ice cover, i.e., the existence of sea ice within a given ICON grid box and the ice fraction, is governed by the data assimilation scheme (cf. the treatment of lake ice). If an ICON grid box has been set ice-free during the initialization, no ice is created over the forecast period. If observational data indicate open water conditions for a given grid box but there was ice in that grid box at the end of the previous ICON run, ice is removed and the grid box is initialized as ice-free. The new ice is formed instantaneously if the data assimilation scheme indicates that there is sea ice in a given grid box, but there was no ice in that grid box in the previous model run. The newly formed ice has the surface temperature equal to the salt-water freezing point. The thickness of newly formed ice is computed as function of the ice fraction.

ICON utilizes a tile approach to compute surface fluxes of momentum and scalars. For the "sea-water type" grid boxes, the grid-box mean fluxes are computed as a weighted mean of fluxes over ice and over open water, using fractional ice cover  $f_i$  and fractional open-water cover  $1 - f_i$  as the respective weights. Sea ice in a given ICON grid box is only considered

#### ICON Model Tutorial

if  $f_i$  exceeds its minimum value of 0.015, otherwise the grid box is treated as ice free (see parameter frsi_min hard-coded in mo_seaice_nwp.f90). Likewise, the open-water fraction less than frsi_min are ignored, and the grid box in question is treated as fully ice-covered ( $f_i$  is reset to 1). The ice fraction is determined during the model initialization and is kept constant over the entire forecast period. If, however, sea ice melts away during the forecast,  $f_i$  is set to zero and the grid box is treated as an open-water water grid box for the rest of the forecast period (prognostic ice thickness is limited from below by a value of 0.05 m, i.e., a thinner ice is removed). The water-surface temperature of that grid box is equal to the observed value from the analysis, or is reset to the salt-water freezing point. The latter situation is encountered when a grid box was entirely covered by ice at the beginning of the forecast, but the ice melts away during the forecast.

In order to run ICON with the sea-ice parameterization scheme switched off, the namelist logical switch lseaice (lnd_nml) should set equal to .FALSE.. This configuration has not been comprehensively tested at DWD and is not recommended.

# 3.8.11. Land-Soil Model TERRA

Section author

J. Helmert, DWD Physical Processes Division

The soil-vegetation-atmosphere-transfer component TERRA (Schrodin and Heise, 2001, Heise et al., 2006, Schulz et al., 2016) in the ICON model is responsible for the exchange of fluxes of heat, moisture, and momentum between land surface and atmosphere. It establishes the lower boundary-condition for the atmospheric circulation model and considers the energy and water budget at the land surface fractions of grid points. Based on a multi-layer concept for the soil, TERRA considers the following physical processes at each of the tiled land-surface columns, where an uniform soil type with physical properties is assumed:

## Radiation

- Photosynthetically active radiation (PAR) is used for plant evapotranspiration
- Solar and thermal radiation budget is considered in the surface energy budget

## Biophysical control of evapotranspiration

- Stomatal resistance concept controls the interchange of water between the atmosphere and the plant
- One-layer vegetation intercepts and hold precipitation and dew, which lowers water input to the soil and enhances evaporation
- Roots with root-density profile determines the amount of water available for evapotranspiration in the soil
- Bare-soil evaporation is considered for land-surface fractions without plants.

## Heat and soil-water transport

- Implicit numerical methods are used to solve the vertical soil water transport and soil heat transfer between the non-equidistant layers.
- In the operational model version seven layers are used in the soil.

- The lower boundary condition for the heat conduction equation is provided by the climatological mean temperature.
- Surface and sub-surface runoff of water is considered.
- The lower boundary condition is given by a free-drainage formulation.
- A rise of groundwater into the simulated soil column is not represented.
- Soil heat conductivity depends on soil-water content.
- Freezing of soil water and melting of soil ice is considered in hydraulic active soil layers.

## Snow

- TERRA offers a one-layer snow model (operational in ICON-NWP) and a multi-layer snow model option (for experiments).
- A prognostic snow density, and snow melting process as well as the time dependent snow albedo are considered
- Surface fractions partly covered with snow are divided in snow-free and snow-covered parts (snow tiles)

#### Coupling to the atmosphere

- Application of the turbulence scheme at the lower model boundary
- Roughness length for scalars implicitly considered by calculation of an additional transport resistance throughout the turbulent and laminar roughness layer.

TERRA requires a number of external parameter fields, see Section 2.4 for details.

## The Tile Approach

The tile approach addresses the problem of calculating proper cell-averaged surface fluxes in the case of large subgrid variations in surface characteristics. The basic idea following Avissar and Pielke (1989) is depicted in Figure 3.11. If patches of the same surface type occur within a grid box, they are regrouped into homogeneous classes (tiles). The surface energy balance and soil physics are then computed separately for each tile, using parameters which are characteristic of each surface type (roughness length, leaf area index, albedo,  $\dots$ ). The atmospheric fields which enter the computations, however, are assumed uniform over the grid cell, i.e. the so-called blending height is located at the lowermost atmospheric model level. The contributions from different tiles are then areally weighted to provide the cell-averaged atmospheric forcing. Note that in this approach the geographical distribution of subgrid heterogeneities is not taken into account.

In ICON, the number of surface tiles is specified by the parameter ntiles (lnd_nml). Setting ntiles=1 means that the tile approach is switched off, i.e. only the dominant land-surface type in a grid cell is taken into account. Setting ntiles to a value n > 1, up to n dominant land tiles are considered per grid cell. Note, however, that for n > 1 the total number of tiles  $n_{tot}$  is implicitly changed to  $n_{tot} = n+3$ , with three additional "water" tiles classified as "open water" (n + 1), "lake" (n + 2), and "sea-ice" (n + 3). Additional snow-tiles can be switched on by setting lsnowtile=.TRUE.. In that case the total number



Figure 3.11.: Tile approach for a grid cell containing various surface types. Patches of the same surface type within a grid box are regrouped into homogeneous classes (tiles) for which the soil and surface parameterizations are run separately.

of tiles is further expanded to  $n_{tot} = 2 \cdot n + 3$ , with the first n tiles denoting the land tiles, the second n tiles denoting the corresponding snow tiles and 3 water tiles as before.

## The process of tile generation in ICON works as follows:

During the setup phase, all land-surface types within a grid box are ranked according to the fractional area f they cover (see Figure 3.12, outer ring). For efficiency reasons, only the ntiles (typically about 3) dominating ones are represented by tiles, with the others being discarded (inner ring). If a grid cell contains non-negligible water bodies (f > 5%), up to 3 more tiles are created (i.e. open water, lake, and sea-ice) even if they are not among the dominating ones. By this approach, the surface types represented by tiles can differ from grid cell to grid cell such that the full spectrum of surface types provided by the land cover data set is retained.

If the model is initialized from horizontally interpolated initial data and ntiles > 1, a tile coldstart becomes necessary. This can be done by setting ltile_init=.TRUE. and ltile_coldstart=.TRUE. in the namelist initicon_nml. Each tile is then initialized with the same cell averaged value. Note that ltile_init=.TRUE. is only necessary, if the initial data come from a model run without tiles.



#### Important note:

Naive horizontal interpolation of tile-based variables is incorrect, since the dominant tiles and/or their internal ranking will most likely differ between source and target cell. Only aggregated fields can be interpolated!



Figure 3.12.: Tile generation for ntiles=3, for the case of a heterogeneous land surface. Outer circles show the fractional areas covered by the respective surfacetype for a given grid cell. Inner circles show the selected tiles. Please note the re-scaling of fractional areas in the inner circle.

# 3.8.12. Reduced Model Top for Moist Physics

A notable means for improving the efficiency of ICON is depicted in Figure 3.13. The switch

htop_moist_proc (namelist nonhydrostatic_nml, floating-point value)

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

One variant of the implemented horizontal transport scheme for passive scalars is capable of performing internal substepping. This means that the transport time step  $\Delta t$  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  $80 \text{ m s}^{-1}$ . In real case applications, this mostly happens in the stratosphere and mesosphere. The recommendation for  $\Delta t$  given in Section 3.7.1 then exceeds the numerical stability range of the horizontal transport scheme. To stabilize the integration without the need to reduce the time step globally, transport schemes with and without internal substepping can be combined. The switch

hbot_qvsubstep (namelist nonhydrostatic_nml, floating-point value)

indicates the height above which the transport scheme switches from its default version to a version with internal substepping. The default value is hbot_qvsubstep=22500m.



Figure 3.13.: 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)

Note that substepping is only performed for a particular tracer if a suitable horizontal transport scheme is chosen. The horizontal transport scheme can be selected individually for each tracer via the namelist switch ihadv_tracer (transport_nml). Variants of the transport scheme with internal substepping are indicated by a two-digit number (i.e. 22, 32, 42, 52). These variants mostly differ w.r.t. the accuracy of the polynomial reconstruction used for the flux estimation. A linear reconstruction is used by the variant 22, whereas 52 uses a cubic reconstruction. See Section 3.6.5 for additional details regarding the transport algorithm.

If moist physics are switched off above 22.5 km (default for NWP applications), internal substepping only needs to be applied for specific humidity  $q_v$ , since the advection of all other moisture fields is switched off anyway. However, be aware that you must explicitly enable internal substepping if moisture physics are not switched off, or if other (non-microphysical) tracers are added to the simulation.

# 3.9. Variable Resolution Modeling

ICON has the capability for static mesh refinement in horizontal directions (Zängl et al., 2022). This is realized through a *multi-grid approach* which means that one or more additional higher resolution (child) domains can be overlaid on a coarser base (parent) domain. This base domain can be a regional or a global domain. Each child domain has a defined parent domain providing lateral boundary conditions, but a parent domain can have several child domains. The child domains can be located in different geographical regions and can also be parent domains for further subdomains. Technically, the number of nested domains is arbitrary, but of course not all choices would make sense from a physical point of view.



Figure 3.14.: Basic example of a multi-grid setup, consisting of a global ICON domain and a child domain over Europe with half grid spacing. This is similar to the deterministic forecast setup that is operationally used at DWD with a horizontal grid spacing of 13 km globally and 6.5 km in the child domain.



The multi-grid approach in ICON closely resembles the classical two-way nesting approach known from many mesoscale models, e.g. MM5 (Grell et al., 1994) or WRF (Skamarock et al., 2019) but differs in the fact that the feedback is based on a Newtonian relaxation approach rather than directly replacing the prognostic fields in the parent domain by upscaled values from the child domain. It also has to be distinguished from recent *uni-grid* approaches, where more cells are added to an existing grid in special areas of interest (*h-refinement*), and where the solver computes a single solution for the whole grid. Atmospheric models capable of static h-refinement are e.g. CAM-SE (Zarzycki et al., 2014) and MPAS-A (Skamarock et al., 2012).

The multi-grid approach easily allows for switching domains on or off at runtime, as well as intertwining one-way and two-way nested domains. Two-way as opposed to one-way nesting means that the solution on the child domain is transferred back to the coarser parent domain every time step by means of a feedback mechanism which is described below.

The basic multi-grid example shown in Figure 3.14 consists of one global domain and one regional domain over Europe. The refinement ratio between the parent domain and the child domain is fixed to a value of 2, i.e. each parent triangle is split into 4 child triangles as shown in Figure 3.15. Consistent with the refinement ratio of 2, the time steps for dynamics and physics are automatically halved for each nesting level. Hence,  $\Delta t$  must be specified for the base (i.e. coarsest) domain only.



Figure 3.15.: Nested grids are constructed by splitting each parent cell into 4 child cells.

The coupling time step  $\Delta t$  between successive nesting levels is the large (fast physics) time step described in Zängl et al. (2015), which is usually five times the so-called dynamics time step limited by the Courant-Friedrichs-Lewy (CFL) stability criterion for horizontal sound-wave propagation, see Section 3.7.1.

The grids corresponding to the different refinement levels are stored in separate files. The usual way to establish a parent-child relationship between these grids is to read the header information from the list of provided grid files:

dynamics_grid_filename (namelist grid_nml, list of string parameters)

This parameter specifies the name(s) of the horizontal grid file(s) and thus implicitly activates the nesting function. For a global simulation with multiple nests a filename must be specified for each domain, see Section 4.1.2 for a practical example. Then, the parent-child relationships can be inferred from the NetCDF attributes uuidOfHGrid and uuidOfParHGrid, that have been described in Section 2.1.2.

## 3.9.1. Parent-Child Coupling

This section describes the exchange of information between a single parent and child domain. As shown in Figure 3.16, a nested domain can conceptually be split into three areas: A boundary interpolation zone (red), a nudging zone (light gray) and a feedback zone (blue). Prognostic computations are restricted to the latter two. The nudging zone only exists for one-way nesting or in limited area mode (LAM), whereas for two-way nesting the feedback zone directly borders on the boundary zone.

In the following let n and n + 1 denote the begin and end of the current large time step. Once the model state on the parent domain  $\mathcal{M}_p$  has been advanced from n to n + 1, the states  $\mathcal{M}_p^n$  and  $\mathcal{M}_p^{n+1}$  are used to update the boundary interpolation zone on the child domain. Hence, the boundary interpolation zone provides the necessary lateral boundary (forcing) data in order to advance the model state on the child domain from  $\mathcal{M}_c^n$  to  $\mathcal{M}_c^{n+1}$ . It has a fixed width of 4 cell rows (see Figure 3.16), which is motivated by the technical constraint that the boundary zone needs to match with parent cell rows (i.e. an odd number of cell rows is not allowed), combined with the fact that 2 cell rows would not be sufficient to cover all stencil operations performed in the dynamical core. For example, the  $\nabla^4$ -diffusion operator (Zängl et al., 2015) requires information from three adjacent cell rows.

In the feedback zone, the updated model state on the child domain  $\mathcal{M}_c^{n+1}$  is transferred (interpolated) back to the parent domain. By this, the parent and child domain remain closely



Figure 3.16.: General structure of a nested domain. Red: boundary interpolation zone consisting of 4 cell rows (see enlarged view). Light-gray: nudging zone with adjustable width, which is only active for one-way nesting and in limited area mode. Blue: feedback zone. Prognostic computations are performed in the feedback and nudging zone.

coupled, and the simulation on the parent domain benefits from the higher-resolution results of the child domain.

In the nudging zone, the model state on the child domain  $\mathcal{M}_c^{n+1}$  is nudged towards the corresponding parent domain state  $\mathcal{M}_p^{n+1}$ , in order to accommodate possible inconsistencies between the two domains. The nudging is essentially a relaxation of the prognostic variables towards the lateral boundary data following Davies (1976). The same method is applied in limited area mode (see Section 6.2).

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

## Lateral Boundary Update: Parent $\rightarrow$ Child

The boundary update mechanism provides the child domain with up-to-date lateral boundary conditions for the prognostic variables  $v_n$ , w,  $\rho$ ,  $\theta_v$ ,  $q_k$ . In order to avoid that parent-tochild interpolated values of  $\rho$  enter the solution of the mass continuity equation, the above set of variables is extended by the horizontal mass flux  $\rho v_n$ . This will allow for parent-child mass flux consistency, as described below.

In general, the boundary update works as follows: Let  $\psi_p^n$ ,  $\psi_p^{n+1}$  denote any of the above variables on the parent domain at time steps n and n+1, respectively. Once the model state on the parent domain  $\mathcal{M}_p$  has been updated from n to n+1, the time tendency

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

is diagnosed. Both, the field  $\psi_p^n$  at time level n and the tendency  $\frac{\partial \psi_p}{\partial t}$  are then interpolated (downscaled) from the parent grid cells/edges to the corresponding cells/edges of the child's boundary zone. With  $\mathcal{I}_{p\to c}$  denoting the interpolation operator, we get

$$\psi_c^n = \mathcal{I}_{p \to c} \left( \psi_p^n \right)$$
$$\frac{\partial \psi_c}{\partial t} = \mathcal{I}_{p \to c} \left( \frac{\partial \psi_p}{\partial t} \right)$$

The interpolated tendencies are generally needed in order to provide the lateral boundary conditions at the right time levels, since two integration steps are necessary on the child domain in order to reach the model state  $\mathcal{M}_c^{n+1}$ , with each step consisting of ndyn_substeps (nonhydrostatic_nml) dynamics sub-steps. E.g. for the first and second (large) integration step on the child domain the boundary conditions read  $\psi_c^n$  and  $\psi_c^n + 0.5 \Delta t_p \partial \psi / \partial t|_c$ , respectively.

Regarding the interpolation operator  $\mathcal{I}_{p\to c}$  we distinguish between cell based variables (i.e. scalars) and edge-based variables  $(v_n \text{ and } \rho v_n)$ . For cell based variables a 2D horizontal gradient is reconstructed at the parent cell center by first computing edge-normal gradients at edge midpoints, followed by a 9-point reconstruction of the 2D gradient at the cell center based on radial basis functions (RBF, Narcowich and Ward (1994)). The interpolated value at the *j*th child cell center is then calculated as

$$\psi_{c_j} = \psi_p + \nabla \psi_p \cdot \boldsymbol{d}(p, c_j), \qquad j \in \{1 \dots 4\}, \qquad (3.43)$$

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

To prevent excessive over- and undershoots of  $\psi_{c_j}$  in the vicinity of strong gradients, a limiter for  $\nabla \psi_p$  is implemented. It ensures that

$$\frac{1}{\beta}\psi_{p,\min} < \psi_{c_j} < \beta\psi_{p,\max} \qquad \forall \, j \in \{1 \dots 4\}$$

on all four child points, where  $\psi_{p,\min}$  and  $\psi_{p,\max}$  denote the minimum and maximum of  $\psi_p$ , respectively, on the above-mentioned reconstruction stencil plus the local cell center, and  $\beta = 1.05$  is a tuning parameter.

Regarding the interpolation of edge-based variables (i.e. the edge-normal vector components  $v_n$  and  $\rho v_n$ ), we distinguish between *outer child edges* that coincide with the edges of the parent cell, and *inner child edges* (see Figure 3.17a).

Edge-normal vector components at the inner child edges are reconstructed by means of a direct RBF reconstruction using the five-point stencil indicated in Figure 3.17a. For a given inner child edge the stencil comprises the edges of the corresponding parent cell, and the two edges of the neighboring parent cells that (approximately) share the orientation of the inner child edge.

For the outer child edges a more elaborate reconstruction is applied, in order to assure that the mass flux across a parent edge equals the sum of the mass fluxes across the corresponding child edges. We start with an RBF reconstruction of the 2D vector of the



Figure 3.17.: Horizontal reconstruction stencil for edge-normal vector components at (a) inner child edges and (b) outer child edges. The child edge under consideration is highlighted in red. Black open dots indicate child edge midpoints, while black solid dots indicate cell circumcenters. Solid red dots represent the reconstruction stencil, i.e. the location of the parent edge-normal vector components entering the reconstruction, and light-blue triangles in (b) indicate the location of the reconstructed 2D vectors. See the text for details.

respective variable at the triangle vertices, using the six (five at pentagon points) edge points adjacent to a vertex (see Figure 3.17b).

The edge-normal vector component  $\phi$  at the child edge is then computed as

$$\phi_{c_e} = \phi_p + \nabla_t \phi_p \cdot \boldsymbol{d}(p, c_e), \qquad e \in \{1, 2\},$$

with  $d(p, c_e)$  denoting the distance vector between the parent and child edge midpoints for a given parent edge, and  $\nabla_t \phi_p$  denoting the gradient of the edge-normal vector component  $\phi_p$  tangent to the parent edge. The latter is computed by projecting the 2D vectors at the two vertices of an edge onto the edge-normal direction and taking the centered difference. Since by construction  $d(p, c_1) = -d(p, c_2)$  holds on the ICON grid, the above mentioned mass flux consistency is ensured.

It is noted that attempts to use higher-order polynomial interpolation methods, which are the standard in mesoscale models with regular quadrilateral grids, were unsuccessful on the triangular ICON grid, because the ensuing equation system led to the inversion of nearly singular matrices.

In order to minimize interpolation errors, the following modifications from the above interpolation procedure are applied: For the thermodynamic variables  $\rho$  and  $\theta_v$  perturbations from the reference state rather than the full values are interpolated, in order to reduce interpolation errors above steep orography.

Rather than interpolating  $v_n$  and its time tendencies, only the time tendencies are interpolated, and then used to update  $v_n$  at child level at every dynamics time step. This methodology has been chosen because the comparatively inaccurate interpolation to the interior child edges tends to induce small-scale noise in  $v_n$ . To suppress the remaining noise arising from the interpolation of the time tendency, a second-order diffusion operator is applied in the inner half of the boundary interpolation zone on  $v_n$ , and the default fourth-order diffusion applied in the prognostic part of the model domain (see Zängl et al. (2015)) is enhanced in the five grid rows adjacent to the interpolation zone. For the other prognostic variables, no special filtering is applied near nest boundaries. In the case of one-way nesting, the second-order velocity diffusion is extended into the nudging zone of the nested domain, replacing the enhanced fourth-order diffusion. More details on the nudging zone are given in Section 3.9.1.

For the horizontal mass flux  $\rho v_n$ , the time average over the dynamic sub-steps, which is passed to the tracer transport scheme in order to achieve mass consistency, is interpolated instead of time level n. Using the mass flux time tendency that is interpolated as well, the related time shift is corrected for when applying the boundary mass fluxes at child level. In the nested domain, the interpolated mass fluxes valid for the current time step are then prescribed at the interface edges separating the boundary interpolation zone from the prognostic part of the nested domain. Due to the flux-form scheme used for solving the continuity equation (see Zängl et al. (2015)), this implies that the interpolated values of  $\rho$  do not enter into any prognostic computations in the dynamical core. They are needed, however, for some computations in the transport scheme. Moreover, no mass fluxes at interior child edges are used, so that the non-conservative interpolation method used for those edges does not affect the model's conservation properties. For  $\theta_v$  and the tracer variables  $q_k$ , the values at the edges are reconstructed in the usual manner (see Zängl et al. (2015)) and then multiplied with the interpolated mass fluxes before computing the flux divergences.

## Feedback: Child $\rightarrow$ Parent

If two-way nesting is activated (lfeedback=.TRUE., namelist grid_nml), the model state  $\mathcal{M}_p^{n+1}$  on the parent domain is relaxed towards the updated model state  $\mathcal{M}_c^{n+1}$  on the child domain at every fast physics time step. In the following we will refer to this as *relaxation-type feedback*. It is restricted to the prognostic variables  $v_n$ , w,  $\theta_v$ ,  $\rho$  plus specific humidity  $q_v$  and the specific contents of cloud water  $q_c$  and cloud ice  $q_i^5$ . Precipitating hydrometeors are excluded because recommended relaxation times (see below) are longer than their typical falling times. Surface variables are excluded as well because they can easily adjust during runtime and a proper treatment of feedback along land-cover inhomogeneities (e.g. coastlines) would be complicated and probably computationally expensive.

Let  $\psi$  denote any of the above mentioned variables. Conceptually, the feedback mechanism is based on the following three basic steps:

- 1. **Upscaling**: The updated field  $\psi_c^{n+1}$  is interpolated (upscaled) from the child domain to the parent domain. The upscaling operators for cell based and edge based variables will be denoted by  $\mathcal{I}_{c\to p}$  and  $\mathcal{I}_{ce\to p}$ , respectively.
- 2. Increment computation: The difference between the solution on the parent domain  $\psi_p^{n+1}$  and the upscaled solution  $\mathcal{I}_{c\to p}(\psi_c^{n+1})$  is computed.

⁵Note that when programming ICON the feedback mechanism can easily be switched on for additional tracers by adding the meta-information lfeedback=.TRUE. to the corresponding add_ref-call, see also Section 9.3.

3. **Relaxation**: The solution on the parent domain is relaxed towards the solution on the child domain. The relaxation is proportional to the increment computed in Step 2.

For cell based variables the upscaling consists of a modified barycentric interpolation from the four child cells to the corresponding parent cell:

$$\mathcal{I}_{c \to p}(\psi_c) = \sum_{j=1}^4 \alpha_j \psi_{c_j} \,.$$

The weights  $\alpha_j$  are derived from the following constraints (3.44)–(3.46). First of all, a desirable property for the value interpolation is that it reproduces constant fields, i.e. the weights are normalized:

$$\sum_{j=1}^{4} \alpha_j = 1.$$
 (3.44)

Moreover, the interpolation is linear: With the four child cell circumcenters  $x_j$  (j = 1, ..., 4), and  $x_p$  denoting the parent cell center, i.e. the interpolation target, we set

$$\sum_{j=1}^{4} \alpha_j (\boldsymbol{x}_j - \boldsymbol{x}_p) = 0. \qquad (3.45)$$

To motivate this constraint, consider the special case of equilateral triangles in which the center point of the inner child cell  $x_1$  coincides with the parent center such that the term  $(x_1 - x_p)$  vanishes. Equation (3.45) now defines a barycentric interpolation within the triangle spanned by the mass points of the three outer child cells  $\{c_2, c_3, c_4\}$  (see Figure 3.17a), where the weights  $\{\alpha_2, \alpha_3, \alpha_4\}$  represent the barycentric coordinates.

Of course, the contribution of the point  $x_1$  closest to the interpolation target is of particular importance. Therefore, the underdetermined system of equations (3.44), (3.45) is closed with a final constraint which reads as

$$\alpha_1 = \frac{a_{c_1}}{a_p} \,, \tag{3.46}$$

where  $a_{c_1}$  and  $a_p$  denote the inner child and parent cell areas, respectively. In other words, the inner child cell  $c_1$  containing the parent cell circumcenter is given a pre-defined weight corresponding to its fractional area coverage. This can be interpreted as a conservation constraint for the special case of a very localized signal at the mass point of the inner child cell.

In summary, this method can be regarded as a *modified* barycentric interpolation for the mass points  $\{x_2, x_3, x_4\}$ , and which accounts for  $x_1$  as an additional fourth source point. A more stringent barycentric interpolation would require an additional triangulation based on the child mass points. This is done for ICON's model output on regular lat-lon grids, see Section 7.1.2.

For velocity points, a simple arithmetic average of the two child edges lying on the parent edge is taken.

$$\mathcal{I}_{ce \to p}(v_{n,e}) = \frac{1}{2} \left[ v_{n,e_{\text{child } 1}} + v_{n,e_{\text{child } 2}} \right]$$

We note that the operator  $\mathcal{I}_{c\to p}$  is not strictly mass conserving and that strict mass conservation would require some means of area-weighted aggregation from the child cells to the parent cells, which is available as an option. The problem with such methods on the ICON grid is related to the fact that the mass points lie in the circumcenter rather than the barycenter of the triangular cells. Using an area-weighted aggregation from the child cells to the parent cells, would map linear horizontal gradients on the child grid into a checkerboard noise pattern between upward and downward oriented triangles on the parent grid.

Another difficulty that was encountered in the context of mass conservation is related to the fact that the density decreases roughly exponentially with height. In the presence of orography, the atmospheric mass resolved on the model grid therefore increases with decreasing mesh size, assuming the usual area-weighted aggregation of the orographic raw data to the model grid. Feeding back  $\rho$  is thus intrinsically non-conservative. To keep the related errors small and non-systematic, and to generally reduce the numerical errors over steep mountains, perturbations from the reference state are used for upscaling  $\rho$  and  $\theta_v$ to the parent grid. A closer investigation of the related conservation errors revealed that the differences between modified barycentric and area-weighted averaging are (with real orography) unimportant compared compared to the resolution-dependent conservation error.

When combining the above mentioned steps, the feedback mechanism for  $\rho$  can be cast into the following form:

$$\rho_{p}^{*} = \rho_{p}^{n+1} + \frac{\Delta t_{p}}{\tau_{fb}} \left( \mathcal{I}_{c \to p} (\rho_{c}^{n+1} - \Delta \rho_{corr}) - \rho_{p}^{n+1} \right)$$
(3.47)

Here  $\rho_p^{n+1}$  denotes the density in the parent cell, which has already been updated by dynamics and physics. The superscript "*" indicates the final solution, which includes the increment due to feedback.  $\Delta t_p$  is the fast physics time step on the parent domain, and  $\tau_{fb}$  is a user-defined relaxation time scale which has a default value of  $\tau_{fb} = 10800 \,\mathrm{s}$ . This value is motivated by the wish to exclude small scale transient features from the feedback, but to fully capture synoptic-scale features. The relaxation time scale is independent of the relaxed field and can be adjusted by means of the namelist variable fbk_relax_timescale (gridref_nml). Finally note that the upscaled quantity includes the correction term  $\Delta \rho_{corr}$  which has been introduced in order to account for differences in the vertical position of the child and parent cell circumcenters. At locations with noticeable orography, cell circumcenter heights at parent cells can differ significantly from those at child cells. If this is not taken into account, the feedback process will introduce a non-negligible bias in the parent domain's mass field. The correction term is given by

$$\Delta \rho_{corr} = \left(1.05 - 0.005 \,\mathcal{I}_{c \to p}(\theta_{v,c}^{\prime \, n+1})\right) \,\Delta \rho_{ref,p} \,,$$

with the parent-child difference in the reference density field

$$\Delta \rho_{ref,p} = \mathcal{I}_{c \to p}(\rho_{ref,c}) - \rho_{ref,p}$$

and the potential temperature perturbation  $\theta_{v,c}^{\prime n+1} = \theta_{v,c}^{n+1} - \theta_{v\,ref,c}$ . The term  $\Delta \rho_{ref,p}$  is purely a function of the parent-child height difference and can be regarded as a first order correction term. In order to minimize the remaining mass drift, the empirically determined

factor  $(1.05 - 0.005 \mathcal{I}_{c \to p}(\theta'_v^{n+1}))$  was added, which introduces an additional temperature dependency. Note that the factor 0.005 is close to near surface values of  $\frac{\partial \rho}{\partial \theta}$  which can be derived from the equation of state. We further note that a possibly more accurate and less ad hoc approach would require a conservative remapping step in the vertical, prior to the horizontal upscaling.

Care must be taken to ensure that the feedback process retains tracer and air mass consistency. To this end, feedback is not implemented for tracer mass fractions directly, but for partial densities. In accordance with the implementation for  $\rho$ , we get

$$(\rho q_k)_p^* = (\rho q_k)_p^{n+1} + \frac{\Delta t_p}{\tau_{fb}} \left[ \mathcal{I}_{c \to p} ((\rho_c^{n+1} - \Delta \rho_{corr}) q_{k,c}^{n+1}) - (\rho q_k)_p^{n+1} \right]$$
(3.48)

Mass fractions are re-diagnosed thereafter:

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

When summing Eq. (3.48) over all partial densities, Eq. (3.47) for the total density is recovered.

A very similar approach is used for  $\theta_v$ . As for  $\rho$ , only the increment of  $\theta_v$  is upscaled from the child- to the parent domain and added to the parent reference profile  $\theta_{vref,p}$ .

$$\theta_{v,p}^* = \theta_{v,p}^{n+1} + \frac{\Delta t_p}{\tau_{fb}} \left( \mathcal{I}_{c \to p}(\theta_{v,c}^{\prime n+1}) + \theta_{v \, ref,p} - \theta_{v,p}^{n+1} \right)$$

The same approach is taken for w, however the full field is upscaled.

$$w^* = w^{n+1} + \frac{\Delta t_p}{\tau_{fb}} \left( \mathcal{I}_{c \to p}(w_c^{n+1}) - w_p^{n+1} \right)$$

In the case of  $v_n$  some numerical diffusion is added to the resulting feedback increment in order to damp small-scale noise.

$$v_{n,p}^* = v_{n,p}^{n+1} + \frac{\Delta t_p}{\tau_{fb}} \left( \Delta v_{n,p} + K \nabla^2 \left( \Delta v_{n,p} \right) \right) ,$$

with the feedback increment

$$\Delta v_{n,p} = \mathcal{I}_{ce \to p}(v_{n,c}^{n+1}) - v_{n,p}^{n+1},$$

and the diffusion coefficient  $K = \frac{1}{12} \frac{a_{p,e}}{\Delta t_p}$ , where  $a_{p,e}$  is the area of the quadrilateral spanned by the vertices and centers adjacent to the parent's edge.

#### Lateral Nudging

If the feedback is turned off, i.e. if one-way nesting is chosen, a nudging of the prognostic child grid variables towards the corresponding parent grid values is needed near the lateral nest boundaries in order to accommodate possible inconsistencies between the two grids, particularly near the outflow boundary. Because lateral boundaries are in general not straight lines on the unstructured ICON grid, attempts to make an explicit distinction between inflow and outflow boundaries (e.g. by prescribing  $v_n$  at inflow boundaries only) were not successful.

To compute the nudging tendencies, the child grid variables are first upscaled to the parent grid in the same way as for the feedback, followed by taking the differences between the parent-grid variables and the upscaled child-grid variables. The differences are then interpolated to the child grid using the same methods as for the lateral boundary conditions (see above). The relaxation uses weighting factors decreasing exponentially from the inner margin of the boundary interpolation zone towards the interior of the model domain. Its width and the relaxation time scale have default values of 8 cell rows and  $0.02 \Delta \tau$  (dynamics time step), respectively, and the nudging coefficients decay with an e-folding width of 2 cell rows. These values can be adjusted by means of the namelist parameters nudge_zone_width, nudge_max_coeff, and nudge_efold_width in the namelist interpol_nml. See Section 6.2 for additional details on lateral boundary nudging. As already mentioned, a second-order diffusion on  $v_n$  is used near the lateral nest boundaries in order to suppress small-scale noise.

## Vertical Nesting

The vertical nesting option allows to set model top heights individually for each domain, with the constraints that the child domain height is lower or at most equal to the parent domain height, and that the child domain extends into heights where the coordinate surfaces are flat (see namelist parameter flat_height (sleve_nml) for the SLEVE vertical coordinate). This allows, for instance, a global domain extending into the mesosphere to be combined with a child domain that extends only up to the lower stratosphere (see Figure 3.18). However, a vertical refinement in the sense that the vertical resolution in the child domain may differ from that in the parent domain is not available. One possible workaround might be to repeat the model run with the desired vertical resolution in limited area mode (see Chapter 6).

In order to reduce the top height for child domains, the namelist parameter num_lev (run_nml), which specifies the number of vertical levels in each domain, must be adapted. In addition, vertical nesting must be enabled by setting lvert_nest=.TRUE.. See Section 3.4 for details.

Vertical nesting requires appropriate boundary conditions for all prognostic variables to be specified at the vertical nest interface level, i.e. the uppermost half level of the nested domain. This is crucial in order to prevent vertically propagating sound and gravity waves from being spuriously reflected at the nest interface. For details regarding the derivation of these boundary conditions, the reader is referred to Zängl et al. (2022).

## 3.9.2. Processing Sequence

So far, we have focused on the coupling of an individual parent and child domain. The coupling of multiple and possibly repeatedly nested domains requires a well conceived processing sequence, whose basics will be described in the following.



Figure 3.18.: Illustration of ICON's vertical nesting. Note that the nested domain may have a lower top level height, while the remaining vertical layers must match between the nested and the parent domain. Further note that the parent-nest levels do not coincide in a strict geometrical sense. Differences exist, caused by the resolution-dependent topography, which are omitted in this figure for clarity.

Figure 3.19 provides a common example where a global domain is combined with two repeatedly nested domains (two-way). The global domain is schematically depicted at the bottom, whereas the nested domains are vertically staggered on top of it. The red and blue regions show the boundary interpolation zones and feedback zones of the individual domains, respectively. The integration time step on the global domain is denoted by  $\Delta t$ . It is automatically reduced by a factor of 2 when moving to the next child grid level.

The processing sequence for the integration of all domains from time step n to n + 1 is shown in the flowchart at the lower left of Figure 3.19. The domains are ordered top down. Open and filled black dots show model states without and with feedback increments, black arrows indicate time integration, and red and blue arrows indicate lateral boundary data interpolation and feedback, respectively.

From an abstract point of view, the flow control of ICON's hierarchical nesting scheme is handled by a recursive subroutine that cascades from the global domain down to the deepest nesting level and for each domain calls the time stepping and the physics parameterizations in basically the same way as for the global domain. The basic processing sequence is as follows:

- 1. A single integration step with  $\Delta t$  is performed on the global domain which, results in an updated model state  $\mathcal{M}_p^{n+1}$ , indicated by an open black circle.
- 2. Boundary data are interpolated from the global domain to the first nested domain (red arrow), followed by an integration step on nested domain 1 over the time interval  $\Delta t/2$ .
- 3. As there exists another nested domain within nest 1, boundary fields based on the model state  $\mathcal{M}_{c1}^{n+1/2}$  are interpolated to the second nested domain. Afterwards, the



Figure 3.19.: Schematic of a global domain with two repeatedly nested domains (twoway). The processing sequence for the time integration of all domains from time step n to n + 1 is shown in the flowchart at the lower left. See the text for details.

model is integrated on nested domain 2 over two times the time interval  $\Delta t/4$ , resulting in the model state  $\mathcal{M}_{c2}^{n+1/2}$ .

- 4. Feedback is performed from nest 2 back to nest 1 (blue arrow), which results in an updated model state  $\mathcal{M}_{c1}^{n+1/2*}$  on nested domain 1 (black filled dot). Then, on nested domain 1 the model is again integrated in time to reach model state  $\mathcal{M}_{c1}^{n+1}$ .
- 5. This is followed by a second lateral boundary data interpolation from nest 1 to nest 2 based on  $\mathcal{M}_{c1}^{n+1}$ . Nest 2 is integrated in time again, to reach its state  $\mathcal{M}_{c2}^{n+1}$ .
- 6. As a final step, feedback is performed from nest 2 to nest 1, followed by feedback from nest 1 to the global domain.

## 3.9.3. Technical and Performance Aspects

Several measures are taken in order to optimize the computational efficiency of the nesting implementation.

In the model grids, grid points lying at or near the lateral boundary of a nested domain are shifted to the beginning of the index vector, ordered by their distance from the lateral boundary. This allows excluding boundary points from prognostic computations accessing non-existing neighbor points without masking operations. In the present implementation, the four outer cell rows constituting the boundary interpolation zone (see Figure 3.16), and the adjacent fifth one participate in the reordering. For additional information on the grid point reordering see Section 9.1 and in particular Figure 9.2.

The reordering makes use of the grid meta-data field refin_c_ctrl which counts the distance in units of cell rows (see Figure 2.12). Correspondingly, there are integer flag arrays for edges and vertices replicating the distance information from the lateral boundary. This distance information is extended to a larger number of grid rows in order to provide the geometric information needed for lateral boundary nudging. Moreover, the flag arrays signify grid points overlapping with a child domain, including a distinction between boundary interpolation points and interior overlap points.

Regarding distributed-memory (MPI) parallelization, the general strategy adopted in ICON is to distribute all model domains among all compute processors. As this implies that child grid points are in general owned by a different processor than the corresponding parent grid point, an intermediate layer having the resolution of the parent grid but the domain decomposition of the child grid is inserted in order to accommodate the data exchange required for boundary interpolation and feedback.

To reduce the amount of MPI communication for complex nested configurations, multiple nested domains at the same nesting level can be merged into one logical domain which is then not geometrically contiguous. This needs to be done during the grid generation process by indicating a list of domains via the grid generator's namelist parameter merge_domain (see Prill, 2020). The lateral boundary points belonging to all components of the merged domain are then collected at the beginning of the index vector. For all prognostic calculations, the multiple domains are treated as a single logical entity, and just the output files may be split according to the geometrically contiguous basic domains. As one-way and two-way nesting cannot be mixed within one logical domain, there may still be two logical domains on a given nest level.

To further optimize the amount of MPI communication, a so-called processor splitting is available that allows for executing several nested domains concurrently on processor subsets whose size can be determined by the user in order to minimize the ensuing load imbalance. This option is currently restricted to the step from the global domain to the first nesting level in order to keep the technical complexity at a manageable level.

# 3.10. Reduced Radiation Grid

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

The implementation is schematically depicted in Figure 3.20:

- 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  $\tau^{SW}$  and longwave fluxes  $F^{LW}$  are scaled down to the full grid.
- Step 3. In a last step we apply empirical corrections to  $\tau^{SW}$  and  $F^{LW}$  in order to incorporate the high resolution information about albedo  $\alpha$  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:



Figure 3.20.: Schematic showing how radiation is computed on a reduced (coarser) grid.

## lredgrid_phys = .FALSE./.TRUE. (namelist grid_nml, logical value)

If set to .TRUE. radiation is calculated on a coarser grid (i.e. one grid level coarser)

## radiation_grid_filename (namelist grid_nml, string parameter)

Filename of the grid to be used for the radiation model. Must only be specified for the base domain, since for child domains the grid of the respective parent domain serves as radiation grid. An empty string is required, if radiation is computed on the full (non-reduced) grid.

Note that running radiation on a reduced grid is the standard setting for operational runs at DWD. Using the reduced radiation grid is also possible for the limited area mode ICON-LAM. In this case, both the computational grid and the reduced radiation grid are regional grids. Make sure to create the latter during the grid generation process by setting dom(:)%lwrite_parent = .TRUE., see Section 2.1.5. Internally, the coarse radiation grid is denoted by the domain index 0.

# 4. Running Idealized Test Cases

The majority of idealized test cases does not require external parameter or analysis fields for initialization. Initial conditions are usually computed within the ICON model itself, based on analytical functions. These are either evaluated point-wise at cell centers, edges, or vertices, or are integrated over the triangular control volume to provide cell averages.

The ability to run idealized model setups serves as a simple means to test the correctness of particular aspects of the model, either by comparison with analytic reference solutions (if they exist), or by comparison with results from other models. Beyond that, idealized test cases may help the scientist to focus on specific atmospheric processes.

ICON provides a set of pre-defined test cases of varying complexity and focus, ranging from pure dynamical core and transport test cases to "moist" cases, including microphysics and potentially other parameterizations. A complete list of available test cases can be found in the namelist documentation, mentioned in Section 1.1.3.

Individual test cases can be selected and configured by namelist parameters of the namelist nh_testcase_nml. To run one of the implemented test cases, only a horizontal grid file has to be provided as input. A vertical grid file containing the height distribution of vertical model levels is usually not required, since the vertical grid is constructed within the ICON model itself, based on the set of namelist parameters described in Section 3.4.

From the set of available test cases we choose the Jablonowski-Williamson baroclinic wave test and Straka density current test and walk through the procedure of configuring and running these tests in ICON.

# 4.1. Main Switches for Idealized Test Cases

This section explains several namelist groups and main switches that are necessary for setting up an idealized model run.

## 4.1.1. Activating/De-activating Main Model Components

#### Namelist run_nml:

ltestcase = .TRUE./.FALSE. (namelist run_nml, logical value)
This parameter must be set to .TRUE. for running idealized test cases.

ldynamics = .TRUE./.FALSE. (namelist run_nml, logical value) Main switch for the dynamical core. If set to .TRUE., the dynamical core is switched on and details of the dynamical core can be controlled via dynamics_nml, nonhydrostatic_nml and diffusion_nml. If set to .FALSE., the dynamical core is switched off completely. This is rarely needed, but can be useful for idealized tests of physical parameterizations with prescribed dynamical forcing.

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

Main switch for the transport of passive tracers. If set to .TRUE., transport is switched on and details of the transport schemes can be controlled via transport_nml (see Section 3.6.5 for additional help). If set to .FALSE., transport of passive tracers is switched off completely.

#### iforcing = 0/2/3 (namelist run_nml, integer value)

Forcing of dynamics and transport by parameterized processes. If set to 0, forcing is switched off completely (pure dynamical core test case). This implies that all physical parameterizations are switched off automatically. If set to 3, dynamics are forced by NWP-specific parameterizations. Individual physical processes can be controlled via nwp_phy_nml, see also Table 3.4. If set to 2, the AES parameterization suite is used. In general, the setting of iforcing depends on the selected test case.

#### msg_level (namelist run_nml, integer value)

You may increase the model output verbosity by setting this namelist parameter to a higher value ( $\leq 20$ ). This option can be particularly useful if the ICON model run fails and the cause of the error still does not become clear from the error message.

#### Namelist dynamics_nml:

## lcoriolis = .TRUE./.FALSE. (namelist dynamics_nml, logical value)

Main switch for activation/deactivation of the Coriolis force. In general, the setting depends on the selected test case.

#### Namelist extpar_nml:

#### itopo = 0/1 (namelist extpar_nml, integer value)

If set to 1, the model tries to read topography data and external parameters from file. If set to 0, no input file is required for model initialization. Instead, all initial conditions are computed within the ICON model itself. Usually, itopo is set to 0 for running idealized test cases.

## 4.1.2. Specifying the Computational Domain(s)

ICON's computational domain(s) is/are specified via the following namelist:

#### Namelist grid_nml:

#### dynamics_grid_filename (namelist grid_nml, list of string parameters)

Here, the name(s) of the horizontal grid file(s) must be specified. For a global simulation without nests, of course, only a single filename is required. For a global simulation with multiple nests a filename must be specified for each domain. Note that each name must be enclosed by single quotation marks and that multiple names must be separated by a comma (see the examples below).

#### Namelist run_nml:

#### num_lev (namelist run_nml, list of integer value)

Comma-separated list of integer values specifying the number of vertical full levels for each domain.

**Example:** Assuming that a global horizontal grid file is provided (named icon_grid_0014_R02B05_G.nc) the settings for a global run with 40 vertical levels are as simple as follows:

dynamics_grid_filename = 'icon_grid_0014_R02B05_G.nc'
num_lev = 40

## 4.1.3. Integration Time Step and Simulation Length

The integration time step and simulation length are defined via the following namelist:

#### Namelist run_nml:

#### dtime (namelist run_nml, real value)

Time step in seconds (for the top-most domain). Note that it is *not* necessary to specify a time step for each domain. For each nesting level, the time step is automatically divided by a factor of two. More details on ICON's time step are given in Section 3.7.1.

#### nsteps (namelist run_nml, integer value)

Number of time steps. An alternative way for setting the simulation length is to specify the simulation start and end date, see Section 5.1.1.

Output is controlled by the namelist group output_nml. It is possible to define more than one output namelist and each output namelist has its own output file attached to it. The details of the model output specification are discussed in Section 7.

# 4.2. Jablonowski-Williamson Baroclinic Wave Test

In order to activate the Jablonowski-Williamson baroclinic wave test, select:

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

The Jablonowski-Williamson baroclinic wave test (Jablonowski and Williamson, 2006) has become one of the standard test cases for assessing the quality of dynamical cores. The model is initialized with a balanced initial flow field. It comprises a zonally symmetric base state with a jet in the mid-latitudes of each hemisphere and a quasi realistic temperature distribution. Overall, the conditions resemble the climatic state of a winter hemisphere. This initial state is in hydrostatic and geostrophic balance, but is highly unstable with respect to baroclinic instability mechanisms. Thus, it should remain stationary if no perturbation is imposed.

To trigger the evolution of a baroclinic wave in the northern hemisphere, the initial conditions are overlaid with a weak (and unbalanced) zonal wind perturbation. The perturbation is centered at 20°E, 40°N. In general, the baroclinic wave starts growing observably around day 4 and evolves rapidly thereafter with explosive cyclogenesis around model day 8. After day 9, the wave train breaks (see Figure 4.1). If the integration is continued, additional instabilities become more and more apparent especially near the pentagon points (see Section 2.1), which are an indication of spurious baroclinic instabilities triggered by numerical discretization errors. In general, this test has the capability to assess

- the diffusivity of a dynamical core,
- the presence of phase speed errors in the advection of poorly resolved waves,
- the strength of grid imprinting.

## 4.2.1. Recommended Namelist Settings

A complete list of the recommended namelist settings is given in Table 4.1. The three parameters listed below are specific to the Jablonowski-Williamson test case and are therefore explained in more detail. Default values are given in red.

### jw_up = 1.0 (namelist nh_testcase_nml, real value)

Amplitude of the u-perturbation in  $m s^{-1}$ . If this parameter is set to 0, the model's ability to maintain the initial steady state can be tested.

```
jw_u0 = 35.0 (namelist nh_testcase_nml, real value)
Maximum zonal wind in ms<sup>-1</sup>
```

```
jw_temp0 = 288.0 (namelist nh_testcase_nml, real value)
horizontal-mean temperature at surface in K
```



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

# 4.2.2. Enabling Passive Tracers

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

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

In the ICON mode, four different tracer distributions are implemented, whose initial distributions are depicted in Figure 4.2. See Jablonowski et al. (2008) for further information on the initial distributions.

Please follow the steps below, in order to enable the transport of one or more predefined tracers:

• Enable the transport module by activating the main switch ltransport=.TRUE. (namelist run_nml).

## ICON Model Tutorial

Namelist	Parameter	Unit	Value
nh_testcase_nml	nh_test_name		'jabw'
	jw_up	m/s	1.0
	jw_u0	m/s	35.0
	jw_temp0	Κ	288.0
run_nml	ltestcase		.TRUE.
	ldynamics		.TRUE.
	ltransport		.FALSE.
	iforcing		0
	num_lev		40
	dtime	s	576
	nsteps		1500
dynamics_nml	lcoriolis		.TRUE.
extpar_nml	itopo		0
grid_nml	dynamics_grid_filename		'icon_grid014_R02B05_G.nc'
sleve_nml	top_height	m	35000
nonhydrostatic_nml	vwind_offctr		0.2
	exner_expol		0.5
	damp_height	m	25000
	rayleigh_coeff		0.1
diffusion_nml	hdiff_order		5
	itype_vn_diffu		2
	itype_t_diffu		2
	hdiff_efdt_ratio		20

- **Table 4.1.:** Recommended namelist settings for the global Jablonowski-Williamson baroclinic wave test case at a horizontal resolution of  $\approx 80 \text{ km}$  (R2B05).
  - Select one or more tracers from the set of pre-defined tracer distributions depicted in Figure 4.2. A specific tracer can be selected by adding the respective tracer number (1,2,3, or 4) to the following namelist variable:
    - tracer_inidist_list(namelist nh_testcase_nml, list of integer values) Comma-separated list of integer values. A value of 1 selects tracer q1, 2 selects q2, and so on. If the list is empty, no passive tracer will be transported.
  - Set the total number of tracers ntracer (namelist run_nml) accordingly.



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

• Add the selected tracers to the list of output fields in the namelist output_nml (namelist parameters ml_varlist and/or pl_varlist).

By default, tracers in idealized tests are named qx, where x is a number indicating the position of the tracer within the ICON-internal 4D tracer container. For this testcase, tracer 1 is named q1, and so on.

Alternatively, the default names can be overwritten via the namelist variable tracer_names (namelist transport_nml, comma-separated list of string parameters). The  $n^{th}$  entry in tracer_names corresponds to the  $n^{th}$  entry in tracer_inidist_list.

## 4.2.3. Activation of Nested Domains

The Jablonowski-Williamson test is well suited to acquaint oneself with ICON's nesting capability, which is described in Section 3.9. The test is used e.g. by Zängl et al. (2022) in order to demonstrate the functionality of the grid nesting and to investigate related numerical errors.

Activating nested domains requires only a small number of additional namelist settings. In the following, we assume that three horizontal grid files are given:



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

icon_grid_0014_R02B05_G.nc icon_grid_0014_R02B05_N06_1.nc icon_grid_0014_R02B05_N06_2.nc,

which contain a global grid and two nested grids at the same nesting level, respectively (see Figure 4.3 for the suggested location and extent of the nested domains). Additional information on the grid file naming convention can be found in Section 2.1. One or both nested domains may be activated by namelist settings:

**Example 1:** Settings for a global run with nest number 2 and 40 vertical levels each:

```
dynamics_grid_filename =
'icon_grid_0014_R02B05_G.nc','icon_grid_0014_R02B05_N06_2.nc'
num_lev = 40,40
```

Example 2: Settings for a global run with both nests and 40 vertical levels each:

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

The parent-child relationships of the individual domains/nests are inferred automatically from the NetCDF attributes uuidOfHGrid and uuidOfParHGrid in the grid files, see Section 3.9.

# 4.3. Straka Density Current Test

Another well-known test case for the evaluation and intercomparison of dynamical cores is the nonlinear 2D density current test case described by Straka et al. (1993). See e.g. Gallus and Ranĉić (1996), Satoh (2002), Skamarock and Klemp (2008), Guerra and Ullrich (2016) for example applications.

In this test case a circular shaped bubble of cold air is initialized a few kilometers above ground in a neutrally stratified ( $\theta = 300 \text{ K}$ ) and hydrostatically balanced atmosphere at rest. The Coriolis force is set to zero. When integrated forward in time, the cold air bubble


Figure 4.4.: Straka density current test case reference results for ICON with  $\Delta x = \Delta z \approx 25 \text{ m}$ . Contours show the potential temperature for  $t = 0 \min, 8 \min, 15 \min$ , respectively. The contour interval is 1 K. Due to the symmetry of the setup only the right moving density current is shown.

accelerates towards the ground and forms two symmetric density currents which spread laterally along the bottom boundary and form shear-driven Kelvin-Helmholtz (K-H) instabilities along their top (see Figure 4.4). The simulation is scale limited due to the application of a second order diffusion operator for potential temperature and momentum with constant diffusion coefficients  $K_h = K_m = 75 \,\mathrm{m}^2 \mathrm{s}^{-1}$ . The resolvable scales are, hence, limited by the viscosity of the simulated medium rather than the spatio-temporal resolution. This enables the computation of a grid-converged reference solution. Straka et al. (1993) showed that for  $\Delta x = \Delta z \approx 25 \,\mathrm{m}$  the result can be regarded as grid-converged, since any additional resolution increase does not have a noticeable effect. For visual intercomparison Straka et al. (1993) provided reference solutions of various dynamical cores. The  $\Delta x = \Delta z \approx 25 \,\mathrm{m}$  reference solution of ICON is shown in Figure 4.4. The computational domain consists of a quasi 2D torus grid (see Section 2.1.9) with doubly-periodic boundary conditions and a width of (at least) 40 km in zonal direction¹. In meridional direction, the domain consist of 4 cell rows (see Figure 4.5). Due to technical reasons it is not possible to further reduce the number of rows, however, the dynamical core gives identical results for each of these rows. The domain height is set to H = 6400 m, and the upper and lower boundary are treated as rigid lid (no flux).



Figure 4.5.: Schematic of the quasi-2D Straka test torus grid, which consists of 4 cell rows in meridional direction. Due to technical reasons it is currently not possible to further reduce the number of rows.

The initial temperature disturbance is applied to the  $\theta$  field and is given by

$$\Delta T = \begin{cases} 15.0 \cos^2\left(\frac{\pi}{2}L\right) &, \text{ if } L \le 1\\ 0.0 &, \text{ if } L > 1, \end{cases}$$

with

$$L = \left[ \left( \frac{x - x_c}{x_r} \right)^2 + \left( \frac{z - z_c}{z_r} \right)^2 \right]^{\frac{1}{2}}.$$

The cold bubble is initially located at  $(x_c, z_c) = (0 \text{ km}, 3 \text{ km})$  and has a radius of  $(x_r, z_r) = (4 \text{ km}, 2 \text{ km})$ .

In general, this test can be used to assess (among other things):

- the order of convergence of the dynamical core
- the quality at resolutions much coarser than  $\Delta x = \Delta z \approx 25$  m. I.e. which resolution is necessary in order to resolve all three K-H rotors?
- the magnitude of phase speed errors, by adding a nonzero background wind and comparing the right- and left moving currents (see also Skamarock and Klemp (2008)).

A complete list of the recommended namelist settings is given in Table 4.2.

#### 4.3.1. Relevant Namelist Switches in nh_testcase_nml:

We conclude this section by a list of parameters that can be used to modify the Straka setup. Again, default values are given in red. They do not necessarily coincide with the recommended Straka settings (see Table 4.2).

¹Results are usually compared after t = 15 min simulation time, when the wave front has traveled approximately 15 km in both directions. Hence, a domain width of 40 km should be sufficient to avoid significant disturbances along the lateral boundaries.

- bub_hor_width = 1000.0 (namelist nh_testcase_nml, real value)
  Horizontal radius of the thermal perturbation in m
- bubctr_z = 1400.0 (namelist nh_testcase_nml, real value)
  Height of the center of the thermal perturbation in m
- bub_amp = 2.0 (namelist nh_testcase_nml, real value)
  Maximum amplitude of the center of the thermal perturbation in K
- $nh_brunt_vais = 0.01$  (namelist  $nh_testcase_nml$ , real value) Initial Brunt-Väisälä frequency (constant with height) in s⁻¹

## nh_u0 = 0.0 (namelist nh_testcase_nml, real value)

Initial constant zonal wind speed. Can be used to break the symmetry of the Straka test case (see Section 4.3).

### ICON Model Tutorial

Namelist	Parameter	Unit	Value
nh_testcase_nml	nh_test_name		'straka93'
	nh_brunt_vais	$s^{-1}$	0.0
	bubctr_z	m	3000
	bub_hor_width	m	4000
	bub_ver_width	m	2000
	bub_amp	K	-15
run_nml	ltestcase		.TRUE.
	ldynamics		.TRUE.
	ltransport		.FALSE.
	iforcing		3
	num_lev		256
	dtime	S	0.18
	nsteps		6800
dynamics_nml	lcoriolis		.FALSE.
extpar_nml	itopo		0
grid_nml	dynamics_grid_filename		'plane-grid_1600_dx25.0.nc
	is_plane_torus		.TRUE.
sleve_nml	top_height	m	6400
	min_lay_thckn		0.0
diffusion_nml	hdiff_order		3
	hdiff_efdt_ratio		10
	hdiff_smag_fac		0.12
turbdiff_nml	lconst_z0		.TRUE.
	const_z0	m	0.0003
nwp_phy_nml	inwp_turb		5
les_nml	is_dry_cbl		.TRUE.
	isrfc_type		0
	ufric		0.0
	<pre>smag_coeff_type</pre>		2
	Km_ext	$m^2s^{-1}$	75.0
	Kh_ext	$m^2s^{-1}$	75.0

Table 4.2.: Recommended namelist settings for the Straka density current test case at a horizontal resolution of 25 m on a plane torus grid. Please note that all NWP physics parameterizations except for turbulence (inwp_turb) must be switched off.

# 5. Running Real Data Test Cases

In this chapter you will learn about how to initialize and run the ICON model in a realistic NWP setup. The namelist settings to start from a DWD Analysis and from an IFS Analysis are discussed.

### 5.1. Model Initialization

The necessary input data to perform a real data run have already been described in Chapter 2. These include

- grid files, containing the horizontal grid information,
- external parameter files, providing information about the Earth's soil and land properties, as well as climatologies of atmospheric aerosols, and
- initial data (analysis) for atmosphere, land and sea.

ICON is capable of reading analysis data from various sources (see Section 2.2), including data sets generated by DWD's Data Assimilation Coding Environment (DACE) and interpolated IFS data. In the following we provide some guidance on how to set up real data runs, depending on the specific data set at hand.

Note that ICON aborts during the setup phase, if any of the required input files has not been found. Therefore, as a first step, check the filenames (and soft links) for the model input files, (see Section 2).

Also make sure that the input data and grid files match. For example, take a look at the global attributes number_of_grid_used and uuidOfHGrid of the grid file(s). These values have to match the corresponding attributes of the external parameters and initial data file(s), see Section 2.1.8.

### 5.1.1. Basic Settings for Running Real Data Runs

Most of the main switches, that were used for setting up idealized test cases, are also important for setting up real data runs. As many of them have already been discussed in Chapter 4, we will concentrate on their settings for real data runs. Settings appropriate for the exercises on this subject are highlighted in red.



Figure 5.1.: Graphical illustration of the parameters for model start and end. The starting and termination of nested domains is explained in Section 5.2. Please also note Fig. 7.3, in which the program sequence is extended by restart.

### Specifying Model Start and End Dates (Namelist time_nml)

For real case runs it is important that the user specifies the correct start date and time of the simulation, see Fig. 5.1.

In ICON there coexist two equally usable ways to control the experiment start and end date – without a compelling reason, though. These two alternatives are listed in the following.

Namelist run_nml:			
time step	dtime	modelTimeStep	
Namelists time_nml (left) and master_time_control_nml (right):			
experiment start	ini_datetime_string	experimentStartDate	
experiment stop	<pre>end_datetime_string</pre>	experimentStopDate	

Please note that the data types of the above-mentioned namelist parameters differ. The parameters that are listed on the right are consistently based upon the ISO 8601 representations of dates and time spans. However, dtime must be specified in seconds.

In the examples of this tutorial, start and end dates are given with ini_datetime_string using the ISO8601 format:

### ini_datetime_string = YYYY-MM-DDThh:mm:ssZ (namelist time_nml)

— This must exactly match the validity time of the analysis data set!

Wrong settings lead to incorrect solar zenith angles and wrong external parameter fields. Setting the end date and time of the simulation via end_datetime_string is optional. If end_datetime_string is not set, the user has to set the number of time steps explicitly in nsteps (run_nml), which is otherwise computed automatically.

### General Settings (Namelist run_nml)

### ltestcase= .FALSE. (namelist run_nml, logical value)

This parameter must be set to .FALSE. for real case runs.

### iforcing= 3 (namelist run_nml, integer value)

A value of 3 means that dynamics are forced by NWP-specific parameterizations.

### ldynamics= .TRUE. (namelist run_nml, logical value)

The dynamical core must, of course, be switched on.

### ltransport= .TRUE. (namelist run_nml, logical value)

Tracer transport must be switched on. This is necessary for the transport of cloud and precipitation variables. Details of the transport schemes can be controlled via the namelist transport_nml (see Section 3.6.5).

### Specifying the Horizontal Grid (Namelist grid_nml)

### dynamics_grid_filename (namelist grid_nml, list of string parameters)

Here, the name(s) of the horizontal grid file(s) must be specified. For a global simulation without nests, of course, only a single filename is required. For a global simulation with multiple nests, a filename must be specified for each domain. Note that each name must be enclosed by single quotation marks and that multiple names must be separated by a comma (see Section 4.1.2 and the examples therein).

#### radiation_grid_filename (namelist grid_nml, string parameter)

If the radiative transfer computation should be conducted on a coarser grid than the dynamics (one level coarser, effective mesh size  $2\Delta x$ ), the name of the base grid for radiation must be specified here. See Section 3.10 for further details.

### Specifying External Parameters (Namelist extpar_nml)

#### itopo= 1 (namelist extpar_nml, integer value)

For real data runs this parameter must be set to 1. The model now expects one file per domain from which it tries to read topography data and external parameters.

#### extpar_filename (namelist extpar_nml, string parameter)

Filename(s) of input file(s) for external parameters. If the user does not provide namelist settings for extpar_filename, ICON expects one file per domain to be present in the experiment directory, following the naming convention

### extpar_filename = "extpar_<gridfile>.nc"

The keyword *<gridfile>* is automatically replaced by ICON with the grid filename specified for the given domain (dynamics_grid_filename). As opposed to the grid-file specification namelist variables (see above), it is not allowed to provide a comma-separated list. Instead, the usage of keywords provides full flexibility for defining the filename structure.

By changing the above setting, the user has full flexibility with respect to the filename structure. The following keywords are allowed for the namelist parameter extpar_filename. The keywords are automatically replaced by ICON with the content described in the right column below.

<path></path>	model base directory
	(namelist parameter model_base_dir, namelist master_nml)
<gridfile></gridfile>	grid filename for the given domain (dynamics_grid_filename)
<nroot></nroot>	grid root division $\mathbf{R}\boldsymbol{x}$ (single digit)
<nroot0></nroot0>	grid root division $Rxx$ (two digits)
<jlev></jlev>	grid bisection level Byy (two digits)
<idom></idom>	domain number (two digits).

### Specifying the Initialization Mode (Namelist initicon_nml)

ICON provides different real data initialization modes which differ in terms of the expected input fields and number of input files. Thereby ICON is able to handle analysis products from different models. The mode in use is controlled via the namelist switch init_mode.

### init_mode (namelist initicon_nml, integer value)

It is possible to

- start from (interpolated) uninitialized DWD analysis without the IAU procedure: init_mode = 1
- start from interpolated IFS analysis: init_mode = 2
- start atmosphere from interpolated IFS analysis and soil/surface from interpolated ICON/GME fields: init_mode = 3
- start from non-interpolated, *uninitialized* DWD analysis, and make use of the IAU procedure to filter initial noise: init_mode = 5
- start from interpolated *initialized* ICON analysis with subsequent vertical remapping: init_mode=7

The most relevant modes are mode 1, 2, 5 and 7. They will be explained in more detail below.

ICON supports NetCDF and GRIB2 as input format for input fields. In this context it is important to note that the field names that are used in the input files do not necessarily coincide with the field names that are internally used by the ICON model. To address this problem, an additional input text file is provided, a so-called *dictionary file*. This file translates between the ICON variable names and the corresponding GRIB2/NetCDF short names.

Generally the dictionary is provided via the following namelist parameter:

### 5.1 Model Initialization

```
ana_varnames_map_file (namelist initicon_nml, string parameter)
```

Filename of the dictionary for mapping between internal names and GRIB2/NetCDF short names. An example can be found in icon/run/ana_varnames_map_file.txt.

## The ICON model contains different map files, sometimes also called dictionaries. These plain text files translate between ICON internal variable names and GRIB2/NetCDF short names. There are several dictionaries that can be specified in the namelists of ICON. ana_varnames_map_file (namelist initicon_nml, string parameter) Used by the input module for first guess and analysis files. Left column: ICON internal name; Right column: input name, e.g. GRIB2 short name latbc_varnames_map_file (namelist limarea_nml, string parameter) Used by the module for lateral boundary conditions. Left column: ICON internal name; Right column: input name, e.g. GRIB2 short name extpar varnames map file (namelist extpar nml, string parameter) Used when reading the external parameter file extpar_filename. This dictionary is practically not used as external parameters are read directly using the NetCDF library. output nml dict (namelist io nml, string parameter) Used by the namelist output module. This dictionary allows to use GRIB2 short names instead of ICON internal names in the output namelists of ICON. Left column: namelist variable name; Right column: ICON internal name A reasonable choice is to use the same table as for reading the initial data with inverted columns. This can be achieved with the namelist parameter linvert_dict (io_nml). netcdf_dict (namelist io_nml, string parameter) The NetCDF dictionary can be used when writing to NetCDF files where variable short names can be chosen freely (in contrast to GRIB2). This allows to adopt ICON output variable names to already available postprocessing scripts.

### 5.1.2. Starting from Uninitialized DWD Analysis

This analysis product is rarely the optimal choice for model initialization, as it generates a significant amount of spurious noise during the first few hours of a model run (see Figure 2.8). Nevertheless, this mode is described for completeness. The process of obtaining the uninitialized DWD analysis for non-incremental update is described in Section 2.2.1.

Model initialization is basically controlled by the following four namelist parameters:

### init_mode = 1 (namelist initicon_nml, integer value)

To start from uninitialized DWD analysis data (without incremental analysis update), the initialization mode must be set to 1.

### lread_ana (namelist initicon_nml, logical value)

By default, this namelist parameter is set to .TRUE..

For lread_ana=.TRUE. the ICON model expects two input files per domain. One containing the ICON first guess (3 h forecast) fields, which served as background fields for the assimilation process. The other contains the analysis fields produced by the assimilation process. ICON reads the fields from the first guess file and replaces these subsequently with fields from the analysis file, where available. See Table 11.2 for a list of variables.

#### dwdfg_filename (namelist initicon_nml, string parameter)

Filename of the DWD first guess input file.

### dwdana_filename (namelist initicon_nml, string parameter)

Filename of the DWD analysis input file.

Remember to make sure that the validity date for the first guess and analysis input file is the same and matches the model start date given by ini_datetime_string.

Input filenames need to be specified unambiguously, of course. By default, if the user does not provide namelist settings for dwdfg_filename and dwdana_filename, the filenames have the form

```
dwdfg_filename = "dwdFG_R<nroot>B<jlev>_DOM<idom>.nc"
dwdana_filename = "dwdANA_R<nroot>B<jlev>_DOM<idom>.nc"
```

This means, e.g., that the first guess filename begins with "dwdFG_", supplemented by the grid spacing RxByy and the domain number DOMii. Filenames are treated case sensitively.¹



By changing the above setting, the user has full flexibility with respect to the filename structure. The following keywords are allowed for the namelist parameters dwdfg_filename, dwdana_filename and ifs2icon_filename (for the latter see Section 5.1.5):

<path></path>	model base directory
	(namelist parameter model_base_dir, namelist master_nml)
<nroot></nroot>	grid root division $\mathbf{R}\boldsymbol{x}$ (single digit)
<nroot0></nroot0>	grid root division $\mathbf{R}\boldsymbol{x}\boldsymbol{x}$ (two digits)
<jlev></jlev>	grid bisection level Byy (two digits)
<idom></idom>	domain number (two digits).

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

For lread_ana=.FALSE., only dwdfg_filename has to be specified. This can be used to run ICON from a first guess file. The combination of init_mode=1 and kwlread_ana=.FALSE., however, is rarely used. There is an overlap in possible applications with init_mode=7, which is usually the better choice due to the flexibility given by the vertical remapping.

### 5.1.3. Starting from Uninitialized DWD Analysis for IAU

As will be described in Section 11.3.1, IAU is a means to reduce the initial noise which typically results from small scale non-balanced modes in the analysis data set. Combining this analysis product with IAU is the preferred method in cases where the horizontal and vertical grid of the intended forecast run exactly match with that of the analysis. Since no horizontal interpolation is required, the forecast run can make use of the surface tile information which is specific to this analysis product. Moreover, this product exhibits the smallest noise level during model start.

The process of obtaining the uninitialized analysis for IAU is described in Section 2.2.1.

Model initialization is basically controlled by the following namelist parameters:

#### init_mode = 5 (namelist initicon_nml, integer value)

To start from DWD analysis data with IAU, the initialization mode must be set to 5.

ICON again expects two input files. One containing the ICON first guess, which typically consists of a 1.5 h forecast taken from the assimilation cycle (as opposed to a 3 h forecast used for the non-IAU case). The other file contains the analysis fields (mostly increments) produced by the assimilation process. See Table 11.1 for a full list of variables.

### dwdfg_filename (namelist initicon_nml, string parameter)

Filename(s) of the DWD first guess input file(s) for each domain. See Section 5.1.2 for an explanation of the filename structure.

#### dwdana_filename (namelist initicon_nml, string parameter)

Filename(s) of the DWD analysis input file(s) for each domain. See Section 5.1.2 for an explanation of the filename structure.

The behavior of the IAU procedure is controlled via the namelist switches dt_iau and dt_shift:

### dt_iau = 10800 (namelist initicon_nml, real value)

Time interval (in s) during which the IAU procedure (i.e. dribbling of analysis increments) is performed.

#### dt_shift = -5400 (namelist initicon_nml, real value)

Time (in s) by which the model start is shifted ahead of the nominal model start date given by ini_datetime_string. Typically dt_shift is set to  $-0.5 * dt_iau$  such that dribbling of the analysis increments is centered around ini_datetime_string.

As explained in Section 11.3.1 and depicted in Figure 5.2, 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.



Figure 5.2.: Schematic illustrating typical settings for a global ICON forecast run starting from a DWD analysis with IAU at 00 UTC. IAU (i.e. analysis filtering by dribbling of analysis increments) is performed over a 3h time interval (dt_iau), with the model start being shifted ahead of the nominal start date by 1.5h (dt_shift). The validity date of the first guess and analysis is 22:30 UTC and 00 UTC, respectively.

The secret of iterative IAU: Some of you might have heard about an ICON feature named *iterative IAU*, though still wondering what's behind it. The *iterative IAU* combines two model runs which serve two different purposes into a single model run. This is achieved by means of an ICON-internal loop structure. It has been implemented for sake of pure convenience.

The first model run is meant to generate a filtered (or initialized) analysis out of the uninitialized analysis for IAU product. To this end an IAU run is launched which, in contrast to the standard IAU run described above, uses a halved asymmetric IAU window of dt_iau=5400 (asymmetric w.r.t. to the validity time of the analysis increments). The shift of the model start remains unchanged (i.e. dt_shift=-5400). The model integration stops after 5400s (at the nominal start date) and the model state is written to disk. For the example in Figure 5.2 the stop date would be 00 UTC.

During the asymmetric IAU window, the analysis increments have been fully incorporated. The resulting model state is termed *filtered* or *initialized analysis*. It is equivalent to the *initialized analysis product* described in Section 2.2.1.

The second model run is a standard forecast run with a centered IAU window, starting from the uninitialized analysis for IAU product as described earlier in this Section.

The key point is that both runs are performed within a single model run by means of an internal loop structure. After ICON's read-in and initialization procedure, the first loop iteration stores the model's initial state, performs the asymmetric IAU run and saves the resulting initialized analysis to disk. During the second loop iteration, the model resets to the previously stored initial state and performs a standard forecast run with a centered IAU window.

The main benefit of merging these runs into a single model run is that the initial conditions (i.e. first guess and analysis file) have to be read only once, which saves a decent amount of time in the operational forecast cycle.

The iterative IAU is activated by setting iterative_iau=.TRUE. in the namelist initicon_nml. Given the namelist parameters dt_shift and dt_iau, they are applied in the following way during iteration 1 and 2:

iteration I:	0.5dt_iau	dt_shift
iteration II:	dt_iau	dt_shift

### 5.1.4. Starting from Initialized DWD Analysis

The initialized analysis is the product of choice in cases where the horizontal and/or vertical grid of the intended model run differs from that of the analysis. Using the uninitialized analysis for IAU is prohibited in such cases, as the horizontal interpolation of tiled surface fields makes no sense. Moreover, the initialized analysis product is less cumbersome to use, as it consists of a single file per domain, only. When compared to the standard uninitialized analysis product, spurious noise is significantly reduced (see Figure 2.8).

The process of obtaining the initialized analysis is described in Section 2.2.1.

Model initialization is basically controlled by the following namelist parameters:

### init_mode = 7 (namelist initicon_nml, integer value)

To start from initialized DWD analysis data, the initialization mode must be set to 7. If the number and or heights of the vertical levels differs between the model and the analysis, the input fields are automatically remapped in the vertical during read-in.

ICON expects a single input file. See Table 11.3 for a full list of variables.

### dwdfg_filename (namelist initicon_nml, string parameter)

Filename(s) of the initialized DWD analysis input file(s) for each domain. Admittedly, the nomenclature "dwdfg" is a bit counter intuitive, as the file contains the full analysis rather than the first guess. See Section 5.1.2 for an explanation of the filename structure.

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

### 5.1.5. Starting from IFS Analysis

No filtering procedure is currently available when starting off from interpolated IFS analysis data. The model just reads in the initial data from a single file and starts the forecast.

The process of obtaining the IFS analysis and its content is described in Section 2.2.2.

#### init_mode= 2 (namelist initicon_nml, integer value)

To start from interpolated IFS analysis data, the initialization mode must be set to 2. Note that for this initialization mode only input data in NetCDF format are supported and the specification of a dictionary file is not possible.

#### ifs2icon_filename (namelist initicon_nml, string parameter)

ICON expects a single file per domain from which interpolated IFS analysis can be read. With this parameter, the filename can be specified. Similar to the namelist parameters dwdfg_filename and dwdana_filename, which have been explained above in Section 5.1.2, the filenames have the form

ifs2icon_filename = "ifs2icon_R<nroot>B<jlev>_DOM<idom>.nc"

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

### 5.2. Starting or Terminating Nested Domains at Runtime

Starting or terminating nested domains at runtime is possible by means of the namelist parameters start_time and end_time in the namelist grid_nml. Model calculations for the nested domain are performed if the simulation time of the parent domain is greater or equal to start_time and less than end_time. The settings are graphically illustrated in Fig. 5.1.

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

Comma-separated list of integer values. For each domain, the start time relative to the experiment start date can be specified in seconds. A value of 0 for the *i*th domain means that it is started at experiment start date which is either defined by ini_datetime_string or experimentStartDate. If Incremental Analysis Update (IAU) is used, start_time must be set equal to dt_shift (initicon_nml) (i.e. negative), in order for the nested domain to be active from the very beginning.

#### end_time (namelist grid_nml, list of real values)

Comma-separated list of integer values. For each domain, the end time relative to the experiment start date can be specified in seconds. I.e. a value of 3600 specified for the *i*th domain means that it is terminated one hour after experiment start.

As discussed in Section 2.2, initial data files are usually required for each nested domain. With only little loss of forecast skill, this rather tedious procedure can be overcome by starting the nested domain(s) shortly after the global domain. In that case, nested domains are initialized by parent-to-child interpolation of the prognostic fields. Note, however, that surface tile information will be lost. Surface fields on the child domain are initialized with *aggregated* values interpolated from the parent domain.

# 6. Running ICON-LAM

The most important first: Running the limited area (regional) mode of ICON does not require a separate, fundamentally different executable. Instead, ICON-LAM is quite similar to the other model components discussed so far: It is easily enabled by a top-level namelist switch

Namelist grid_nml: l_limited_area = .TRUE.

Other namelist settings must be added, of course, to make a proper ICON-LAM setup. This chapter explains some of the details.

**Chapter Layout.** Some of the pre-processing aspects regarding the regional mode have already been discussed in Section 2.3. Based on these prerequisites this chapter explains how to actually set up and run limited area simulations.

In the following, technical details on the limited area mode are provided, in particular on how to control the read-in of initial data and boundary data.

### 6.1. Limited Area Mode vs. Nested Setups

In Section 3.9.1 the nesting capability of ICON has been explained. Technically, the same computational grids may be used either for the limited area mode or the nested mode of  $ICON^1$ . Furthermore, both ICON modes aim at simulations with finer grid spacing and smaller scales. They therefore choose a comparable set of options out of the portfolio of available physical parameterizations.

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

- ICON-LAM is driven by externally supplied boundary data which may come from a global model or a coarser resolution LAM that has been run in advance – that's an obvious difference! During the simulation, boundary conditions are updated at regular time intervals by reading input files. Between two lateral boundary data samples the boundary data is linearly interpolated.
- Lateral boundary updates happen (significantly) less frequently compared to oneway nesting.
- The driving model may be different from the limited area model and may run on different computer sites. Both models may even differ in terms of the governing equations as well as numerical methods used.

¹Here, we do not take the reduced radiation grid into account, see Section 3.10. This serves to simplify the discussion at this point.

- ICON-LAM allows for a more flexible choice of vertical levels: Nested domains may differ from the global, "driving" grid only in terms of the top level height, but vertical layers must match between the nested and the parent domain (see Section 3.9.1). In contrast to that, the limited area mode performs a vertical interpolation of its boundary data. This is the default namelist parameter setting itype_latbc=1 in the namelist limarea_nml. The level number and the level heights may therefore be chosen independently.
- ICON-LAM allows for a more flexible choice of the horizontal resolution. While for nested setups the increase in horizontal resolution per nesting level is constrained to a factor of 2, the resolution of the limited-area domain can be freely selected. However, resolution jumps much larger than a factor of ~ 5 between the forcing data resolution and the target resolution should be avoided, since it will negatively impact the forecast quality.

### 6.2. Nudging in the Boundary Region

In order to prevent outward-propagating waves from reflecting back into the domain, a sponge layer is implemented along the lateral boundaries. Within this sponge layer the interior flow is relaxed towards externally specified boundary data. In addition to this lateral boundary nudging, upper boundary nudging along the model top can be switched on by choosing nudge_type =1 (namelist nudging_nml). The default value is 0, i.e. it is switched off. Figure 6.1 schematically depicts the partitioning of the limited area domain into the *lateral boundary zone*, labeled 0, the adjacent *lateral nudging zone 1*, the *upper nudging zone 2*, the *nudging overlap zone 3* and the "free" model atmosphere zone 4. In the lateral boundary zone 0, which has a fixed width of 4 cell rows, externally supplied boundary data are prescribed.

The mathematical implementation of the sponge layer in the nudging zones follows the work by Davies (1976, 1983). An additional "forcing" term is added to the right hand side of the prognostic equations for  $v_n$ ,  $\theta_v$ ,  $\rho$ , and  $q_v$  and is applied at each fast physics time step  $\Delta t$ :

$$\psi(t) = \psi^*(t) + \alpha_{\text{nudge}} \underbrace{\left[\psi_{\text{bc}}(t) - \psi^*(t)\right]}_{=\delta\psi}, \qquad (6.1)$$

where  $\psi_{bc}$  is the externally specified value of the prognostic variable  $\psi$  at time t, and  $\alpha_{nudge}$  is a dimensionless coefficient that controls the strength of the nudging. This coefficient gradually decreases with increasing distance from the boundary and is of the form

$$\alpha_{\text{nudge}} = \begin{cases} A_0 \exp\left(-\frac{|r-r_0|}{\mu}\right), & \text{if } r-r_0 \leq L \text{ (in region 1)} \\ B_0 \left(\frac{z-z_{\text{start}}}{z_{\text{top}}-z_{\text{start}}}\right)^2, & \text{in region 2} \\ \max\left\{A_0 \exp\left(\cdots\right), B_0 \left(\cdots\right)^2\right\}, & \text{in region 3} \\ 0, & \text{in region 4} \end{cases}$$
(6.2)

with  $A_0$  the maximum relaxation coefficient in the lateral nudging zone, L the width of the lateral nudging zone given in units of cell rows,  $\mu$  the e-folding width given in



Figure 6.1.: Schematic illustration of the lateral boundary zone (blue) and the lateral and upper boundary nudging zones (gray) in the limited-area mode.

units of cell rows, r the actual cell row index beginning with 1 in the outermost cell row of the boundary zone, and  $r_0$  the cell row index at which the nudging zone starts (typically grf_bdywidth_c+1, see Figure 6.1). The parameters L,  $\mu$ , and  $A_0$  can be specified via nudge_zone_width, nudge_efold_width, and nudge_max_coeff in the namelist interpol_nml. The nudge zone width should at least comprise 8 (better 10) cell rows in order to minimize boundary artifacts. For the variables  $v_n$  and  $q_v$  the parameter  $A_0$  is multiplied by the factor 0.5.

 $B_0$  is the maximum nudging coefficient in the upper boundary nudging zone between the model top at height  $z_{top}$  (sleve_nml: top_height) and the nudging start height  $z_{start}$  (nudging_nml: nudge_start_height). The value of  $B_0$  is controlled by max_nudge_coeff_vn (nudging_nml) for the horizontal wind  $v_n$ , and max_nudge_coeff_thermdyn for  $\theta_v$  and  $\rho$ .

Note that positive water vapor increments  $\delta \psi = \delta q_v > 0$  are cut to zero in supersaturated regions  $(q_c > 0)$  in the lateral boundary nudging zone, in order to avoid an undesirable positive feedback on the growth of the amount of cloud water. In addition, water vapor is not subject to nudging in the upper boundary nudging zone. If the nudging data from the driving model contain hydrostatic variables (i.e. hydrostatic pressure), it might be more consistent to formulate the nudging in terms of the basic hydrostatic variables: pressure and temperature. This option is controlled by the namelist switch nudge_hydro_pres (limarea_nml), which applies to both lateral and upper boundary nudging. If set to .TRUE. (default), nudging increments of the hydrostatic pressure and the temperature,  $\delta p$  and  $\delta T$ ,

are computed and transformed into virtual potential temperature and density increments following the linear mapping

$$\delta \rho = X_{\rho} \delta p + Y_{\rho} \delta T + Z_{\rho} \delta q_{v}$$
$$\delta \theta_{v} = X_{\theta_{v}} \delta p + Y_{\theta_{v}} \delta T + Z_{\theta_{v}} \delta q_{v}$$

which is motivated by the total differential of the thermodynamic state equations. The factors X, Y and Z are determined by the state before the nudging  $(\psi^*)$ . Note again that water vapor increments are nonzero only in the lateral boundary nudging zone.

The nudging in ICON should not be confused with a Newtonian relaxation approach in the proper sense

$$\psi(t) = \psi^*(t) + \frac{\delta\psi}{\tau_{\text{relax}}} \Delta t \,. \tag{6.3}$$

If we identify Eq. (6.1) with Eq. (6.3), we find for the relaxation time

$$\tau_{\rm relax} = \frac{\Delta t}{\alpha_{\rm nudge}} \,. \tag{6.4}$$

It is proportional to the time step  $\Delta t$  ( $\rightarrow dtime$ ) and, hence, inversely proportional to the horizontal mesh size, i.e.  $\tau_{relax}$  decreases with increasing horizontal resolution. The nudging coefficient  $\alpha_{nudge}$  is a non-dimensional parameter and should not be considered as an inverse relaxation time. Given a proper relaxation time  $\tau_{relax} = \text{const.}$ , Eq. (6.3) converges to a differential equation in the limit  $\Delta t \rightarrow 0$ . Equation (6.1) would, however, diverge in this limit, so no corresponding differential equation exists. This fact should not be considered a problem, as the nudging describes no physical process anyway. If desired, you may of course emulate a relaxation approach. Choose the desired relaxation time and compute the corresponding nudging coefficients  $A_0$  or  $B_0$  according to Eqs. (6.4) and (6.2). You may want to implement this as a function in your runscript, in order to avoid (easily forgotten) recomputations by hand whenever you change the time step dtime.

For the sake of completeness, we mention the global nudging option nudge_type=2. In fact, it is intended for global simulations (1_limited_area=.FALSE.), but it may be used in the limited area mode as well. For the most part, it makes use of the same infrastructure as the upper boundary nudging. Taking a look at Figure 6.1, global nudging under the limited area mode means, in practical terms, the absence of the lateral and overlap nudging zones 1 and 3, and of the free model atmosphere zone 4. Instead, region 2, now termed global nudging zone, covers the entire domain interior. In addition to the vertical nudging shape factor  $[(z - z_{start})/(z_{top} - z_{start})]^2$  (see Eq. (6.2), region 2), global nudging offers a small selection of shape factors that enable a more uniform nudging throughout the vertical air column of the model². Please note that this nudging option is not in operational use, still mostly experimental and comes with no warranty. As its control parameters in nudging_nml are described in detail in the namelist documentation icon/doc/Namelist_overview.pdf, they will not be discussed further here. Just a few remarks: Upper boundary and global nudging share most namelist parameters. Where global nudging assumes different default values, this is marked by ()_{glbndg}. Where parameters apply to global nudging only, this is

²You can find schematic representations of the available vertical nudging shape factors at the end of the source code file icon/src/configure_model/mo_nudging_config.f90.

indicated in the *Description* and *Scope* columns. In contrast to upper boundary nudging, global nudging applies to the primary domain only. Within nested domains, if present, there is no direct forcing by global nudging.

Important note: The treatment of the vertical velocity component w along the model top is independent of the upper boundary nudging: w and corresponding vertical mass fluxes are set to zero at the uppermost half level of the computational domain. Starting from the height specified by damp_height (nonhydrostatic_nml), the vertical velocity is damped towards zero following the method proposed by Klemp et al. (2008).

If upper boundary nudging is switched on (nudging_nml: nudge_type=1), "lateral boundary" data (the driving data for the nudging) have to be provided for the entire limited area domain rather than the lateral boundary region, only. This mode requires the namelist parameter latbc_boundary_grid (limarea_nml) defining the grid file on which the lateral boundary data are defined to be empty, i.e. latbc_boundary_grid=" ". The same applies to global nudging (nudging_nml: nudge_type=2). Driving data have to be present for the entire primary domain (either a limited area or a global domain).

As with upper boundary nudging, details on the driving data for global nudging must be specified in namelist limarea_nml.

Upper boundary nudging is **not** restricted to the primary limited area domain. In multi-domain simulations, it is possible to switch on upper boundary nudging for nested domains, by setting the corresponding entries in **nudge_type** (comma separated list) to 1. All domains are nudged towards the same driving data, i.e. nested domains are equally nudged towards the "lateral boundary" data of the primary limited area domain.

Running the model in regional mode is quite often accompanied by choosing a lower model top height compared to global simulations. In these cases, the neglected air mass above model top can have a noticeable impact regarding the attenuation of the incoming solar irradiance and can be the source of a small but noticeable amount of downward long-wave irradiance. To account for that in a rather ad hoc manner, an additional model layer above model top can be added by setting latm_above_top =.TRUE. (namelist nwp_phy_nml). It is used by the radiation scheme, only. The additional layer has a (hard-coded) thickness of 1.5 times the thickness of the uppermost model layer. Currently, temperature is linearly extrapolated, assuming a vertical gradient of  $-5 \,\mathrm{K \, km^{-1}}$ . For ozone, aerosols and cloud fields, a simple no-gradient condition is assumed. Despite this rather ad hoc solution, it is suggested to activate the additional layer. Please note that this option works only in combination with a reduced radiation grid.

### 6.3. Model Initialization

The necessary input data to perform a limited area run are basically identical to those required for a global run (i.e. horizontal grid(s), initial conditions, external parameter; see

Section 5.1), with the exception that lateral boundary data are required in addition in order to drive the model.

Technically it is possible to combine initial- and boundary data from different sources (e.g. one might take boundary data from IFS and initial data from ICON). In general, however, it is better to use boundary and initial data from the same source.

Dependent on the available initial data, the following initialization modes can be used in limited area mode:

### init_mode (namelist initicon_nml, integer value)

<pre>init_mode = 2</pre>	initialize from <b>IFS data</b> . This mode has already been described in Section 5.1.5 in the context of reading IFS analysis data.
<pre>init_mode = 3</pre>	initialize from <b>IFS atmospheric</b> and <b>ICON surface data</b> . This mode is of special interest for operational weather services who want to perform cold starts with IFS atmospheric data.
<pre>init_mode = 7</pre>	initialize from <b>ICON data</b> . This mode has already been described in Section 5.1.4 in the con- text of reading in DWD's initialized analysis product.

These modes have in common that the read-in process is followed by a vertical interpolation of the input fields to the target vertical grid. Thus the target vertical grid can be chosen independent of the vertical grid on which the input is defined. Note that in case of init_mode = 7 the vertical interpolation requires that the field HHL (vertical half level heights) is contained in the initial data.

### Specifics of init_mode=2

- Only input data in NetCDF format are supported.
- A single input file per domain is expected, containing the analysis (or, more generally, the initial state). The filename must be provided for ifs2icon_filename (see also Section 5.1.5).
- The required input fields are depicted in Figure 6.2.

#### Specifics of init_mode=7

- A single input file per domain is expected, containing the analysis (or, more generally, the initial state). The filename must be specified with the parameter dwdfg_filename(initicon_nml).
- As we do not make use of a second input file containing explicit analysis information, it is good practice to indicate this via the following namelist parameter.

Atmosphere  $\begin{cases}
U, V \\
or \\
VN
\end{cases}, W, T, LNPS, GEOP_ML, QV, QC, QI, QR, QS$ Soil/Surface SMIL1, SMIL2, SMIL3, SMIL4, STL1, STL2, STL3, STL4, LSM, CI, GEOP_SFC, ALB_SNOW, SST, SKT, T_SNOW, W_SNOW, RHO_SNOW, W_I

Figure 6.2.: Required variable set when initializing ICON-LAM from IFS data, i.e.
init_mode=2. Optional fields are marked in gray. Please note that although
the field is called W (vertical velocity) ICON expects the content of this field
to be OMEGA (vertical wind in a pressure based coordinate system) in case
of init_mode=2!

### lread_ana (namelist initicon_nml, logical value)

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

Note that in the recent ICON version lread_ana=.FALSE. is set automatically for init_mode= 7, if it has been forgotten by the user.

• The required input fields are listed in Table 11.3. A valid option is to use DWD's initialized analysis product for initialization. See Section 2.2.1 for ways to obtain it.

### Specifics of init_mode=3

- Two files are needed, dwdfg_filename containing the surface fields and ifs2icon_filename including the atmospheric fields (both in initicon_nml).
- For the atmospheric data the same procedures are used internally as for init_mode=2. As a consequence, the same set of atmospheric variables is required (see Figure 6.2) and the data needs to be in NetCDF format.
- Internally, the same procedures are used for the surface data as for init_mode=7. Hence, the same surface fields are required (see Table 11.3).
- Initially, this init_mode was designed to combine IFS atmospheric data with GME surface data. As there is no fundamental difference between the ICON and GME surface parameterizations, this init_mode can be used for both.

### 6.4. Reading Lateral Boundary Data

The read-in of lateral boundary data is fortunately less cumbersome than the read-in of initial data, as it is based on a decision tree. The user is no longer required to select a



Figure 6.3.: Read-in of lateral boundary data. Based on this decision tree, ICON investigates the data file contents and diagnoses additional fields.
The following fields are read additionally from file: velocity fields VN (or U, V) and mixing ratios QV, QC, QI (optional: QR, QS). The fields W and OMEGA are optional; if they are unavailable, the vertical wind is initialized with zero. For the input from a pressure based coordinate system (right branch), note that ICON expects the field OMEGA under the name W.

specific mode which (hopefully) fits the data at hand. Instead, ICON scans the boundary data file and, dependent on its content, ICON diagnoses additional fields so as to obtain the internally required set of variables. The decision tree is depicted in Figure 6.3. If the provided data set does not match any of the trees, an error is thrown. As a result, ICON can handle variable sets from hydrostatic models (e.g. IFS) as well as non-hydrostatic models (e.g. COSMO, ICON) without the assistance of the user.

As apparent from the decision tree, three different variable sets can be handled. See Figure 2.11 for its specific content.

*Important note:* Boundary data sets originating from a non-hydrostatic model with height based vertical coordinates (e.g. COSMO or ICON) must contain the field HHL (vertical half level heights). It is required by the vertical interpolation procedure. Note, however, that the field only needs to be present in the boundary data set whose validity date equals the model start date.

Troubleshooting: The usage of a decision tree as depicted in Figure 6.3 has consequences when searching for the cause of an error. For example, a wrong specification of HHL in latbc_varnames_map_file leads to the following error behavior: ICON does not find HHL, so it erroneously takes the right branch of the decision tree. ICON then looks for the geopotential GEOSP (or alternatively GEOP_ML). This is also not found which results in the error that the geopotential was not found.

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

The type of lateral boundary conditions is specified by

### itype_latbc (namelist limarea_nml, Integer value)

If set to 1 time-dependent boundary conditions are used. ICON then tries to read external data files at regular time intervals from a particular location specified by latbc_filename and latbc_path (see below).

If set to 0, time-constant lateral boundary conditions are used which are derived from the initial conditions.

Boundary data is read at regular time intervals. This is specified by the following namelist parameter:

### dtime_latbc (namelist limarea_nml, floating-point value)

Time difference in seconds between two consecutive boundary data sets. At intermediate times, boundary conditions are computed by linear interpolation in time.

### 6.4.1. Naming Scheme for Lateral Boundary Data

Naturally, the sequence of lateral boundary data files must satisfy a consistent naming scheme. It is a good idea to consider this convention already during the pre-processing steps (see Section 2.3).

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

By default, the filenames are expected to have the following form:

"prepiconR<nroot>B<jlev>_<y><m><d><h>.nc"

Here, several keywords are used which are further explained below. This naming scheme can be flexibly altered via the namelist parameter latbc_filename (namelist limarea_nml) using the available keywords. The absolute path to the boundary data can be specified with latbc_path (string parameter, limarea_nml).



By changing the above setting, the user has full flexibility with respect to the filename structure. The following keywords are allowed for the namelist parameter latbc_filename:

<nroot>

grid root division  $\mathbf{R}\boldsymbol{x}$  (single digit)

<nroot0></nroot0>	grid root division $\mathbf{R}\boldsymbol{x}\boldsymbol{x}$ (two digits)
<jlev></jlev>	grid bisection level $Byy$ (two digits)
<dom></dom>	domain number (two digits)
<i><y></y></i>	year (four digits)
<m></m>	month (two digits)
<d></d>	day in month (two digits)
<h>&gt;</h>	hour (UTC) (two digits)
<min></min>	minutes (UTC) (two digits)
<sec></sec>	seconds (UTC) (two digits)
<ddhhmmss></ddhhmmss>	elapsed days, hours, minutes and seconds since
	<pre>ini_datetime_string or experimentStartDate (each two digits)</pre>
<dddhh></dddhh>	elapsed days and hours since ini_datetime_string
	or experimentStartDate (three digits day, two digits hours).

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

ICON supports NetCDF and GRIB2 as input format for boundary fields. Field names in input files do not necessarily coincide with internal ICON field names. Hence, an additional input text file (*dictionary file*) can be provided. This two-column file translates between the ICON variable names and the corresponding DWD GRIB2 short names or NetCDF variable names.

Specifying a valid dictionary file is currently mandatory, if pre-fetching of boundary data is selected num_prefetch_proc=1 (see below).

### Boundary grid: latbc_boundary_grid (namelist limarea_nml, string)

As it has been explained in Section 2.3, the lateral boundary data can be defined on an auxiliary grid, which contains only the cells of the boundary zone for optimization purposes.

If this is the case for the applied boundary data, the filename of this grid file must be specified with this namelist parameter.

### 6.4.2. Pre-Fetching of Boundary Data (Mandatory)

*Pre-fetching* strives to avoid blocking of the computation due to reading of boundary data. The term denotes the reading of files ahead of time, i.e. the next input file will be processed simultaneously with the preceding compute steps. This avoids waiting for the I/O processes during the time consuming procedure of opening, reading and closing of the input files.

#### num_prefetch_proc = 1 (namelist parallel_nml, integer value)

If this namelist option is set to 1, one MPI process will run exclusively for asynchronously reading boundary data during the limited area run. This setting, i.e. the number of pre-fetching processors, can be zero or one.

Enabling the pre-fetching mode is **mandatory** for the described LAM setup.

## 6.5. Tropical Setup

The tropical setup namelist configuration of the COSMO model was and still is applied by many users of the COSMO model. As ICON is a global model, it is routinely applied to a wider range of conditions than a limited-area only model like the COSMO model. Nevertheless, when choosing an area close to the equator some namelist parameters have to be adapted compared to mid-latitude high-resolution limited-area setups. For some of these parameters, the necessary adaptions are quite straight forward. Other parameters require rigorous testing and tuning of the configuration in order to find a good combination for these parameters.

Parameter	Mid-Latitudes	<b>Tropical Setup</b>	Description
tune_zvz0i (nwp_tuning_nml)	0.85	test & tune	Terminal fall velocity of ice, meaningful range for tuning between 1. and 3.5
rat_sea (turbdiff_nml)	0.8	test & tune	Ratio of laminar scaling factors over sea and land, meaningful range for tuning between 1. and 20.
<pre>inwp_convection lshallowconv_only (nwp_phy_nml)</pre>	1 .false.	test & tune	Convection parameterization, inwp_convection=1 and lshallowconv_only=.true. for shallow convection only, inwp_convection=0 for no convection parameterization
top_height (sleve_nml)	22000.	30000.	Model top height (only for ivctype=2, see Section 3.4)
damp_height (nonhydrostatic_nml)	12250.	18000.	Height at which Rayleigh damping of vertical wind starts

**Table 6.1.:** List of namelist parameters that are sensitive to the choice of the limited-areadomain location. Values that are listed in the table are suitable for setupsaround 2.5 km effective grid spacing.

Table 6.1 provides an overview on namelist parameters that are sensitive to the choice of the domain location. For the terminal fall velocity of ice (tune_zvz0i) literature suggests higher values than used in current ICON setups (e.g., Heymsfield and Donner (1990) suggest a value which is higher by a factor of 3). Such literature values are derived under certain conditions (e.g., particle shape, temperature range). Hence, the value which is required by a model in order to get good results can differ. This value has a strong impact on the radiative properties of ice clouds and can be used to compensate biases. In summary, tune_zvz0i is a tuning parameter worth investigating if there are biases in radiation under cloudy conditions.

The namelist parameter **rat_sea** increases the thickness of the laminar sublayer over sea. Larger values mean larger laminar sublayers, i.e. less heat and moisture fluxes over sea.



Figure 6.4.: Schematic illustration of the choice of the namelist parameters damp_height and top_height for tropical and mid-latitude setups.

For example, tuning this parameter to lower values might be beneficial if there is too little ocean-atmosphere exchange in case of tropical cyclones. This parameter can only have an impact if a significant part of the model is covered by ocean.

At effective grid spacings of 5 km or less, deep convection is resolved explicitly. The namelist switch lshallowconv_only allows to turn off the deep convection parameterization but keeping the shallow convection parameterization active. However, especially for arid regions at effective grid spacings of <3 km it can be beneficial to turn off convection completely by setting inwp_convection=0. Otherwise the already sparsely available water vapor is lifted by the shallow convection parameterization from the boundary layer into the free atmosphere. This can lead to a too strong cloud formation inhibition in arid regions.

The probably most important change for tropical setups is depicted schematically in Figure 6.4. The model top height (top_height) and the height above which the Rayleigh damping of the vertical velocity becomes active (damp_height) have to be chosen such that convection is not inhibited by the Rayleigh damping. The tropical tropopause reaches to higher altitudes than the mid-latitude and polar tropopause. While it is sufficient for mid-latitude setups to choose the model top at 22 km and start the damping layer at about 12 km, the tropical tropopause is typically located at an altitude of 17 km. Hence, the relaxation of the vertical velocity would in this case inhibit deep convection. It can be easily avoided by extending the vertical extent of the model simulation to, for example, top_height=30000.0 and damp_height=18000.0. This requires a sufficient number of vertical levels (e.g., num_lev=65).

ES

Tuning documentation: Recent ICON model versions contain a document named *ICON model parameters suitable for model tuning* which is located in doc/tuning/icon_tuning_vars.pdf.

6. Limited Area Mode

# 7. Model Output

In this chapter we describe advanced settings for the namelist controlled model output. In particular, the ICON model offers several options for internal post-processing, such as the horizontal remapping of the prognostic output onto regularly spaced ("longitude-latitude") grids and vertical interpolation, for example on pressure levels. Another type of output products is ICON's checkpointing feature which allows to restart the execution from a pre-defined point using the data stored in a file.

### 7.1. Settings for the Model Output

Model output is enabled via the namelist run_nml with the main switch output. By setting this string parameter to the value "nml", the output files and the fields requested for output can be specified by special namelists¹. In the following, this procedure will be described in more detail.

In general the user has to specify five individual quantities to generate output of the model. These are:

- a) The time interval between two model outputs.
- b) The name of the output file.
- c) The name(s) of the variable(s) to output.
- d) The type of the vertical output grid, e.g., pressure levels or model levels.
- e) The type of the horizontal output grid, i.e. ICON grid or geographical coordinates.

All of these parameters are set in the namelist output_nml. Multiple instances of this namelist may be specified for a single model run, where each output_nml creates a separate output file. The options d) and e) require an interpolation step. They will be discussed in more detail in Section 7.1.1.

In the following, we give a short explanation for the most important namelist parameters:

### output_filename (namelist output_nml, string parameter)

This namelist parameter defines a prefix for the output filename (which may include the directory path). The domain number, level type, file number and file format extension will be appended to this prefix.

¹Another possibility is to set **output="none"**. This can be used to make scalability tests without being influenced by writing time.

### output_bounds (namelist output_nml, three floating-point values)

This namelist parameter defines the start time and the end time for the model output and the interval between two consecutive write events. The three values for this parameter are separated by commas and, by default, they are specified in seconds.

### ml_varlist (namelist output_nml, character string list)

This parameter is a comma-separated list of variables or variable groups (the latter are denoted by the prefix "group:"). The ml_varlist corresponds to model levels, but all 2D variables (for example surface variables) are specified in the ml_varlist as well. It is important to note that the variable names follow an ICON-internal nomenclature. The temperature field, for example, is denoted by the character string "temp". A list of available output fields is provided in Appendix B.

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, character string)

This character string specifies a list of model levels for which the variables and groups should be written to output. Level ordering does not matter.

Allowed is a comma- (or semicolon-) separated list of integers, and of integer ranges like "10...20". One may also use the keyword "nlev" to denote the maximum integer (or, equivalently, "n" or "N"). Furthermore, arithmetic expressions like "(nlev-2)" are possible.

Basic example: m_levels = "1,3,5...10,20...(nlev-2)"

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

Related to setups with nests, i.e. multiple domains: This namelist parameter sets the domains for which this namelist is used. If not specified (or specified as -1), this namelist will be used for all domains.

### remap (namelist output_nml, integer value: 0/1)

This namelist parameter is related to the horizontal interpolation of the output to regular grids, see Sections 7.1.1 and 7.1.2.

### filetype (namelist output_nml, integer value: 2/4)

ICON offers the possibility to produce output either in NetCDF or GRIB2 format. This can be chosen by the namelist parameter filetype of the namelist output_nml. Here, the value filetype=2 denotes the GRIB2 output, while the value filetype=4 denotes the NetCDF file format.



The namelist parameter output_filename provides only partial control over the resulting filename, namely its prefix. Complete control over the resulting filename can be achieved with the namelist parameter filename_format (namelist output_nml, character string). By default, filename_format is set to:

### "<output_filename>_DOM<physdom>_<levtype>_<jfile>"

The following keywords are allowed for filename_format (the list is incomplete and shows only the most important options):

model base directory
(namelist parameter model_base_dir, namelist master_nml)
domain number (two digits)
Level type (ML, PL, HL, IL)
ISO-8601 date-time stamp in format YYYY-MM-DDThh:mm:ss.sssZ
elapsed days, hours, minutes and seconds since
<pre>ini_datetime_string or experimentStartDate (each two digits)</pre>
ISO-8601 date-time stamp in format YYYYMMDDThhmmssZ
Consecutive file id.

As it has been stated before, each output_nml creates a separate output file. To be more precise, there are a couple of exceptions to this rule. First, multiple time steps can be stored in a single output file, but they may also be split up over a sequence of files (with a corresponding index in the filename), see the namelist parameter steps_per_file. Second, an instance of output_nml may also create more than one output file if grid nests have been enabled in the model run together with the global model grid, see the namelist parameter dom. In this case, each of the specified model domains is written to a separate output file. Finally, model output is often written on different vertical axes, e.g., on model levels and on pressure levels. The specification of this output then differs only in the settings for the vertical interpolation. Therefore it is often convenient to specify the vertical interpolation in the same output_nml as the model level output, which again leads to multiple output files.

### 7.1.1. Output on Regular Grids and Vertical Interpolation

Many diagnostic tools, e.g., to create contour maps and surface plots, require a regularly spaced distribution of the data points. Therefore, the ICON model has a built-in output module for the interpolation of model data from the triangular mesh onto a regular longitude-latitude grid. Further information on the interpolation methods can be found in the database documentation Reinert et al. (2020), see Section 0.3.

The relevant namelist parameters for the *horizontal* interpolation of the output fields are set in the namelist output_nml. As it was already mentioned in Section 7, multiple instances of this namelist may be specified for a single model run, where each output_nml creates a separate output file.

#### remap (namelist output_nml, integer value 0=triangular / 1=lon-lat)

Set this namelist parameter to the value 1 to enable horizontal interpolation onto a regular grid. This option needs to be defined combination with reg_lat_def / reg_lon_def.

### reg_lat_def / reg_lon_def (namelist output_nml)

Latitudes and longitudes for the regular grid points are each specified by three values:

start, increment, end value; given in degrees. Alternatively, the user may set the number of grid points instead of an increment.

Furthermore, the model output can be written on a different *vertical* axis, e.g., on pressure levels, height levels or isentropes. In the following we will describe how to specify these options in the namelist output_nml. The relevant namelist parameters for the vertical interpolation of the output fields are:

### hl_varlist / pl_varlist / il_varlist (character string lists)

Similar to the namelist parameter ml_varlist mentioned above, these parameters are comma-separated lists of variables or variable groups. While the hl_varlist sets the output for height levels, pl_varlist defines variables on pressure levels and il_varlist specifies output on isentropic levels.

### h_levels / p_levels / i_levels (floating point values, comma-sep.)

Comma separated list of height, pressure, and isentropic levels for which the variables and groups specified in the above mentioned variable lists should be output. Height levels must be given in m, pressure levels in Pa and isentropes in K. Level ordering does not matter.

ICON's interpolation on pressure levels is extrapolating into the topography. This has the simple reason that contour plots, e.g. for 850hPa, usually do not show missing values over regions like Antarctica. There is no option to change this behavior in ICON. If needed, this has to be accounted for in the post-processing.

### 7.1.2. Remarks on the Horizontal Interpolation

First of all, it should be noted that all explanations in this section also apply to the iconremap tool, which interpolates ICON data as a pre-processing step.

ICON supports several numerical methods for interpolating data horizontally from the native triangular grid onto a regular lat-lon grid (or, in the iconremap case, to the interpolation between different triangular grids):

- Radial basis functions (RBF)
- Barycentric interpolation
- Nearest-neighbor interpolation

The concrete interpolation procedure depends on the variable and its physical characteristics. It is prescribed for the output module as indicated in the output product tables of ICON's database description, see Reinert et al. (2020). For the iconremap tool the interpolation method is set explicitly by the user for each field.

First, a small number of output fields is treated with a *nearest-neighbor interpolation*. The nearest neighbor algorithm selects the value of the nearest point and does not consider the values of neighboring points at all, yielding a piecewise-constant interpolant.



Figure 7.1.: Left: Examples for over- and undershoots for an RBF-based interpolation. Right: RBF interpolant with cut-off applied.

*Barycentric interpolation* is a two-dimensional generalization of linear interpolation. This method uses just three near-neighbors to interpolate and avoids over- and undershoots, since extremal values are taken only in the data points. For a detailed treatment we refer to the literature, for example to Press et al. (2007), Section 21.7.1: "Two-Dimensional Interpolation on an Irregular Grid". This interpolation makes sense for fields where the values change in a roughly piecewise linear way.

Barycentric interpolation needs to be enabled with the following namelist setting (otherwise it is replaced by a fallback interpolation):

support_baryctr_intp = .FALSE. (namelist interpol_nml, logical value)

The *icondelaunay* tool: The DWD ICON Tools contain the *icondelaunay* binary, which processes existing ICON grid files. It appends a Delaunay triangulation of the cell circumcenters to the grid file. This auxiliary triangulation can be used then to speed up the interpolation process.

Note that this way of pre-processing the ICON grid files is mandatory for DWD's NEC SX-Aurora platform where the spherical Delaunay algorithm has not been vectorized.

Most of the output data on regular grids is processed using an *RBF-based interpolation method*. The algorithm approximates the input field with a linear combination of radial basis functions (RBF) located at the data sites, see, for example, Ruppert (2007). RBF interpolation typically produces over- and undershoots at position where the input field exhibits steep gradients. This behavior is illustrated in Fig. 7.1. Therefore, the internal interpolation algorithm performs a cut-off by default. Note that RBF-based interpolation is *not conservative*.

The shape parameter (or scale parameter) is an important parameter which affects the quality of the RBF interpolation. The core of the interpolation method are the radial basis functions which are for the ICON tools chosen to be of Gaussian type, i.e.

$$f(x) := e^{-\left(\frac{x}{a}\right)^2}$$
,  $a > 0$ ,

with shape parameter a.

When we choose a smaller value for *a* then the RBF basis functions approach the Dirac delta function, which yields an almost-nearest-neighbor interpolation. Larger values for *a* generally reduce the interpolation error, but there exists a (grid specific) bound where the Cholesky decomposition of certain dense matrices fails, that are necessary for the RBF weight computation (see the info box below). The ICON model (as well as the ICON tools) provide a heuristic which estimates a proper RBF shape parameter for which the Cholesky decomposition succeeds in floating-point arithmetic. This estimation method is applied when the user does not provide a specific value via the namelist. For the latter case, see ICON's namelist documentation for the namelist parameter rbf_scale (output_nml).

The estimated value is reported in the log output of the ICON model:

```
mo_intp_lonlat::rbf_setup_interpol_lonlat_grid:
    auto-estimated shape_param = 1.1730307423896675E-002
```

A similar output for iconremap is available when applying the command-line option -vv.

*Cholesky Decomposition Fails:* The Cholesky decomposition of the RBF interpolation weight computation may fail with an error message of the following kind:

# mo_math_utilities:choldec: error in matrix inversion, nearly singular matrix mo_remap_rbf_errana::rbf_error: Cholesky decomposition failed!

This may happen for example when a bad value for the shape parameter has been chosen manually. However, the *automatic* shape parameter estimation may fail as well: This algorithm estimates largest-as-possible shape parameter by extrapolation from a number of sample (test) decompositions. When it fails to compute these samples, even the automatic estimator may abort with the above error message,

In these cases, please adjust the shape parameter manually (which may require several trial-and-error steps).

### 7.1.3. Interpolation onto Rotated Lat-Lon Grids

Users of the COSMO model are familiar with *rotated lat-lon grids*: Here, the computational spherical coordinate system is rotated in such a way that a pole problem is avoided and minimal convergence of the meridians is achieved. To some extent, this output can be reproduced by the ICON model:

### north_pole (namelist output_nml)

Definition of the north pole for rotated lat-lon grids ([longitude, latitude]). The default is north_pole = 0,90.

Note, however, that the "COSMO output" is in detail not quite what the COSMO user expects, especially with regard to wind speeds U, V: First, the basis vectors, i.e. the meridional and longitudinal directions, are not rotated. Second, in COSMO these speeds are defined on horizontally and vertically shifted grids ("staggered grids"). This is not the case with ICON and especially difficult to detect in data sets.



Figure 7.2.: Schematic illustrating the distribution of output load onto three output processes, using num_io_procs=3 and stream_partitions_ml=3.

**Regular grids over the poles.** If one wishes to obtain "circular" lat-lon grids near the poles, that is, e.g., grids that cover everything north of a chosen latitude circle, then the unrotated definition of the area in the output_nml can be used, provided it includes the pole.

Namelist example:

reg_lat_def = 40,0.5,90
reg_lon_def = 0,0.5,360

For rotated lat-lon grids – which then turn out rectangular in the satellite projection – one additionally uses the namelist parameter **north_pole** (output_nml), which defines the coordinate north pole of a rotated ( $\lambda$ ,  $\varphi$ ) system (longitude, latitude). The rotation operation is quite well described in the documentation of the COSMO model from which it was adopted for ICON (Section 2.3 in Doms and Baldauf (2018); Section 4.1 and Appendix A in Doms et al. (2003)).

Namelist example for ICON's output_nml:

reg_lon_def	= -10.,0.5,10.0
reg_lat_def	= -15.,0.5,15.0
north_pole	= 0.,0.

### 7.1.4. Output Rank Assignment

When a large number of different output files is written during the simulation, the task of formatting and writing may put an excessive load on the output processes. The number of output processes which share this output load can be increased by setting the



Figure 7.3.: Specifying experiment restart; compare this illustration to Fig. 5.1. The namelist parameters are explained in Section 7.2. Here we prefer the ISO8601 date-time specification (e.g. checkpointTimeIntVal) over the over the older settings (e.g. dt_checkpoint).

num_io_procs (parallel_nml) namelist parameter, see Section 8.2. If there exist multiple groups (e.g. output files, different variable sets, output intervals, interpolation grids) then these so-called *streams* will be distributed automatically over the available output processes.

### stream_partitions_ml (namelist output_nml, integer)

It may be even useful to spread the files of a *single* stream over multiple output processes. For example, when each output file is relatively large, then the subsequent file of this stream can be written by a different output process in order to diminish the risk of congestion. Please use the namelist parameter **stream_partitions_ml** to set the number of output processes among which the output files should be divided.

The distribution of output load using stream_partitions_ml is illustrated in Fig. 7.2.

#### pe_placement_ml (namelist output_nml, integer array)

This array is related to the namelist parameters num_io_procs and stream_partitions_ml and allows for an even more fine-tuned distribution of the output workload. At most stream_partitions_ml different ranks can be specified, ranging between 0 ... (num_io_procs - 1). This explicitly assigns the output streams to specific PEs and facilitates a load balancing with respect to small and large output files.

### 7.2. Checkpointing and Restart

There are many reasons why a simulation execution may be interrupted prematurely or unexpectedly. The checkpoint/restart option can save you from having to start the ICON
model over from the beginning if it does not finish as expected. It allows you to restart the execution from a pre-defined point using the data stored in a checkpoint file.

**Activating the restart.** The checkpoint/restart functionality is controlled by the following namelist parameters, which are also illustrated in Fig. 7.3.

#### dt_checkpoint (namelist io_nml, floating-point value)

This parameter specifies the time interval for *writing* restart files. The restart files are written in NetCDF format, and their names are specified by the namelist parameter restart_filename, see below.

Note that if the value of dt_checkpoint resulting from the model default or user's specification is larger than dt_restart (see below), then it will be automatically reset to dt_restart, s.t. at least one restart file is generated during the restart cycle.

Similar to the namelist parameters described in 5.1.1, which specify the model start and end dates, there exist character string replacements for dt_checkpoint and dt_restart:

- restartTimeIntVal (namelist master_time_control_nml, ISO8601, character string)
- checkpointTimeIntVal (namelist master_time_control_nml, ISO8601, character string)

#### lrestart (namelist master_nml, logical value)

If this namelist parameter is set to .TRUE. then the current experiment is resumed from a restart file.

Instead of searching for a specific data filename, the model reads its restart data always from a file with name restart_atm_DOMO1.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 (see Section 8) also offers the option to reserve a dedicated MPI task for writing checkpoint files. This feature can be enabled by setting the parameter num_restart_procs in the namelist parallel_nml to an integer value larger than 0. **Restart modes.** Different restart write modes are available, which allow for a distributed writing and read-in of restart files, depending on the parallel setup. These different restart modes are controlled via the namelist parameter restart_write_mode (io_nml).

Allowed settings for restart_write_mode (character strings!) are:

#### "joint procs multifile"

All worker processes write restart files to a dedicated directory. Therefore, the directory itself represents the restart data set. The information is stored in a way that it can be read back into the model independent from the processor count and the domain decomposition.

*Read-in:* All worker processes read the data in parallel.

#### "dedicated procs multifile"

In this case, all the restart data is first transferred to memory buffers in dedicated restart writer processes. After that, the work processes carry on with their work immediately, while the restart writers perform the actual restart writing asynchronously. Restart processes can parallelize over patches and horizontal indices.

*Read-in:* All worker processes are available to read the data in parallel (though this is usually limited by the number of restart files).

```
"sync"
```

'Old' synchronous mode. Process#0 reads and writes restart files. All other processes have to wait.

"async"

'Old' mode for asynchronous restart writing: Dedicated processes  $(num_restart_proc > 0)$  write restart files while the simulation continues. Restart processes can only parallelize over different patches.

*Read-in:* Processes # 0 reads while other processes have to wait.

Fallback mode.

If num_restart_proc (parallel_nml) is set to 0, then this behaves like "sync", otherwise like "async".

## 7.3. Meteogram Output

The ICON model also features a special output product called *meteograms*, containing the model variables with respect to time for a particular location, i.e. at single grid points.

ICON's built-in meteograms are intended for *non-operational use*. They should be seen as a by-product of the usual data output, rather for the purpose of error detection during development. Meteogram data files are written in the NetCDF data format, where an example of the (non-standard) file structure is given below. The output is enabled via the namelist setting output = 'nml' (namelist io_nml) in combination with a special namelist, meteogram_output_nml:

In addition to the namelist parameters in the example above, the following settings are worth mentioning:

- zprefix (namelist meteogram_output_nml, character string)
   string with file name prefix for output file

During the model simulation, one of the asynchronously running output processes (see Section 8) collects the meteogram buffers from the compute processes and writes the data to a file. Meteograms do not use ICON's variable list infrastructure (see Section 9.3). However, the output can be easily extended to sample of additional model variables. To this end, see the extensive comments in the source code, src/io/atmo/mo_mtgrm_output.f90.

For basic textual output, there exists an auxiliary NCL script

#### scripts/postprocessing/tools/mtgrm_cosmo.ncl

For an introduction to the NCAR Command Language NCL see Section 10.3.3. This script can be applied to a data file with the following command:

```
ncl -n mtgrm_cosmo.ncl DataFileName='"METEOGRAM.nc"' itime=0;
```

The same directory also contains scripts for plotting meteogram data with NCL.

**File format description.** As mentioned before, the NetCDF meteogram output has a non-standard file structure. We list the most important file entries in the following:

meteogram station info:

station_name, station_lon, station_lat, station_hsurf,

and  $\texttt{station_idx}, \texttt{station_blk}:$  global triangle adjacent to meteogram station

sample date info:

date (sample dates) and time_step (plain time step indices).

info and value buffer for surface (2D) variables, 1,...,nsfcvars: sfcvar_name, sfcvar_long_name, sfcvar_unit, sfcvalues(time, nsfcvars, nstations): value buffer

info and value buffer for 3D variables 1,...,nvars: var_name, var_long_name, var_unit, heights(max_nlevs, nvars, nstations): level heights, var_levels: plain level indices, values: value buffer

# 8. Parallelization and Performance Aspects

This chapter gives an overview of the different mechanisms for parallel execution of the ICON model. These settings become important for performance scalability when increasing the model resolution and core counts.

## 8.1. Modes of Parallel Execution

The ICON model supports different modes of parallel execution:

- In the first place, ICON has been implemented for distributed memory parallel computers using the Message Passing Interface (MPI). MPI is a library specification, proposed as a standard by a broadly based committee of vendors, implementors, and users, see http://www.mcs.anl.gov/research/projects/mpi. Multiple ICON processes (processing elements, PEs) are started simultaneously and communicate by passing messages over the network. Each process is assigned a part of the grid to processe.
- Moreover, on multi-core platforms, the ICON model can run in parallel using sharedmemory parallelism with OpenMP. The OpenMP API is a portable, scalable technique that gives shared-memory parallel programmers a simple and flexible interface for developing parallel applications on platforms ranging from embedded systems and accelerator devices to multi-core systems and shared-memory systems, see http://openmp.org. An implementation of OpenMP ships with your Fortran compiler. OpenMP-parallel execution therefore does not require the installation of additional libraries.

To be precise, "OpenMP" refers to shared-memory parallelization within the host processor. The OpenMP target offloading for accelerators is not supported in ICON. Instead, ICON uses OpenACC for accelerator support, see below.

• As a third option, ICON can make use of graphics processing units (GPUs) to accelerate calculations. Compared to standard x86 CPUs, GPUs typically have many times the number of cores but require a sophisticated host-device memory management when computational problems are offloaded. ICON-Atmosphere (ICON-A) has been gradually ported to GPUs using OpenACC directives, see https://www.openacc.org (see Section 8.5).

These mechanisms are not mutually exclusive. A *hybrid* approach is also possible: Multiple ICON processes are started, each of which starts multiple threads. The processes communicate using MPI. The threads communicate using either OpenMP or OpenACC. Finally, note that although ICON has been implemented for distributed memory parallel computers using the *Message Passing Interface* (MPI), the model can also be installed on sequential computers, where MPI and/or OpenMP are not available. Of course, this execution mode limits the model to rather small problem sizes.

## 8.2. Settings for Parallel Execution

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

Worker PEs	this is the majority of MPI tasks, doing the actual work
I/O PEs	dedicated output server tasks ¹
Restart PEs	for asynchronous restart writing (see Section $7.2$ )
Prefetch PE	for asynchronous read-in of boundary data in limited area mode
	(see Section $6.4.2$ )
Test PE	MPI task for verification of MPI parallelization (debug option)

The configuration settings are defined in the namelist parallel_nml. To specify the number of output processes, set the namelist parameter num_io_procs to a value larger than 0, which reserves a number of processors for output. While writing, the remaining processors continuously carry out calculations. Conversely, setting this option to 0 forces the worker PEs to wait until output is finished. For the writing of the restart checkpoints (see Section 7.2), there exists a corresponding namelist parameter num_restart_procs.

During start-up, the model prints out a summary of the processor partitioning. This is often helpful to identify performance bottlenecks. First of all, the model log output contains a one-line status message:

Number of procs for

test: xxx, work: xxx, I/O: xxx, Restart: xxx, Prefetching: xxx

Afterwards, the sizes of grid partitions for each MPI process are summarized as follows:

Number of compute PEs used for this grid: 118 # prognostic cells: max/min/avg xxx xxx xxx

Given the case that the partitioning process would fail, these (and the subsequently printed) values would be grossly out of balance.

#### Batch queuing system

Apart from the namelist settings, the user has to specify the computational resources that are requested from the compute cluster. In addition to the number of MPI tasks and OpenMP threads, here the user has to set the number of cluster-connected *nodes*.

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

Increasing the number of nodes allows to use more computational resources, since a single compute node comprises only a limited number of PEs and OpenMP threads. On the other hand, off-node communication is usually more expensive in terms of runtime performance.

The computer platform at DWD, the NEC SX-Aurora, uses the batch system PBS to control the requested resources. When using the qsub command to submit a script file, the batch system PBS allows for specification of options at the beginning of the file prefaced by the **#PBS** delimiter followed by PBS commands.

Finally the user has to set the correct options for the application launcher, which is the **mpirun** command on the NEC SX-Aurora platform. The Appendix A contains a description of the most important settings.

# 8.3. Best Practice for Parallel Setups

## 8.3.1. MPI Tasks and OpenMP Threads

ICON employs both distributed memory parallelization and shared memory parallelization, i.e. a "hybrid parallelization". Only the former type actually performs a decomposition of the domain data, using the de-facto standard MPI. The shared memory parallelization, on the other hand, uses OpenMP directives in the source code. In fact, nearly all DO loops that iterate over grid cells are preceded by OpenMP directives. For reasons of cache efficiency the DO loops over grid cells, edges, and vertices are organized in two nested loops: "jb loops" and "jc loops"¹ Here the outer loop ("jb") is parallelized with OpenMP.

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

In all of the other cases, the parallelization setup depends on the hardware platform and on the simulation size. In practice, 4 threads/MPI task have proven to be a good choice on x86-based systems. This should be combined with the *hyper-threading* feature, i.e. a feature of the x86 architecture where one physical core behaves like two virtual cores.

Starting from this number of threads per task the total number of MPI tasks is then chosen such that each node is used to an equal extent and the desired time-to-solution is attained – in operational runs at DWD this is  $\sim 1$  h. In general one should take care of the fact that the number of OpenMP threads evenly divides the number of cores per CPU socket, otherwise inter-socket communication might impede the performance.

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

¹This implementation method is known as *loop tiling*, see also Section 9.3.

## 8.3.2. Blocking (nproma)

ICON runs on x86 computer architectures, as well as on classical vector processors, and graphics accelerators. Depending on the given architecture, the namelist parameter **nproma** must be adapted accordingly. The abbreviation **nproma** (probably) stands for *n*ombre de *pro*fondeurs *m*aximal (maximum of maximal depths).

For x86 architectures we suggest nproma=16, such that the line index fits into the cache. In contrast to this, on GPUs the nproma parameter should equal the number of grid points handled by each MPI process, see Section 8.5.2. Finally, for NEC's SX vector architecture, a much larger value nproma=752, say, is suitable to match the very wide vector length.

There is another important namelist option which is enabled for the NEC vector machine. On the NEC system the parameter proc0_shift (namelist parallel_nml, INTEGER value) is set to "1" which means that the first MPI rank (worker) does not take part in the actual computation – it gets 0 cells in the domain decomposition process. The reason behind this setting is the so-called "hybrid execution" on the NEC: The first MPI rank runs on an x86 CPU (vector host) and executes the read-in procedures only while the remaining processes run on a vector machine, doing the computational work. On x86 architectures or GPU accelerators the proc0_shift setting is unnecessary.

## 8.3.3. Mixed Single/ Double Precision in ICON

To speed up code parts strongly limited by memory bandwidth, an option exists to use single precision for variables that are presumed to be insensitive to computational accuracy – primarily the dynamical core and the tracer advection.

This affects most local arrays in the dynamical core routines, some local arrays in the tracer transport routines, the metrics coefficients, arrays used for storing tendencies or differenced fields (gradients, divergence etc.), reference atmosphere fields, and interpolation coefficients. Prognostic variables and intermediate variables affecting the accuracy of mass conservation are still treated in double precision. To activate the mixed-precision option, run the configure script with the -enable-mixed-precision flag.

## 8.3.4. Bit-Reproducibility

Bit-reproducibility refers to the feature that running the same binary multiple times should ideally result in bit-wise identical results. Depending on the compiler and the compiler flags used this is not always true if the number of MPI tasks and/or OpenMP threads is changed in between. Usually compilers provide options for creating a binary that offers bitreproducibility, however this is often payed dearly by strong performance losses. With the NEC SX-Aurora compiler, it is however possible to generate an ICON binary offering bitreproducibility with only little performance loss. The ICON binary used in this workshop gives bit-reproducible results.

Bit-reproducibility is generally an indispensable feature for debugging. It is helpful

- for checking the MPI/OpenMP parallelization of the code. If the ICON code does not give bit-identical results when running the same configuration multiple times, this is a strong hint for an OpenMP race condition. If the results change only when changing the processor configuration, this is a hint for an MPI parallelization bug.
- for checking the correctness of new code that is supposed not to change the results.

# 8.4. Basic Performance Measurement

The ICON code contains internal routines for performance logging for different parts (setup, physics, dynamics, I/O) of the code. These may help to identify performance bottlenecks. ICON performance logging provides timers via the two namelist parameters ltimer and timers_level (namelist run_nml).



*Note for advanced users:* The built-in timer output is rather non-intrusive. It is therefore advisable to have it enabled also in operational runs.

With the following settings in the namelist run_nml,

ltimer	=	.TRUE.
timers_level	=	10

the user gets a sufficiently detailed output of wall clock measurements for different parts of the code:

name	# calls	total min (s)	total max (s)
total	237	 903.085	903.089
L integrate_nh	170640	 884.128	892.143
L nh_solve	5972400	 401.055	428.694
L nh_solve.veltend	7166880	 36.469	51.376
physics	49678	 103.107	104.759
L nwp_radiation	10030	 40.402	42.985
L radiation	220674	 31.845	34.963
 model_init	711	 59.875	59.876

Note that some of the internal performance timers are nested, e.g. the timer log for radiation is contained in physics, indicated by the "L" symbol. For correct interpretation of the timing output and computation of partial sums one has to take this hierarchy into account.

• The column "total max (s)" contains the maximum timing in seconds (maximum over all MPI tasks, OpenMP master thread).

- The row "model_init" contains the measurements for the model setup (allocation, read-in, etc.).
- The row "total" contains the model run-time, *excluding* the initialization and finalization phase.

## 8.5. ICON on Accelerator Devices (GPUs)

Section author

M. Jacob, DWD Numerical Models Division

ICON supports massively parallel accelerator devices such as GPUs (Graphics Processing Units). These devices use a different parallelization model than OpenMP and MPI, as described in Section 8.1. The following section outlines the deployed technique and how to configure ICON for compilation and runtime.

ICON-Atmosphere (ICON-A) has been gradually ported to GPUs using OpenACC, with almost all components used for limited area and global weather prediction now supported (as of January 2023). In particular, the dynamical core and NWP physics package have been ported in a joint effort by MeteoSwiss, CSCS and DWD, with technical support by NVIDIA. In addition to the NWP application, ICON-A is also ready for climate-oriented studies on GPUs such as conducted by Giorgetta et al. (2022). Details about the porting strategy can be found in that publication as well.

Some components of ICON-A may have limited GPU support for certain variants and options. Unsupported model configurations will result in error warnings when executed with OpenACC. Supported and tested configurations can be found in the example namelists in the run directory, under experiments exp.mch_* and exp.dwd_run_ICON_09_R2B4N5_EPS*. It should be noted that the OpenACC version of ICON is not yet operational, and that these experiments only run for short periods, so software bugs may still be discovered after extended use. Therefore, it is advisable to consider the OpenACC version of ICON experimental, but worth trying, if a GPU machine is available.

OpenACC is an API for offloading programs written in C, C++, and Fortran from a host CPU to an attached accelerator device (see https://www.openacc.org). It uses compiler directives (pragmas) and is similar to classical OpenMP, which handles shared-memory parallelization within the host processor (OpenMP target offloading for accelerators is not supported in ICON).

OpenACC is used to parallelize the innermost loops with unit stride memory rather than the blocks, which are parallelized with OpenMP. Current GPUs used for HPC have dedicated device memory. For example, a NVIDIA A100 GPU has up to 80 GB memory. This means that OpenACC must manage this dedicated accelerator memory as well.

OpenACC code has been added to ICON to specify both parallel execution on accelerators and memory management. The host CPU is always responsible for managing the accelerator. Each CPU process of ICON can only utilize one accelerator, but OpenACC acceleration and MPI process parallelism can be combined to use multiple accelerators. Direct MPI communication between GPU memory is possible when ICON is compiled with the __USE_G2G macro. OpenMP and OpenACC cannot be combined in ICON, although technically a program could use OpenMP for CPU thread parallelism and OpenACC for acceleration. OpenACC is not to be confused with CUDA, the latter being a proprietary API developed by NVIDIA for direct programming of NVIDIA GPUs.

## 8.5.1. Configuring and Compiling ICON-OpenACC

To compile ICON for use with an accelerator, a compiler with OpenACC support is necessary.

Currently, the only known compiler that supports ICON-OpenACC is *nvfort*, formerly known as the PGI compiler, which is part of the NVIDIA HPC SDK (https://developer.nvidia.com/hpc-sdk). This means that a NVIDIA GPU is required to use ICON-OpenACC. Efforts are underway to expand the number of supported hardware platforms and compilers. The minimum tested version of the NVIDIA HPC SDK is 21.2, but it is recommended to use a more recent version such as 22.5.

The ICON repository contains configuration wrappers for different machines, including:

- config/cscs/balfrin.gpu.nvidia for *Balfrin* at CSCS
- config/cscs/daint.gpu.nvidia for Piz Daint at CSCS
- config/dwd/linuxWS.gpu.pgi for Linux workstations
- config/jsc/juwels.gpu.ompi_nvhpc-22.7 and config/jsc/juwels.gpu.psmpi_nvhpc-22.7 for *JUWELS* at JCE.
- A wrapper for Levante's GPU partition (at DKRZ) will be added soon.

To compile ICON on a Linux workstation, create a build directory and configure and compile ICON as described in Section 1.2 using the following commands. The paths in the wrapper are prepared for DWD systems and the network mount /uwork1 is required to access the NVIDIA SDK.

```
mkdir build_linuxWS.gpu.pgi
cd build_linuxWS.gpu.pgi
../config/dwd/linuxWS.gpu.pgi --disable-rte-rrtmgp
make -j 6
```

8. Parallelization & I/O

The configuration on emphPiz Daint uses the package management tool *Spack* (see https://spack.readthedocs.io). Use the following commands to build ICON on *Piz Daint*. Set config_dir to the path of your copy of ICON. Alternatively, one can follow the commands in the script config/buildbot/daint_gpu_nvidia to use ones own instance of *Spack*.

## 8.5.2. Special Namelist Options for ICON-OpenACC

Some namelist settings in ICON-OpenACC must be adjusted for optimal performance.

The most critical setting is nproma (namelist parallel_nml), which governs the length of the "jc loops". These loops are parallelized with OpenACC, so they should be longer than the number of parallel computing units on the accelerator, e.g. 3456 double precision cores for a NVIDIA A100 GPU. The larger the nproma, the faster the computation, but if it is too large, memory will be wasted and performance will be reduced. Having less grid cells than computing units is inefficient and will increase MPI communication.

For a nested single-domain experiment, **nproma** should equal the number of grid points handled by each MPI process. The goal is one execution of a very long jc loop, i.e. having only one jb block loop over all cells (see also Section 9.3).

However, instead of setting nproma manually, the number of cell blocks can be set directly using nblocks_c = 1 (namelist parallel_nml, INTEGER value). Note that setting nblocks_c > 1 overwrites nproma.

For a nested multi-domain experiment, predetermining an optimal nproma is not possible due to different numbers of grid points per domain handled by each process. In this case, nproma must be set manually, and nblocks_c cannot be used. nproma should be set to a value greater than the number of grid points per process in the smallest domain. As an optimization starting point, set nproma =20000 (with default nblocks_c =0), then conduct a series of short benchmark experiments to find the optimal nproma value.

Another setting, nproma_sub (namelist parallel_nml, INTEGER value), governs the chunk size for sub-blocks used in the ecRad and RRTMGP radiation codes. The sub-blocks help to reduce the memory footprint of the radiation. For single-domain experiments, nblocks_sub (namelist parallel_nml, INTEGER value) can be used instead of nproma_sub. A recommended starting value for nblocks_sub is 6, however, it is advisable to experiment with smaller or larger values to determine the optimal setting based on memory availability. In contrast, for nested multi-domain experiments, the optimal setting



Figure 8.1.: Program flow of ICON-OpenACC.

for nproma_sub cannot be predetermined and must be set manually. It is recommended to start with a value of 5000 and adjust based on performance results obtained from a series of short benchmark experiments.

The ecRad radiation scheme (inwp_radiation =4) offers a McICA radiation solver optimized for OpenACC, which can be enabled by setting ecrad_isolver =2 (namelist radiation_nml, INTEGER value). It is important to note that for an experiment run on an accelerator, ecrad_isolver =2 must be set instead of the default ecrad_isolver =0 that would be used on a classical CPU system.

The namelist setting proc0_shift is not required for ICON-OpenACC and must not be set or set to its default value 0. proc0_shift is only required for the experiments on the NEC SX Aurora system.

## 8.5.3. Implementation Details

Current GPU supercomputers have separate GPU and CPU memories. The data transfer between the two is relatively slow compared to the direct access of local memory. Hence, recurrent data transfers in the time loop between CPU and GPU must be avoided, as the compute intensity (ratio of computations to memory load) in ICON is relatively low. On the other hand code that runs only once per simulation is less performance critical such that its porting expenses can be saved. These principles result in an ICON-OpenACC program flow (Figure 8.1) that runs the model initialization on CPU only, and activates the GPU when the first physics components are called. At the moment it seems not worth the effort to port a few rarely called code packages such as the output for data assimilation. However, these rare packages require the proper data transfer from GPU to CPU and possibly vice versa in the code.

The typical structure of an OpenACC-accelerated loop is presented in the following listing. The OpenACC code starts with the **!\$ACC** pragma. Note the different levels that are parallelized using OpenMP or using OpenACC.

In principle, ACC kernels (i.e. the code between ACC PARALLEL and ACC END PARALLEL) should run asynchronously so that the CPU can prepare the next kernel while the accelerator runs the current kernel.

Certain code such as utility functions or communication is used during the initialization as well as the time loop. This means that such code must be able to run on CPU and the accelerator using the respective memory. The OpenACC standard provides the IF(condition) clause to execute a code on the accelerator if condition evaluates to .TRUE. and on the CPU otherwise. All (modern) ICON subroutines that support OpenACC have an optional lacc argument that is used for this decision. As lacc can only be used in the subroutine if it is passed as an argument (present() in the Fortran sense), the non-optional variable lzacc is used in the subroutine code. The value of lzacc can be derived from lacc by calling set_acc_host_or_device(lzacc, lacc). The routine set_acc_host_or_device defaults to .FALSE. if lacc is not present. This means that code, which has no OpenACC support, does not have to be changed when a routine is ported and the routine's argument list is thus extended by lacc.

The dynamical core and legacy code use the global variable i_am_accel_node as a condition for the IF() clause. Originally, this variable have been introduced for a validation mode where a non-accelerator-processes computes the whole domain on CPU and compares it to the domain decomposed results from GPU nodes. Routines that use lacc and call other routines that depend on i_am_accel_node should have a CALL assert_lacc_equals_i_am_accel_node(routine_name, lacc) at their top.

The OpenACC directive code in ICON follows the "ICON OpenACC style and implementation guide" which can be found in the ICON developer wiki on Gitlab. The ICON ACC beautifier (https://gitlab.dkrz.de/dwd-sw/icon-openacc-beautifier) can be used to apply the spacing, comma, colon, capitalization and line continuation rules automatically as outlined in the style guide.

# 9. Programming ICON

Just because something doesn't do what you planned it to do doesn't mean it's useless.

Thomas Edison

The previous chapters' topics have been guided by questions of how to run ICON simulations in various settings and how to control and understand the model's characteristics. In this short chapter, instead, we will introduce ICON's inner workings, i.e. the code layout and the most important data structures.

The description is detailed enough to make it relatively easy for the reader to modify the code. We exemplify this in Section 9.3 by implementing an own simple diagnostic.

## 9.1. Representation of 2D and 3D Fields

We begin with a suitable representation of two- and three-dimensional fields. Here, we refer to a discrete variable as a 2D field if it depends on the geographical position only. A 3D field, in addition, contains a vertical dimension, associated with the grid column.

**Indexing.** Recalling the unstructured nature of ICON's computational grids (see Section 2.1) there is no obvious order of the cells in a 2D array like indexing them according to longitudes and latitudes. Instead, we just order the cells in a deliberate way and index them in this order with ascending integer numbers. This means that our 2D field becomes a 1D array, referenced by the cell indices as subscript values.

Most arrays are associated with the centers of the triangular grid cells, but we do that in a similar way for the edges and vertices of the triangles. An extension to 3D fields, i.e. including a vertical dimension, results in 2D arrays, the first index being the cell (or edge or vertex) index, the second index being the height level.

**Blocking.** For reasons of cache efficiency nearly all D0 loops over grid cells, edges, and vertices are organized in two nested loops: "jb loops" and "jc loops". Often, the outer loop ("jb") is parallelized with OpenMP.

With respect to the data layout, this means that the long vector is split into several chunks of a much smaller length nproma (this is a run-time parameter, defined in the namelist parallel_nml, see also Section 8.3.2). We store the long vector in a 2D array, the first ICON Model Tutorial



Figure 9.1.: Illustration of the 2D field representation. The original spherical domain is decomposed (light-gray: halo region). Afterwards, the long vector of grid cells is split into several chunks of a much smaller length nproma.

index counting the elements in a block (*line index*), the second index counting the blocks. The last block may be shorter since nproma is not necessarily a divisor of the number of cells. The blocking procedure is illustrated in the lower half of Figure 9.1. There exist auxiliary functions idx_no, blk_no and idx_1d, which help to calculate the blocked indices from the 1D array index and vice versa (declared in the module mo_parallel_config).

Finally, let us consider the 3D fields that were stored in 2D arrays, with the cell index as the first dimension and the second being the vertical coordinate. With index blocking, these fields will be stored in 3D arrays with the first index counting the elements in a block, the second index counting the levels and the third index counting the blocks. The reason is that the blocks are often passed one by one to some subprograms which are called in a loop over the blocks. Since Fortran stores arrays in column-major order, the data for a single jb is stored contiguously in memory. Thus we can pass this chunk of data to the subprograms without any reshaping of the arrays.

**Domain decomposition.** Domain decomposition is, naturally, a prerequisite for scalability on modern parallel computers. For large scale realistic ICON setups and with operational core counts in the range of tens of thousands, the use of persistent global-sized arrays is unacceptable. Each model domain is therefore distributed onto several processors¹. This means that we have only certain regions of a domain on each processor.

¹In the following, we will use the generic term "Processing Element" (PE).

Generally, several subdivision steps are performed recursively. The division criterion (module mo_setup_subdivision.f90) subdivides a partition wrt. the cell *latitudes* if the range of covered latitudes is larger than the range of longitudes, otherwise the subdivision operates on the *longitudes*. This procedure vaguely reminds of the creation of a kd-tree², but the method also accounts (empirically) for the convergence of meridians and it does some boundary smoothing to reduce inter-process communication.

Each processor's region consists of an inner portion and a lateral boundary portion. The latter may be either a lateral boundary for the entire domain or a *halo region*, i. e. a lateral boundary of the partial domain which is overlaid with neighboring partial domains. The halo region (which is also known as a *ghost-cell region*) is illustrated in the upper half of Figure 9.1.

The programmer is responsible for the distribution of the data among the processors and the correct communication through MPI calls. This means that all halo regions have to be updated by the neighboring partial domains. We will sketch this synchronization process in Section 9.2.4 below.

Keep in mind that it is the width of the halo region which defines a size limit for your stencil calculations: It is not possible to include cells in a stencil which extend beyond the halo region!

**Index ordering.** After the domain decomposition, which takes place in the model initialization phase, each PE performs a sorting of its locally allocated cells (and edges and vertices). This *local* index ordering is determined by the refin_xxx_ctrl index which counts the distance from the lateral boundary in units of cell/edge/vertex rows. In particular, note the refin_c_ctrl array which already played a role for the preparation of lateral boundary input data, see Section 2.3. Portions of the triangular cells correspond to different values of refin_c_ctrl, which allows a sorting into the following categories: the cell rows at the lateral boundary, the nudging zone, the inner cells, and the halo region. Of course, for a global domain only the two latter categories exist.

The upper part of Figure 9.2 schematically shows the different parts of a computational domain, subdivided between two PEs: Each PE "owns" a subset of interior cells and part of the lateral boundary. The halo region is shared between the PEs.

The lower part of Figure 9.2 visualizes the ordering of the grid portions in the index vector. It can be seen that the leftmost indices (i.e. the smallest subscripts) correspond to the lateral boundary region, followed by the prognostic cells. The sorting of these prognostic cells with respect to their cell row stops after the first cell row (denoted by "sorted" vs. "unsorted" in Fig. 9.2).

Thus the indices are ordered in such a way that typical iterations over grid portions like prognostic cells, lateral boundary points etc. can be realized without conditional statements. Each portion is annotated by its start index, where the subscript corresponds to the refin_c_ctrl value. For convenience, there exists the auxiliary function get_indices_c

²https://en.wikipedia.org/wiki/K-d_tree



start_idx(1) start of nudging zone start_idx(grf_bdywidth_c+1)

Figure 9.2.: Schematic illustration of ICON's index ordering (grid cells).

end idx(max rlcell)+1

The upper part schematically shows the different portions of a computational domain, subdivided between two PEs. The lower part of the illustration depicts the index vector for PE #0, see the explanation in the text. Both PEs "own" a subset of interior cells and part of the lateral boundary. The halo region that surrounds each partition is used to exchange data between the PEs (dashed): The halo-exchange operation copies the contents of the cells on the partition border (violet, pink) to the halo cells of the adjacent process (light violet, light pink).

= end_idx(min_rlcell_int)+1

(Note: The sizes of the regions shown differ from the real situation.)

(declared in the module mo_loopindices) which helps to adjust the loop iteration accordingly: For a given value of refin_c_ctrl and a specific block index we get the start and end indices to loop over.

Only the halo cells deserve some further remark (note the comment "when not overlapping with lateral boundary" in Fig. 9.2): In the special case, when no lateral boundary is present (for a global grid, say, or when a PE operates only on an inner portion of the domain), the halo cells are stored in a contiguous fashion at the end of the index vector. When a lateral boundary is present, however, there exist some halo cells which also belong to the lateral boundary. These cells are then sorted into the leading part of the index array, since the ability to address boundary cells in a contiguous fashion is much more important in practice. Note that this exceptional sorting of halo cells does not affect all halo cells of the lateral boundary region, though, but only their outermost rows. A possibility to

distinguish between prognostic cells and halo points is provided by the decomp_domain data structure and the "owner info" field, see the following section.

# 9.2. Data Structures

This section describes ICON's most important data structures. The majority of the data structures mentioned here exists several times, and a separate structure is created for each computation domain.

89

The number of computational domains can be taken from the variable n_dom which corresponds to the number of entries in the namelist parameter dynamics_grid_filename, see Page 118. Here, the reduced radiation grid (Section 3.10) is not taken into account.

## 9.2.1. Description of the Model Domain: t_patch

The t_patch data structure contains all information about the grid coordinates and topology, as well as parallel communication patterns and decomposition information. It is declared in src/shr_horizontal/mo_model_domain.f90 as an array of length (# domains), where the coarsest base grid is denoted by the index 1, while the refined domains are denoted by numbers 2, 3 and so on.

All contained data arrays and indices relate to the index/block ordering described in Section 9.1 and non-existent indices are denoted by -1. The most important contents of the t_patch data structure are

t_patch	
grid_filename	character string, containing grid file name
ldom_active	indicator if current model domain is active,
	see Section 5.2
parent_id	domain ID of parent domain
child_id(1:n_childdom)	list of child domain ID's
n_patch_cells/edges/verts	number of locally allocated cells, edges
n_patch_XXXX_g	global number of cells, edges and vertices
nblks_c/e/v	number of blocks
npromz_c/e/v	chunk length in last block
cells / edges / verts	grid information, see below
comm_pat_c/e/v	halo communication patterns, see Section 9.2.4
	:

When it comes to dimensioning fields, the application programmer normally uses the size n_patch_cells (or n_patch_cells_g for global arrays which should generally be

avoided) for cells, and n_patch_edges, n_patch_verts for edges and vertices, respectively. The product nproma*nblks_c only provides an upper bound: The last block of nproma indices does not necessarily have to be filled completely, which is indicated by the variable npromz_c.

The data members cells, edges, and verts, which are of the types t_grid_cells, t_grid_edges, and t_grid_vertices, respectively, give us information about the grid cells themselves, in particular about their geographical coordinates. For example,

t_grid_cells	
<pre>center(:,:)</pre>	longitude and latitude of cell circumcenters,
	dimensions: [1:nproma, 1:nblks_c]
<pre>neighbor_idx(:,:,:)</pre>	line indices of triangles next to each cell,
	dimensions: [1:nproma, 1:nblks_c, 1:3]
decomp_info	information on domain decomposition
	:

Essentially, all data arrays which are contained in the grid files and which are described in Section 2.1.1 have a counterpart in this derived data type.

Besides, the data member decomp_info which separately exists for cells, edges and vertices, deserves additional comments. Its data type t_grid_domain_decomp_info is declared in /src/parallel_infrastructure/mo_decomposition_tools.f90:

t_grid_domain_decomp_info	
<pre>glb_index(:)</pre>	global index of local cell,
	dimension: 1:n_patch_cells
<pre>decomp_domain(:,:)</pre>	domain decomposition flag, 0: owned (for cells), dimensions: [1:nproma, 1:nblks_c]
	:

The global index (glb_index) is particularly useful to perform operations (or write out data) which must not depend on the parallel domain decomposition of the model run. The "owner info" (decomp_domain) can be used to distinguish between prognostic cells and halo points whose values are just copied from adjacent PEs.

## 9.2.2. Date and Time Variables

When installing own processes within ICON's time loop, the question for the current (simulation) time naturally arises. All global date and time variables are contained in the data structure time_config, which is of the derived data type t_time_config and declared in src/configure_model/mo_time_config.f90. The dates are initialized with the corresponding namelist parameters given in Section 5.1.1.

9.2 Data Structures

t_time_config	
tc_exp_startdate	experiment start (tc means "time control")
tc_exp_stopdate	experiment stop
tc_startdate	start of current simulation. In case of restart this is the date at which the simulation has been continued.
tc_stopdate	end of single run
tc_current_date	current model date
	:

The dates and time spans make use of the mtime calendar library³ which is precise up to milliseconds without round-off errors. The mtime library resides in the directory externals/mtime. It is written in C and has a Fortran interface (module mtime).

We motivate the use of the mtime module by two examples. First, we perform a date calculation, adding a time span of 1 day to a given date. We make use of two variables: mtime_date (TYPE(datetime), POINTER) and mtime_td (TYPE(timedelta), POINTER).

```
mtime_td => newTimedelta("P01D")
mtime_date => newDatetime("2014-06-01T00:00:00")
mtime_date = mtime_date + mtime_td
CALL datetimetostring(mtime_date, dstring)
WRITE (*,*) "2014-06-01T00:00:00 + 1 day = ", TRIM(dstring)
CALL deallocateDatetime(mtime_date)
CALL deallocateTimedelta(mtime_td)
```

As a second example, we demonstrate the mtime event mechanism which may be used to start certain processes in the program. An event (TYPE(event), POINTER) is defined by a start date, a regular trigger interval and an end date. Besides, let RefDate denote the event reference date (anchor date) in our example. Then the event triggers every RefDate + k * interval, but only within the bounds given by startDate and endDate.

## 9.2.3. Data Structures for Physics and Dynamics Variables

On each model domain we need the same collection of 2D and 3D fields in order to describe the state of the atmosphere. These fields are collected in the data

³The NWP mode uses the *proleptic* Gregorian calendar that is a backward extension of the Gregorian calendar to dates before its introduction October 15, 1582.

structure t_nh_state. This derived type and the following types are declared in src/atm_dyn_iconam/mo_nonhydro_types.f90.

First, the prognostic fields, which are integrated over time, are collected in the data structure t_nh_prog. Elements of t_nh_prog are allocated for each time slice that is needed for the time integration. For the nonhydrostatic time integration, the number of time slices is two, time t for the current time and  $t + \Delta t$  for the prediction.

t_nh_prog	
W	orthogonal vertical wind $[m s^{-1}]$
vn	orthogonal normal wind $[m s^{-1}]$
rho	density $[kg m^{-3}]$
exner	Exner pressure
tke	turbulent kinetic energy $[m^2 s^{-2}]$
tracer	tracer concentration $[kg kg^{-1}]$
	:

A global variable p_nh_state of type t_nh_state is instantiated in the module /src/atm_dyn_iconam/mo_nonhydro_state.f90. This is an array whose index corresponds to the model domain. The density of the atmosphere as state variable of the nonhydrostatic dynamical core is therefore given as

```
p_nh_state(domain)%prog(time slice)%rho(index, level, block)
```

Regarding the *time slice* argument, we briefly comment on ICON's handling of the twotime-level scheme and the mechanism to avoid reallocation or unnecessary data copies:

For each prognostic variable in the two-time-level scheme, two arrays are pre-allocated:

```
p_nh_state%prog(1)%field(:,:)
p_nh_state%prog(2)%field(:,:)
```

Additionally, we introduce two global  $\tt INTEGER$  index variables  $\tt nnow$  and  $\tt nnew$  which we initialize at model start with

nnow = 1nnew = 2

The result values of the integration scheme (time slice  $t + \Delta t$ ) are stored on the **nnew** time level. We therefore access the data on this time level by

```
p_nh_state%prog(nnew)%field(:,:)
```

While the calculations in the dynamical core fill this array with values, the prognostic state of the "old" time step can be accessed by

```
p_nh_state%prog(nnow)%field(:,:)
```

Then, at the end of each dynamic (sub-)step, the time step n+1 becomes the "old" one, while the time step n is freed and can be used as the new working array for the time stepping. This operation does not require any copying but merely exchanges the roles of nnow and nnew:

### CALL swap(nnow, nnew)

Alas, the whole process is complicated by the following two facts: First, nnow, nnew are defined for each domain separately. The above examples therefore require the domain index jg, i.e. nnow(jg), nnew(jg). Second, as it has been explained in Section 3.7.1, different integration time steps are applied for efficiency reasons. A separate nnow_rcf/nnew_rcf accounting is required for the basic time step which is used for tracer transport, numerical diffusion and the fast-physics parameterizations, to distinguish it from the short time step used within the dynamical core⁴.

The data type t_nh_diag (defined in src/atm_dyn_iconam/mo_nonhydro_types.f90) contains a collection of diagnostic fields, determined by all prognostic variables, boundary conditions and the compositions of the atmosphere.

t_nh_diag	
u	zonal wind $[m s^{-1}]$
v	meridional wind $[m s^{-1}]$
temp	temperature [K]
pres	pressure [Pa]
	÷

Similar to the prognostic fields, the domain-wise data of type t_nh_diag can be accessed via p_nh_state(*domain*)%diag.

## 9.2.4. Parallel Communication

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

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

These take the specific halo region as an argument (data type t_comm_pattern) and several exchange patterns are pre-defined for each domain (see the derived data type t_patch). For example, comm_pat_c, defines the halo communication for *cells* which are owned by neighboring processes. The subroutine call

```
CALL exchange_data(patch%comm_pat_c, array)
```

would perform a typical synchronization of the halo regions.

Additionally, there exist variants of the exchange_data routine for gather operations. Calling exchange_data with an argument of type t_comm_gather_pattern, typically the pre-defined data structure patch%comm_pat_gather_c, takes elements from many processes and gathers them to one single process. There are corresponding data structures for communicating edges and vertices.

As it has been noted in Section 8.2, there exist different process groups in ICON: I/O, restarting and computation. These groups are related to MPI communicators which are

 $^{^4\}mathrm{Here},$  the suffix  $\mathtt{rcf}$  stands for "reduced calling frequency".

ICON Model Tutorial

defined in the module src/parallel_infrastructure/mo_mpi.f90. Probably the most important MPI communicator in the ICON code is p_comm_work (which is identical to the result of a call to get_my_mpi_work_communicator()). This is the MPI communicator for the *work group*.

The work group has a total size of num_work_procs, where each process may inquire about its rank by calling get_my_mpi_work_id(). On a non-I/O rank, its work group comprises all processes which take part in the prognostic calculations. It is therefore used by the exchange_data synchronization routine.

A final remark is related to the everlasting chit-chat of the status log (screen) output: The process_mpi_stdio_id is always the 0th process of the MPI communicator process_mpi_all_comm, which is the MPI communicator containing all PEs that are running this model component. A typical code line for printing out a message to screen would be

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

With this, the message print-out would be suppressed on all PEs 1, 2, ...

Of course, there already exists an auxiliary subroutine **message()** in ICON, whose exact purpose is what we achieved manually above:

```
CALL message('caller', 'message text')
```

The message() subroutine is located in the module mo_exception and restricts the printout to PE #0. It takes the caller's subroutine name as an additional argument.

# 9.3. 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. Here, we try to keep things as simple and short as possible.

Adding new modules: The dependency generator of ICON automatically detects Fortran (.f90) and C (.c) files in the src/ directory. These files are automatically included in the compilation process. Thus, when creating a backup copy of a file, it must not end with .f90 or .c to avoid ambigious module and subroutine names.

Adding new fields. ICON keeps so-called *variable lists* of its prognostic and diagnostic fields. This global registry eases the task of memory (de-)allocation and organizes the field's meta-data, e.g., its dimensions, description and unit. The basic call for registering a new variable is the add_var command (module mo_var_list). Its list of arguments is rather lengthy and we will discuss 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). To keep this presentation as short as possible we have omitted the necessary USE statements:

The derived types t_cf_var and t_grib2_var are defined in the modules mo_cf_convention and mo_grib2, respectively. The expression grib2_var is actually a call to a constructor function, also defined in mo_grib2, which takes a triple of integers (*discipline*, *parameterCategory*, *parameterNumber*) as the field specifier.

Let us create an INTEGER array of length 2 with the name shape2d_c, denoting the dimensions of the new variable. The dimensions of a 2D field will be explained below. Here we take them as given:

shape2d_c = (/ nproma, nblks_c /)

Now, with the essential ingredients at hand, we define our new field by the following call. We will place it at the very end of the subroutine new_nh_state_diag_list in the module mo_nonhydro_state.

CALL add_var( p_diag_list, 'newfield',	&
<pre>p_nh_state(domain)%diag%newfield,</pre>	&
GRID_UNSTRUCTURED_CELL, ZA_SURFACE,	&
cf_desc, grib2_desc,	&
ldims=shape2d_c, lrestart=.FALSE. )	

The INTEGER parameters GRID_UNSTRUCTURED_CELL and ZA_SURFACE define the type of the horizontal grid and the (trivial) vertical axis. From now on the new field can be specified in the output namelists that were described in Section 7:

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

83

The extra_2d and extra_3d fields: When debugging the model code, it is often advantageous to be able to output intermediate results and ad hoc calculated diagnostic fields. However, it would be an unnecessary effort to define and allocate new variables especially for these test situations. ICON has a special mechanism for this purpose:

By setting the namelist parameter inextra_2d (namelist io_nml, INTE-GER value) or inextra_3d (namelist io_nml, INTEGER value), respectively, a number of 2D or 3D cell-based floating-point arrays with the names extra_2d1, extra_2d2, ..., and extra_3d1, extra_3d2, ... is automatically created. Inside the model code, these can be accessed as

p_nh_state(domain)%diag%extra_2d_ptr(1)%p_2d(:,:)

and similar. Note that all extra variables are actually stored in common buffers

p_nh_state(domain)%diag%extra_2d(:,:,1:inextra_2d)
p_nh_state(domain)%diag%extra_3d(:,:,:,1:inextra_3d)

The fields can be used as output buffers for the temporary output data then.

Looping over the grid points. Of course, the newly created field 'newfield' still needs to be filled with values. As explained in Section 9.1 above, nearly all DO loops that iterate over grid cells are organized in two nested loops: "jb loops" and "jc loops". Here the outer loop ("jb") is parallelized with OpenMP and limited by the *cell block number* nblks_c. The innermost loop iterates between 1 and the block length nproma.

Since the ICON model is usually executed in parallel, we have to keep in mind that each process can perform calculations only on a portion of the decomposed domain. Moreover, some of the cells between interfacing processes are duplicates of cells from neighboring sub-domains (so-called *halo cells*). Often it is not necessary to loop over these halo points, since they will be updated by the next parallel synchronization call.

The auxiliary function get_indices_c (declared in the module mo_loopindices) helps to adjust the loop iteration accordingly:

The constants grf_bdywidth_c and min_rlcell_int can be found in the modules mo_impl_constants_grf and mo_impl_constants, respectively. Note that these constants have to be inserted in start_block, end_block and also in the argument list of get_indices_c. A graphical interpretation of these constants is provided by Figure 9.2. The loop example therefore iterates over all prognostic cells (denoted by the blue area).

*Loop exchange:* The special pre-processor flag __LOOP_EXCHANGE can be found in numerous places of the ICON model code. It causes a loop interchange in many performance critical loops: If applied, the variable used in the inner loop switches to the outer loop.

Usually, this means that the loop over the vertical levels becomes the fastest running loop, compared to the iteration indices for the horizontal location. When arrays do not contain vertical levels, access to array elements may take advantage of the CPU cache.

**Placing the subroutine call.** Having encapsulated the computation together with the DO loops in a custom subroutine, we are ready to place this subroutine call in between ICON's physics-dynamics cycle.

Let us take a look at Figure 3.8: The outer loop "Dynamics  $\rightarrow$  Physics  $\rightarrow$  Output" is contained in the core module mo_nh_stepping inside the TIME_LOOP iteration. For diagnostic calculations it is important to have all necessary quantities available for input. On the other hand the result must be ready before the call to the output module,

CALL write_name_list_output(jstep)

The fail-safe solution here is to place the call immediately above this call.

Having inserted the call to the diagnostic field computation, we are done with the final step. Recompile the model code and you are finished!

*Style recommendations:* When writing your own extensions to ICON it is always a good idea to keep an eye on the quality of your code.

Make sure that there is **no duplicate functionality** and try to improve the readability of your subroutines through **indentation**, **comments** etc. This will make it easier for other developers to understand and assimilate. Better introduce own **modules** with complete interfaces and avoid USEs and PUBLIC fields.

A good starting point for your own project are the **template files**, given in the subdirectory **src/templates** of the ICON source code. These provide examples for modules, functions and subroutines.

Additional remarks: 3D fields, accumulated quantities. For the sake of brevity, only the simple case of two-dimensional fields has been discussed so far.

ICON Model Tutorial

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

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

This information needs to be provided to the constructor add_var(...), together with an INTEGER parameter which indicates the type of the vertical axis:

Usually, the ICON generates its vertical axis on-the-fly, i.e. during the model setup. The user may choose between the hybrid Gal-Chen coordinate and the (more common) SLEVE coordinate via namelist parameters, see Section 3.4. However, it is practically impossible to encode the exact vertical coordinate parameters themselves in the data sets which are produced by the ICON model. Apart from very basic information like the number of vertical levels, only a number identifying the special vertical grid used is provided. This indirect approach is indicated by the parameter ZA_REFERENCE.

Finally, it is often necessary to **reset accumulated quantities** in regular intervals. This can be achieved by

```
action_list = actions(new_action(ACTION_RESET, interval))
```

For example, by setting *interval* = "PTO6H", the respective field would be reset to zero every 6 hours.

## 9.4. NWP Call Tree

All of ICON's NWP and infrastructure modules, however numerous they may be, can roughly be classified into an initialization phase, the time integration loop and the cleanup phase. In Figure 9.3 we restrict ourselves to the most important subprograms.

These are listed together with a short description of their location and purpose, which, of course, change gradually between released versions. We recommend to compare this to the flow chart of processes in the physics-dynamics coupling, see Fig. 3.8.



Figure 9.3.: Call tree of ICON's NWP component (note that this list has been restricted to the most important subroutines).

9. Programming ICON

# 10. Post-Processing and Visualization

Oh my God! Look at that picture over there! There's the Earth coming up. Wow, is that pretty.

W. Anders, Apollo 8

ICON offers the possibility to produce output either in NetCDF or GRIB2 format. Many visualization environments such as GrADS, Matlab or R now include packages with which NetCDF data files can be handled. The GRIB format, which is also commonly used in meteorology, can be processed with these tools as well. However, since the standardization of unstructured GRIB records is relatively new, many post-processing packages offer only limited support for GRIB data that has been stored on the triangular ICON grid.

Since the visualization of *regular* (lat-lon) grid data is relatively straightforward, we limit ourselves in our description to a very simple program, ncview, which does not have a large functionality but is an easy-to-use program for a quick view of NetCDF output files. It is therefore very useful for a first impression.

Model data that has been stored on the *triangular* ICON grid can be visualized with the Python programming language, the NCL scripting language, R or the Generic Mapping Tools (GMT). Section 10.3 contains some examples how to visualize NetCDF data sets without the need of an additional regridding.

# 10.1. Retrieving Data Set Information

We begin with command-line utilities which provide a textual description of the data sets. For a quick overview of dimensions and variables, the command-line utility ncdump can be used. This program will shortly be described first. More sophisticated tools exist, for cutting out subsets of data, e.g., and producing averages or time series. One of these tools are the cdo utilities.

## 10.1.1. The ncdump Tool

The tool ncdump comes with the NetCDF library as provided by Unidata and generates a text representation of a NetCDF file on standard output. The text representation is in a form called CDL (network Common Data form Language). ncdump may be used as a simple browser for NetCDF data files, to display the dimension names and sizes, variable names, types and shapes, attribute names and values, and optionally, the data values themselves for all or selected variables in ASCII format. For example, to investigate the structure of a NetCDF file, use

ncdump -h data-file.nc

With this command, dimension names and sizes, variable names, dependencies and values of dimensions will be displayed. To show the type version of a NetCDF file, type

ncdump -k data-file.nc

This gives information about the NetCDF base format of a specific file, for example NetCDF Classic Format, NetCDF 64-bit Offset Format or NetCDF-4 Format.

NetCDF data can also be redirected to a text file with

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

This produces an annotated CDL version of the structure and the data in the NetCDF file *data-file.nc*. You can also save data for specified variables to text files just using:

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

For further information on working with ncdump see

https://docs.unidata.ucar.edu/nug/current/netcdf_utilities_guide.html

### 10.1.2. CDO – Climate Data Operators

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

#### https://code.mpimet.mpg.de/projects/cdo

There is a possibility to get support via the forums¹ or issue tracking system of CDO. Further details are provided in the  $FAQ^2$ .

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

¹https://code.mpimet.mpg.de/projects/cdo/boards

²https://code.mpimet.mpg.de/projects/cdo/wiki/FAQ



Figure 10.1.: Screenshot of the ncview NetCDF plotting tool.

```
cdo sinfov data-file.nc
```

which prints out short information of each field.

## 10.2. Plotting Data Sets on Regular Grids: ncview

ncview is a visual browser for NetCDF format files developed by David W. Pierce. Using ncview you can get a quick and easy look at *regular* grid data in your NetCDF files.

To install ncview on your local platform, see the ncview website: http://meteora.ucsd.edu/~pierce/ncview_home_page.html

You can run the program by typing:

ncview data-file.nc

which will open a new window with the display options. It is possible to view simple movies of data, view along different dimensions, to have a look at actual data values at specific coordinates, change color maps, invert data, etc., see the screenshot in Fig. 10.1.

If *data-file.nc* contains wildcards such as '*' or '?' then all files matching the description are scanned, if all of the files contain the same variables on the same grid. Choose the variable you want to view. Variables which are functions of longitude and latitude will be displayed in two-dimensional images. If there is more than one time step available you can easily view a simple movie by just pushing the forward button. The appearance of the image can be changed by varying the colors of the displayed range of the data set values or by adding/removing coastlines. Each one- or two-dimensional subset of the data can be selected for visualization. **ncview** allows the selection of the dimensions of the fields available, e.g. longitude and height instead of longitude and latitude of 3D fields.

The pictures can be sent to Postscript (*.ps) output by using the function print. Be careful that whenever you want to close only a single plot window to use the close button, because clicking on the  $\boxtimes$ -icon on the top right of the window will close all ncview windows and terminate the entire program!

## 10.3. Plotting Data Sets on the Triangular Grid

Let us now focus on ICON's original computational mesh, the triangular grid. In this section we will sketch several approaches for the visualization of data sets based on the triangular cells.

In recent years, numerous open source packages have appeared: We will present quick start examples for the Python programming language, NCL, R, and GMT. The different tools vary significantly in terms of their functionality, but also regarding their state of development – some software packages (PyNGL, NCL) have been discontinued, so no further development is taking place. All approaches have in common that the NetCDF grid file must be read in together with the data file.

### 10.3.1. Visualization with Python

By now there is quite a number of visualization packages available that rely on the Python ecosystem.

Here, as a introductory example, we will be using Matplotlib, a plotting library for the Python programming language and its numerical mathematics extension NumPy. The Cartopy package extends the Matplotlib functionality and offers map projection definitions, and arbitrary point, line, and polygon transformations (see the Cartopy list of projections for more information).

The following minimal example displays ICON surface data. First, we load the necessary Python modules for data read-in and plotting:

```
import pathlib, numpy, netCDF4, cartopy
from matplotlib import pyplot as plt
```

Second, set the file locations. For convenience, we use a path specification relative to the user's **\$HOME** directory:

```
home = str(pathlib.Path.home())
grid_filename = home + "icon_DOM01.nc"
data_filename = home + "external_parameter_icon_DOM01_tiles.nc"
```

Open the NetCDF grid file and load the data sites. The data sites for the surface height field topography_c are the triangle circumcenters, located at clon, clat.

```
ds = netCDF4.Dataset(grid_filename)
cx = numpy.degrees(numpy.asarray(ds["clon"]))
cy = numpy.degrees(numpy.asarray(ds["clat"]))
```

Load external parameters data set topography_c from a second file:



Figure 10.2.: The plot generated by the Python example script using Matplotlib and Cartopy, see Section 10.3.1.

```
ds = netCDF4.Dataset(data_filename)
src_data = numpy.degrees(numpy.asarray(ds["topography_c"]))
```

Finally, we plot with the function tricontourf, resulting in the plot shown in Fig. 10.2:

```
fig = plt.figure(figsize=(9, 9))
ax = plt.axes(projection=cartopy.crs.PlateCarree())
ax.add_feature(cartopy.feature.BORDERS,edgecolor='gray')
ax.tricontourf(cx, cy, src_data, transform=cartopy.crs.PlateCarree())
plt.show()
fig.savefig("HSURF.png", dpi=200)
```

*Psyplot:* For more advanced visualizations we recommend at this point the package *psyplot*, again based on the Matplotlib package. See the website https://psyplot.github.io for further information including a rich collection of documented examples. Psyplot even features a viewer application for netCDF files, see https://psyplot.github.io/psy-view/index.html, that is highly motivated by the neview software but works also for unstructured grids.

## 10.3.2. Visualization with PyNGL

The Python module PyNGL (pronounced "pingle") is a Python language module for generating high-quality, 2D visualizations of scientific data. PyNIO is a Python module used for reading and writing NetCDF, GRIB, and HDF.

**Development status.** PyNGL has been developed by the Computational and Information Systems Lab at the National Center for Atmospheric Research (NCAR). Unfortunately, in 2020 the further development of PyNGL has been abandoned, in favor of Mat-PlotLib, Cartopy, and other alternative Python ecosystem packages.

#### **PyNGL** Installation

For completeness, we will explain the basic usage of PyNGL in a step-by-step tutorial in the following. PyNGL and PyNIO can be installed on Linux and MacOS systems via  $conda^3$ , or its slimmed-down version *miniconda*:

First, download the *miniconda* installer script to your machine. In our example we will install the stable release version for Linux x86 architectures. Visit the *conda* web page to find out which version is the right one for your platform.

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
source $HOME/miniconda3/etc/profile.d/conda.sh
```

Once *miniconda* is installed, we use the **conda** command to install the *PyNGL* and *PyNIO* Python packages:

```
conda create --name pyn_env --channel conda-forge numpy pyngl pynio
```

Now, you can activate Python environment via

conda activate pyn_env

and the newly installed packages *PyNGL* and *PyNIO* are ready to use.

89

NumPy and threading issues: On some platforms (like DWD's rcl.dwd.de) you may encounter a SegFault when using the NumPy package in Python+conda. A similar problem occurs when using the packages PyNGL and PyNIO. This problem, which is related to the multi-threaded BLAS library, can be avoided by setting in your shell environment

export OMP_NUM_THREADS=1

#### PyNGL Step-by-step Tutorial

In the following we provide a detailed step-by-step tutorial for producing graphics from an ICON data set. To this end, we create a file visualization_tutorial.py where a sequence of commands can be stored and executed with

#### python visualization_tutorial.py

 $^{^3}Conda$  is an open source package management system see https://conda.io.
We begin by loading some Python modules which provide high-level functions for plotting and numerical computations

# load Python modules for plotting
import Ngl, Nio, numpy

These lines also contain a comment. Comment lines in Python are preceded by the hash sign '#'.

**Step 1: Reading grid coordinates from file.** Since 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 = Nio.open_file("gridfile.nc")
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 fields (this is explained in more detail in Section 2.1.1, page 21). For PyNGL we convert from steradians to degrees:

```
vlon = numpy.rad2deg( gridfile.variables["vlon"][:] )
vlat = numpy.rad2deg( gridfile.variables["vlat"][:] )
```

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

edge_vertices = gridfile.variables["edge_vertices"][:]

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

edge #i:  $(vlon[q_1], vlat[q_1]) - (vlon[q_2], vlat[q_2])$ 

where

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

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

**Step 2: Creating a plot of the triangular grid.** Producing graphics with PyNGL requires the creation of a so-called *workstation*, i.e. a description of the output device. In this example, this "device" will be an image file plot.png, but we could also define a different output format, e.g. PostScript "ps" instead.

wks = Ngl.open_wks("png","plot")

Then the map settings have to be defined and we collect these specifications in a "resource 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 the resource attributes bglFrame and nglDraw 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 = Ngl.Resources()
config1.nglFrame = False
config1.nglDraw = False
config1.mpProjection = "Orthographic"
config1.mpGreatCircleLinesOn = True
config1.mpCenterLonF = 10
config1.mpCenterLatF = 50
```

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

map = Ngl.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 dimension 2 * nedges:

```
ecx = numpy.ravel( vlon[ edge_vertices-1 ], order='F' )
ecy = numpy.ravel( vlat[ edge_vertices-1 ], order='F' )
```

This operation deserves additional comments: First of all, since the PyNGL plotting function below will expect one-dimensional lists for its interface, we use the auxiliary function numpy.ravel for reshaping the array of lines. Second, with order='F' we define the array ordering of vlon, vlat to be "Fortran-style". Compared to Python (numpy) this is the transpose memory layout.

Now, there exists an PyNGL high-level command for plotting lines, Ngl.add_polyline.

```
lines_cfg = Ngl.Resources()
lines_cfg.gsSegments = numpy.arange(0, edge_vertices.size, 2)
poly = Ngl.add_polyline(wks, map, ecx, ecy, lines_cfg)
```



Figure 10.3.: The two plots generated by the PyNGL example script in Section 10.3.2.

The whole plotting process is then triggered by the command

Ngl.draw(map) Ngl.frame(wks) Ngl.end()

Here, the call to Ngl.end() terminates the PyNGL script. The first of the resulting image files plot.000001.png will contain an illustration similar to Fig. 10.3 (left part).

**Step 3: Loading a data set from a second file.** We now assume that the data sets produced by the ICON model have been stored in NetCDF format. This allows to visualize the unstructured data without additional Python packages. 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 = Nio.open_file("datafile.nc")
topo = datafile.variables["topography_c"]
print(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 print, the data sites for the field topography_c are the triangle circumcenters, located at clon, clat.

```
clon = numpy.rad2deg( gridfile.variables["clon"][:] )
clat = numpy.rad2deg( gridfile.variables["clat"][:] )
```

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.

<pre>config2 = Ngl.Resources()</pre>	
config2.mpProjection	<pre>= "CylindricalEquidistant"</pre>
config2.cnFillOn	= True
config2.cnLinesOn	= False
config2.cnLineLabelsOn	= False
config2.sfXArray	= clon
config2.sfYArray	= clat

Afterwards, we generate the plot (image file plot.000002.png) with a call to Ngl.contour_map.

map = Ngl.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 PyNGL 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 PyNGL. This information is also available from the grid file and is stored in the fields clon_vertices, clat_vertices.

```
clonv = numpy.rad2deg( gridfile.variables["clon_vertices"][:] )
clatv = numpy.rad2deg( gridfile.variables["clat_vertices"][:] )
config2.sfXCellBounds = clonv
config2.sfYCellBounds = clatv
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 Ngl.contour_map.

```
map = Ngl.contour_map(wks,topo,config2)
```

Do you see the difference?

## 10.3.3. NCL – NCAR Command Language

The NCAR Command Language (NCL) is an interpreted language designed specifically for scientific data analysis and visualization. It is built on top of the same "resource model" used in the PyNGL Python package.

NCL allows convenient access to data in a variety of formats such as NetCDF and GRIB1/2, among others. It has many features common to modern programming languages, such as types, variables, operators, expressions, conditional statements, loops, and functions and procedures, see https://www.ncl.ucar.edu for details.

Besides an interactive mode, NCL allows for script processing (recommended). NCL scripts are processed on the command-line by typing

```
ncl filename.ncl
```

For visualizing ICON data on the native triangular grid, we recommend using NCL 6.2.0 or higher.

**Development status.** Note that NCAR has made the decision to adopt Python as the scripting language platform of choice for future development of analysis and visualization tools. NCAR will not add new features to the NCL language or function library although the NCL software package continues to build on currently supported platforms.

### NCL Quick-Start Example

The following example script creates a temperature contour plot with NCL (see Figure 10.4):

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
                                         ; define triangulation
                           = vlon
 ResC@sfYCellBounds
                             vlat
                                         ; define triangulation
 ResC@cnFillOn
                                         ; do color fill
                             True
 ResC@cnFillMode
                             "cellfill"
 ResC@cnLinesOn
                           = False
                                         ; no contour lines
; plot temperature level
plot = gsn_csm_contour_map(wks,temp_ml(0,80,:),ResC)
end
```

To open a data file for reading, the function addfile returns a file variable reference to the specified file. Second, for drawing graphics, the function gsn_open_wks creates an output resource, where the "ps", "pdf" or "png" format are available. Third, the command gsn_csm_contour_map creates and draws a contour plot over a map.

Loading the coordinates of the triangle cell centers into NCL (resources sfXArray and sfYArray) is essential for visualizing ICON data on the native grid. Loading the vertex coordinates of each triangle (resources sfXCellBounds and sfYCellBounds), however, is optional. If not given, a Delaunay triangulation in the 2D plane will be performed by NCL, based on the cell center information. If given, the triangles defining the mesh will be deduced by sorting and matching vertices from adjacent cell boundaries. If you are interested in the correct representation of individual cells, the resource sf[X/Y]CellBounds should be set.

Creating a plot can get very complex depending on how you want to look at your data. Therefore we refer to the NCL documentation that is available online under



#### http://www.ncl.ucar.edu

Figure 10.4.: ICON temperature field on a specific model level produced with the above NCL script.

Note that there is another NCL example contained in the course material, which very closely resembles the steps explained in the PyNGL tutorial in Section 10.3.2. 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.

## 10.3.4. Visualization with R

Section authors J. Förstner and M. Köhler, DWD Physical Processes Division

R is a free software environment for statistical computing and graphics. R is available as free software under the terms of the Free Software Foundation's GNU General Public License. More Information can be found here:

https://www.r-project.org

To start R in an interactive mode simply type

R

on the command line. Afterwards R commands can be entered, which are then interpreted line by line. R can be extended (easily) via packages. Additional packages have to be installed via the R command

```
install.packages("package_name")
```

For the compilation of some packages it might be necessary to provide specific (development) libraries on the system and to give information about the include and library paths as additional arguments to that command.

Most packages are available through the CRAN family of internet sites. An exception to this is the gribr package, which is used in the example script below to read in GRIB2 data:

#### https://github.com/nawendt/gribr

This package uses and therefore needs a recent installation of ecCodes (it is not working with the GRIB API). Additional information about (other) prerequisites and the installation of the package can be found on the given website.

# E?

Installation of R on the DWD computer system: The R software has been preinstalled on DWD's rcl.dwd.de as a software module, where R/3.6.2 is the current default (for which the below example has been tested). Further modules have to be loaded as well, see the comment in the following example.

Besides an interactive mode, R allows for script processing (recommended). R scripts are processed on the command-line by typing

Rscript filename.R

As default a PDF named **Rplots.pdf** will be created.

#### **R Quick-Start Example**

The following example script icon_native_4_tutorial.R can be found in the tarball directory scripts. It creates a global temperature contour plot with R (see Figure 10.5):



Figure 10.5.: ICON 2 m temperature field produced with the given example script for R.

```
# ... on DWD's RCL, as a prerequisite issue the following commands:
# module load netcdf4 oracle
# module load R
# module load gribr
#
# load necessary libraries
library(gribr)
library(RNetCDF)
library(data.table)
library(ggplot2)
library(dplyr)
 library(colorRamps)
# --- setup ---
shortName <- "T 2M"
          <- paste(shortName," ICON (R2B6)\n")
title
```

```
# data file names
grid <- "./icon_grid_0024_R02B06_G.nc"</pre>
data <- "./T_2M.R2B06_ICON_global.grb"</pre>
# --- icon grid ------
ncHandle <- open.nc(grid)</pre>
# function to convert coordinates in degress
rad2deg <- function(rad) {(rad * 180) / (pi)}</pre>
# get longitudes and latitudes of triangle vertices of a cell
 vlon <- rad2deg(var.get.nc(ncHandle,"clon_vertices"))</pre>
 vlat <- rad2deg(var.get.nc(ncHandle,"clat_vertices"))</pre>
 close.nc(ncHandle)
# --- icon data -------
 gribHandle <- grib_open(data)</pre>
# select the grib record based on a given list of keys
 gribRecord <- grib_select(gribHandle, list(shortName = shortName))</pre>
 grib_close(gribHandle)
# create a data table with data to plot - ids and values are tripled
 DT <- data.table(lon = as.vector(vlon),</pre>
                 lat = as.vector(vlat),
                 id = rep(1:(dim(vlon)[2]), each=3),
                 var = rep(gribRecord$values, each=3))
# --- domain and edges ------
# Global
xrange <- c(-180., 180.)</pre>
yrange <- c( -90.,
                    90.)
# select the subset of ids in the domain
usedIDs <- unique(DT[lon%between%xrange & lat%between%yrange]$id)</pre>
# use only the respective subset of the data
DT <- DT[id %in% usedIDs]
# special treatment for the cells near the date line
# ... when the 3 corners on opposite sides of the date line move one corner
IDsR = DT[,list( (max(lon)-min(lon))>200 & mean(lon)>0.0 ), by=id][V1==T]$id
 IDsL = DT[,list( (max(lon)-min(lon))>200 & mean(lon)<0.0 ), by=id][V1==T]$id</pre>
 DT[id %in% IDsR & lon < 0.0, lon := lon + 360.0]
DT[id %in% IDsL & lon > 0.0, lon := lon - 360.0]
# ... copy triangles by 360deg to fill holes near date line
DTT <- DT[id%in%IDsR]
 DTT[lon > 0.0, lon := lon - 360.0]
```

```
DTT$id <- DTT$id + max(DT$id)</pre>
 DT
       <- rbind(DT, DTT)
 DTT
        <- DT[id%in%IDsL]
 DTT[lon < 0.0, lon := lon + 360.0]
 DTT$id <- DTT$id + max(DT$id)</pre>
       <- rbind(DT, DTT)
 DT
# --- plot data using ggplot2 with geom_polygon ------
# landscape mode
pdf(paper="a4r", width=11.692, height=8.267)
# create the plot object
pp <- ggplot() +</pre>
      geom polygon(data = DT, aes(x = lon, y = lat, group = id, fill = var)) +
      borders(colour = "black", xlim = xrange, ylim = yrange)
      scale_fill_gradientn(colours = matlab.like(100))
                                                                               +
+
      coord_cartesian(xlim = xrange,ylim = yrange, expand = FALSE)
      scale_x_continuous(breaks = seq(xrange[1], xrange[2],20))
      scale_y_continuous(breaks = seq(yrange[1], yrange[2],15))
      theme bw()
      labs(x="longitude", y="latitiude", fill="[K]")
      theme(axis.text = element_text(size=14),
            axis.title = element_text(size=16),
            plot.title = element_text(size=16, hjust=0),
            legend.title = element text(size=16),
            legend.text = element text(size=14))
      ggtitle(paste0(title,
              " min: ", round(min (DT$var, na.rm=TRUE),3), ", ",
              " max: " , round(max (DT$var, na.rm=TRUE),3), ", ",
              "mean: ", round(mean(DT$var, na.rm=TRUE),3), ", ",
              " std: ", round(sd (DT$var, na.rm=TRUE),3)))
# issue the plot object
 pp
```

# 10.3.5. 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:

https://www.generic-mapping-tools.org

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
         = horizonal
#
# File = NWP_DOMO1_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  $-\mathbf{Z})$  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 10.6.

#!/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 some recent releases 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 10.6.: ICON temperature field on a specific model level produced with the above GMT script.

# **10.4.** Post-Processing of Data Sets

# 10.4.1. Post-Processing using the CDO

The CDO (Climate Data Operators) have already been introduced in the previous Section 10.1.2 for retrieving information on data files. Additionally, the CDO are also capable of remapping data to regular grids. Basically, two steps are necessary to process the ICON output files:

• In the first call of the CDO ("gencon"), 1st order conservative remap weights are generated and stored to a separate file:

```
cdo gencon, lat-lon_grid_description icon_grid coefficient_file
```

The generation of the interpolation weights may take some time.

• Afterwards, the interpolation is done with

```
cdo remap, grid_description, coefficient_file in_file out_file
```

Here, the file *icon_grid* is a NetCDF file which contains the topological and geometric information about the triangular ICON grid. This grid information must correspond to the data set *in_file* and it is not part of the data set itself. Instead, it must be downloaded separately from the web. See http://icon-downloads.mpimet.mpg.de/dwd_grids.xml for the list of "official" DWD ICON grids⁴.

With respect to data sets with "missing values" the above CDO workflow with precalculated weights fails. The following remark applies, for example, to the ICON-D2 data sets:

Since the "missing values" are not part of the source grid definition, the CDOs detect the masking only when reading the data file. Then, to avoid the risk of using undefined values for the interpolation stencils, the whole process is aborted in this case. A workaround when interpolating a single file is to do the weight calculation simultaneously with the interpolation:

```
cdo remapcon, grid_description -setgrid, icon_grid:2 in_file out_file
```

In this example we have also set the source grid index explicitly to 2 to distinguish between the cell grid, edge grid, and the vertex grid in the  $icon_grid$  file. The concrete index can be obtained from the output of cdo sinfov data-file.nc.

*DWD OpenData Documentation:* A valuable web resource in this context is DWD's OpenData website. Here, ICON data sets are made freely available to the public and the whole procedure of remapping ICON data with the CDO tool is also described.

OpenData information on ICON forecast data:

⁴Currently in operational use is the global grid #26 with 13 km horizontal grid spacing (http:// icon-downloads.mpimet.mpg.de/grids/public/edzw/icon_grid_0026_R03B07_G.nc).

https://www.dwd.de/DE/leistungen/opendata/hilfe.html

Furthermore, there even exists a Docker software container in which all necessary packages for the CDO and the ecCodes are combined:

https://github.com/deutscherwetterdienst/regrid

The structure of the lon-lat grid description file is explained in the CDO documentation⁵. A global regular grid, for example, would be defined as

gridtype	=	lonlat
xsize	=	1440
ysize	=	721
xfirst	=	0.00
xinc	=	0.25
yfirst	=	-90.00
yinc	=	0.25

*Caveat:* Alas, the CDO do not handle the entire set of GRIB2 meta-data correctly. Some meta-data items, for example those which regard ensemble runs, still remain unsupported. However, this limitation does not matter if the desired output format is NetCDF and not GRIB2, or if the processed data fields are rather standard. Alternatives for remapping ICON data sets to lon-lat grids are the *fieldextra* software (which is a COSMO software) or the DWD ICON Tools, which are internally used at DWD but lack official support.

### 10.4.2. Post-Processing using Fieldextra

Section authors

P. Baumann and J.-M. Bettems, MeteoSwiss

Fieldextra is an official post-processing software of the COSMO Consortium. It is maintained and developed at MeteoSwiss.

Fieldextra is a generic tool to manipulate NWP model data and gridded observations. It offers best compatibility with ICON(-ART), COSMO(-ART), and IFS models. Besides support of regular grids, the program supports both GRIB2 and NetCDF data that has been stored on the triangular ICON grid, and provides transparent access to the DWD ICON tools interpolation methods (see Section 1.3.1).

The software is implemented in Fortran 2008 as a single but modular code. A control file composed of Fortran namelists defines the set of operations to apply on the input data. Simple data processing and more complex data operations are supported. Fieldextra is designed as a toolbox; a set of primitive operations is provided, which can be freely combined and iterated.

As examples of post-processing tasks, we mention

⁵Appendix D, https://code.mpimet.mpg.de/projects/cdo/embedded/index.html.

- complex transformations from one grid to another, including the vertical dimension, for example the interpolation of pollen boundaries from ICON-ART to a grid at higher horizontal and vertical resolution,
- computation of the height of the tropopause, based on complex conditions on meteorological fields, including gradients,
- computation of ensemble-based products, like probabilities, quantiles, standard deviation,
- format transformations between GRIB1, GRIB2, and NetCDF (including EXTPAR output).

But fieldextra is also useful to solve pre-processing tasks like

- remapping gridded Radar observations from a regular grid to the ICON triangular grid to feed the latent-heat-nudging process,
- merging the sea surface temperature from the IFS into the ICON analysis at sea points,
- generating initial conditions on the ICON unstructured grid from initial conditions calculated on a regular grid, like COSMO type data,
- interpolating IFS boundary conditions from the regular grid to the ICON unstructured grid.

Since the primary focus of the program is the production environment, a lot of effort has been put into

- robustness of the code,
- extensive reporting of exceptions,
- careful processing of field meta-information, with systematic checks to avoid meaningless products,
- IO, memory, and CPU optimization (OpenMP parallelism), and
- comprehensive diagnostic and profiling.

The program is used, in particular, for production at MeteoSwiss, at DWD and other centers, and for COSMO-LEPS at ECMWF.

An overview document, a primer, release notes, and other information are publicly available at the dedicated fieldextra GitHub wiki at https://github.com/COSMO-ORG/fieldextra-wiki/wiki.

Full installations of the latest releases of fieldextra are accessible for the UNIX group cfxtra on the Atos HPCF at ECMWF in /ec/res4/hpcperm/chcosmo/projects/fieldextra.

If you want to install field extra on your own platform, you can either retrieve the code from the private field extra repository on GitHub⁶, or download a self-contained package

⁶https://github.com/COSMO-ORG/fieldextra, restricted access

of the latest major production release from the COSMO web page at http://www.cosmo-model.org/content/support/software/default.htm#fieldextra.

In order to access the GitHub repository, you need to have access to the GitHub platform. For that, you first have to create a GitHub account, which is free of charges, and then ask for access to the fieldextra repository on GitHub via the mailing list fieldextra@cosmo-model.org.

The code is portable. It uses only standard Fortran features and should work on any UNIX / Linux platform. Once the code is installed, all resources files used at run time are also provided. Furthermore, comprehensive documentation, including an introductory tutorial, and many commented examples are available. A set of command line tools, based on fieldextra, is also part of the installation.

Fieldextra is subject to licensing. Its usage is free to all COSMO members. Community support is available via the mailing list fieldextra@cosmo-model.org. Licences are free for the R&D community, but without support.

For a first introduction to fieldextra, take a look at the overview and the first contact documents, which are available at the fieldextra GitHub wiki⁷. The first contact document in particular provides some insight into the internal processing of the software, which helps to better understand the namelists. As a next step, the commented examples in the cookbook folder of the code repository provide an overview of the large spectrum of possible applications (see cookbook/README.cookbook), and are a good start to develop your own namelists. A comprehensive description of the program functionalities is given in the file documentation/README.user. Also available is a FAQ document in documentation/FAQ; consult it before looking for support!

⁷https://github.com/COSMO-ORG/fieldextra-wiki/wiki

# 11. ICON's Data Assimilation System and Analysis Products

In this chapter you will get to know basic components of the ICON data assimilation system. It consists of a whole collection of programs and modules both for the atmospheric variables of the model as well as for soil, snow, ice and sea surface, all collected into the *Data Assimilation Coding Environment* (DACE). The analysis products of this software package are discussed in Section 11.3.

# 11.1. Data Assimilation

Numerical weather prediction (NWP) is an initial value problem. The ability to make a skillful forecast heavily depends on an accurate estimate of the present atmospheric state, known as *analysis*. In general, an analysis is generated by combining, in an optimal way, all available observations with a short term forecast of a general circulation model (e.g. ICON).

Stated in a more abstract way, the basic idea of data assimilation is to fit model states x to observations y. Usually, we do not observe model quantities directly or not at the model grid points. Here, we work with observation operators H which take a model state and calculate a simulated observation y = H(x). In terms of software, these model operators can be seen as particular modules, which operate on the ICON model states. Their output is usually written into so-called feedback files, which contain both the real observation  $y_{meas}$  with all its meta data (descriptions, positioning, further information) as well as the simulated observation y = H(x).

However, data assimilation cannot be treated at one point in time only. The information passed on from the past is a crucial ingredient for any data assimilation scheme. Thus, *cycling* is an important part of data assimilation. It means that we

- 1. Carry out the core data assimilation component 3D-VAR to calculate the so-called analysis  $x^{(a)}$ , i.e. a state which best fits previous information and the observations y,
- 2. Propagate the analysis  $x_k^{(a)}$  to the next analysis time  $t_{k+1}$ . Here, it is called *first guess* or *background*  $x_{k+1}^{(b)}$ .
- 3. Carry out the next analysis by running the core data assimilation component, generating  $x_{k+1}^{(a)}$ , then cycling the steps.

See Figure 11.1 for a schematic of the basic assimilation process.



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

## 11.1.1. Variational Data Assimilation

The basic 3D-VAR step minimizes the functional

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

where B is the background state distribution *covariance matrix* which is making sure that the information which is available at some place is distributed into its neighborhood properly, and R is the error covariance matrix describing the error distribution for the observations. The minimizer of (11.1) is given by

$$x^{(a)} = x^{(b)} + BH^T (R + HBH^T)^{-1} (y - H(x^{(b)}).$$
(11.2)

The background or first guess  $x^{(b)}$  is calculated from earlier analysis by propagating the model from a state  $x_{k-1}$  at a previous analysis time  $t_{k-1}$  to the current analysis time  $t_k$ . In the data assimilation code, the minimization of (11.1) is not carried out explicitly by (11.2), but by a conjugate gradient minimization scheme, i.e. in an iterative manner, first solving the equation

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

in observation space calculating  $z_k$  at time  $t_k$ , then projecting the solution back into model space by

$$\delta x_k = x_k^{(a)} - x_k^{(b)} = BH^T z_k.$$

We call  $\delta x_k$  the analysis increment.

The background covariance matrix B is calculated from previous model runs by statistical methods. We employ the so-called NMC method initially developed by the US weather bureau. The matrix B thus contains statistical information about the relationship between different variables of the model, which is used in each of the assimilation steps.

### 11.1.2. Ensemble Kalman Filter

To obtain a better distribution of the information given by observations, modern data assimilation algorithms employ a dynamical estimator for the covariance matrix (*B*-matrix). Given an ensemble of states  $x^{(1)}, ..., x^{(L)}$ , the standard stochastic covariance estimator calculates an estimate for the *B*-matrix by

$$B = \frac{1}{L-1} \sum_{\ell=1}^{L} (x_k^{(\ell)} - \overline{x}_k) (x_k^{(\ell)} - \overline{x}_k)^T, \qquad (11.3)$$

where  $\overline{x}$  denotes the mean defined by

$$\overline{x}_k = \frac{1}{L} \sum_{\ell=1}^{L} x_k^{(\ell)}, \ k \in \mathbb{N}.$$

This is leading us to the *Ensemble Kalman Filter* (EnKF), where an ensemble is employed for data assimilation and the covariance is estimated by (11.3). Here, we use the name EnKF (ensemble Kalman filter) as a generic name for all methods based on the above idea.

In principle, the EnKF carries out cycling as introduced above, just that the propagation step carries out propagation of a whole *ensemble* of L atmospheric states  $x_k^{(a,\ell)}$  from time  $t_k$ to time  $t_{k+1}$ , and the analysis step has to generate L new analysis members, called the *analysis ensemble* based on the first guess or background ensemble  $x^{(b,\ell)}$ ,  $\ell = 1, ..., L$ .

Usually, the analysis is carried out in observation space, where a transformation is carried out. Also, working with a low number of ensemble members as it is necessary for large-scale data assimilation problems, we need to suppress spurious correlations which arise from a naive application of Eq. (11.3). This technique is known as *localization*, and the combined transform and localization method is called *localized ensemble transform Kalman filter* (LETKF), first suggested by Hunt et al. (2007).

The DWD data assimilation coding environment (DACE) provides a state-of-the-art implementation of the LETKF which is equipped with several important ingredients such as different types of covariance *inflation*. These are needed to properly take care of the *modeling error*. The original Kalman filter itself does not know what error the model has and thus by default under-estimates this error, which is counter-acted by a collection of tools.

#### 11.1.3. Hybrid Data Assimilation

The combination of variational and ensemble methods provides many possibilities to further improve the state estimation of data assimilation. Based on the ensemble Kalman filter LETKF the data assimilation coding environment provides a hybrid system EnVar, the ensemble variational data assimilation.

The basic idea of EnVar is to use the dynamical flow dependent ensemble covariance matrix B as a part of the three-dimensional variational assimilation. Here, localization is a crucial issue, since in the LETKF we localize in observation space, but 3D-VAR employs B in state space. Localization is carried out by a *diffusion*-type approximation in DACE.

The cycling for the EnVar needs to cycle both the ensemble  $x^{(\ell)}$ ,  $\ell = 1, ..., L$  and one deterministic state  $x_{det}$ . The resolution of the ensemble can be lower than the full deterministic resolution. By default we currently employ a 40 km grid spacing for the ensemble and a 13 km global grid spacing for the deterministic state. The ensemble *B* matrix is then carried over to the finer deterministic resolution by interpolation. See Section 11.2 for more details on the operational assimilation system at DWD.

## 11.1.4. Surface Analysis

DACE provides additional modules for Sea Surface Temperature (SST) analysis, Soil Moisture Analysis (SMA) and snow analysis. Characteristic time scales of surface and soil processes are typically larger than those of atmospheric processes. Therefore, it is often sufficient to carry out surface analysis only every 6 to 24 hours.

# 11.2. Assimilation Cycle at DWD

The assimilation cycle iterates the steps described in Section 11.1: updating a short-range ICON forecast (first guess) using the observations available for that time window to generate an analysis, from which then a new updated first guess is started.

The core assimilation for atmospheric fields is based on a hybrid system (EnVar) as described in Section 11.1.3. At every assimilation step (every 3 h) an LETKF is ran using an ensemble of ICON first guesses. Currently, the ensemble consists of 40 members with a horizontal grid spacing of 40 km and a 20 km nest over Europe. A convex linear combination of the 3D-VAR climatological and the LETKF's (flow dependent) covariance matrix is then used to run a deterministic 3D-VAR analysis at 13 km horizontal grid spacing.

In addition, the above mentioned surface modules are run: Sea Surface Temperature (SST) analysis, Soil Moisture Analysis (SMA) and snow analysis.

Note that for the ICON-EU nest no assimilation of atmospheric fields is conducted. Instead the necessary atmospheric analysis increments are interpolated from the underlying global grid. Together with the available first guess fields on the nest they form the nest analysis. A separate surface analysis, however, is conducted.

The deterministic as well as the ensemble analysis is generated 8 times a day. Based on the former, deterministic forecasts are launched at approx. 13 km horizontal grid spacing globally with a 6.5 km nest over Europe. The maximum forecast time of the whole system is limited to +30 h lead time at 03/09/15/21 UTC. Otherwise, the system is integrated up

to  $+120\,\mathrm{h}$  while at  $00/12\,\mathrm{UTC}$  the integration on the global domain (only) is prolonged to  $+180\,\mathrm{h}$  lead time.

Since the beginning of 2018 ICON ensemble forecasts are conducted as well. On the basis of the analysis ensemble 40 short to medium forecasts at 40 km globally and 20 km nested over Europe are run 8 times a day. The maximum forecast times are equivalent to those of the deterministic system (see above). The primary purpose of the ensemble forecasts is to estimate the forecast uncertainty, which arises due to uncertainties in the initial conditions and the model error.

The input, output and processes involved in the assimilation cycle are briefly described below:

## **Atmospheric Analysis**

Fields modified by the atmospheric analysis: (see Appendix B for a description of each variable) t, p, u, v, qv.

#### Grid(s) on which it is performed: global

Carried out at every assimilation time step (3 h) using the data assimilation algorithms described in the previous sections.

Main input: First guess, observations, previous analysis error, online bias correction files.

Main output: Analysis of the atmospheric fields, analysis error, bias correction files, feedback files with information on the observation, its departures to first guess and analysis.

The system can make use of the following observations: radiosondes, weather stations, buoys, aircraft, ships, radio occultations, AMV winds and radiances. Available general features of the module are variational quality control and (variational) online bias correction. Regarding EnKF specifics, different types of inflation techniques, relaxation to prior perturbations and spread, adaptive localization, SST perturbations and SMA perturbations are available.

#### **Snow Analysis**

Fields modified by the snow analysis: (see Appendix B 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 (3h).

Main input: SYNOP snow depth observations if the coverage is sufficient. If this is not the case, more sources of information are looked for until the number of observations is high enough, namely (and in this order), precipitation and 2 m temperature, direct observations (wwreports) and the NCEP external snow analysis.

Main output: Analysis of the snow fields.

## Sea Surface Temperature Analysis

Fields modified by the SST analysis: (see Appendix B 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 B for a description of each variable)  $w_{so}$ .

Grid(s) on which it is performed: global, EU-nest

Carried out only once a day, at 0 UTC.

Main input: Background fields for relevant fields at every hour since last assimilation, 2 m-temperature analysis (see below) to be used as observations.

Main output: Soil moisture analysis and estimated error.

#### **2m Temperature Analysis**

Although carried out only at 0 UTC, it is run for several time steps in between to provide the output (2 m temperature) needed by the SMA analysis. Uses observations from SYNOP stations on land and METAR information from airports.

# **11.3.** Analysis Products

This section provides an overview of DWD's analysis products.

#### 11.3.1. Uninitialized Analysis for IAU

In the *incremental analysis update* (IAU) method (Bloom et al., 1996, Polavarapu et al., 2004) the analysis increment is not added completely at a particular time step, but it is embedded into the model integration and added to the model states  $x_k^{(b)}$  during an interval  $\Delta t$ , which by default is  $\Delta t = 3$  h for global forecasts. This method of tentatively pulling the model from its current state (first guess) towards the analyzed state acts as a low pass

filter in the frequency domain on the analysis increments, such that small scale unbalanced modes are effectively filtered.

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

- The analysis file has to contain analysis increments (i.e. deviations from the first guess) instead of full fields, with validity time 00 UTC. The only exceptions are FR_ICE and T_SEA (or alternatively T_SO(0)), which must be full fields (see Table 11.1).
- The model must be started from a first guess which is shifted back in time by 1.5 h w.r.t. the analysis. Thus, in the given example, the validity time of the first guess must be 22:30 UTC of the previous day. This is because "dribbling" of the analysis increments is performed over the symmetric 3 h time window [00 UTC 1.5h, 00 UTC + 1.5h]. See Section 5.1.3 for an illustration of this process.

Table 11.1 provides an overview of the fields contained in the *uninitialized analysis product* for IAU for 00 UTC. Columns 1 to 3 show DWD's GRIB2 shortName, the unit and a short description of the fields, respectively. Columns 4 and 5 indicate, whether the field is part of the first guess file and/or analysis file. The marker  $\otimes$  highlights analysis increments as opposed to full fields. First guess fields which are optional for starting the model with IAU are highlighted in blue, with their scope being indicated in the description. If one or more of these fields are unavailable, a cold-start of these fields is performed given that the parameterizations for which they are needed are activated.

As explained in Section 11.2, the atmospheric analysis is performed more frequently than the surface analysis. Therefore, the analysis product at times different from 00 UTC usually contains only a subset of the fields provided at 00 UTC. Consequently, Table 11.1 will look different for non-00 UTC runs in such a way that the fields

FR_ICE T_SEA W_SO T_2M

will not be provided in the analysis file. The first three fields of this list will be contained in the first guess file instead. Table 11.1.: Content of the uninitialized analysis product for IAU, separated into first guess (FG) and analysis (ANA). Optional fields for model initialization are marked in blue. The marker ⊗ indicates analysis increments as opposed to full fields. Analysis fields highlighted in red are only available for 00 UTC. The validity date of the first guess is shifted back by 1.5 h w.r.t. the start date.

shortName	Unit	Description		urce
		-	$\mathbf{FG}$	ANA
DEN	${ m kg}{ m m}^{-3}$	air density	Х	
Р	Pa	pressure		$\otimes$
QC	${\rm kgkg^{-1}}$	cloud liquid water mass fraction	×	
QI	${\rm kgkg^{-1}}$	cloud ice mass fraction	×	
QR	${\rm kgkg^{-1}}$	rain water mass fraction	×	
QS	${\rm kgkg^{-1}}$	snow mass fraction	×	
QV	${\rm kgkg^{-1}}$	water vapor mass fraction	×	$\otimes$
Т	Κ	air temperature		$\otimes$
THETA_V	Κ	virtual potential temperature	×	
TKE	$\mathrm{m}^2\mathrm{s}^{-2}$	turbulent kinetic energy	×	
U, V	${ m ms^{-1}}$	horizontal velocity components		$\otimes$
VN	${ m ms^{-1}}$	edge normal velocity component	×	
W	${ m ms^{-1}}$	vertical velocity	×	
ALB_SEAICE	%	<pre>sea ice albedo scope: lprog_albsi=.TRUE. (namelist lnd nml)</pre>		
C_T_LK	1	shape factor w.r.t. temp. profile in the thermocline)		
EVAP_PL	${\rm kg}{\rm m}^{-2}$	<pre>evaporation of plants (integrated since "nightly reset") scope: itype_trvg=3 (namelist lnd_nml)</pre>	×	
FRESHSNW	1	age of snow indicator	×	$\otimes$
FR_ICE	1	sea/lake ice fraction		×
H_ICE	m	sea ice depth	×	
H_ML_LK	m	mixed-layer thickness	×	
H_SNOW	m	snow depth	×	$\otimes$
HSNOW_MAX	m	maximum snow depth reached within current snow-cover period scope: itype_snowevap=3 (namelist lnd nml)	×	
QV S	$\mathrm{kg}\mathrm{kg}^{-1}$	surface specific humidity	×	
RHO_SNOW	${ m kg}{ m m}^{-3}$	snow density	×	

Continued on next page

SKT	Κ	skin temperature	×	
		lnd nml)		
CNOAC	d	duration of automatic anomy server period	~	
SNUAG	u	duration of current show-cover period	X	
		scope: itype_snowevap=5 (namenst		
a	07	ING_NMI)		
SNOWC	%	snow cover	X	
T_BOT_LK	Κ	temperature at water-bottom sediment	×	
		interface		
T_G	Κ	surface temperature	×	
T_ICE	Κ	sea ice temperature	×	
T_MNW_LK	Κ	mean temperature of the water column	×	
T_SEA	Κ	sea surface temperature		×
T_SNOW	Κ	snow temperature	×	
T_WML_LK	Κ	mixed-layer temperature	×	
W_I	${\rm kgm^{-2}}$	water content of interception layer	×	
ZO	m	surface roughness length	×	
T_SO	Κ	soil temperature	×	
T_2M	Κ	$2\mathrm{m}$ temperature bias		$\otimes$
		<pre>scope: itype_vegetation_cycle=3</pre>		
		(namelist extpar_nml)		
T 2M FILTBIAS	Κ	Time-filtered T 2M bias	×	
		scope: itype vegetation cycle=3		
		(namelist extpar nml)		
W_SO	${ m kg}{ m m}^{-2}$	soil water content (liq. $+$ ice)	×	$\otimes$
W SO ICE	$kg m^{-2}$	soil ice content	×	-
	0			

Table 11.1.: Continued from previous page

Please note that all GRIB2-keys of the fields T_2M and T_2M_FILTBIAS are identical, except for the key *typeOfGeneratingProcess*. In order to decode T_2M_FILTBIAS correctly and to distinguish it from T_2M, it is recommended to make use of the most recent DWD-specific GRIB definition files (see Section 1.1.2).

## 11.3.2. Uninitialized Analysis

The *uninitialized analysis* without IAU can be used if, for some reason, the model should be started without any noise filtering procedure. The first guess and analysis file are read in and merged by the model, i.e. the model state is abruptly pulled towards the analyzed state right before the first time integration step. This conceptually easy approach comes at the price of a massively increased noise level at the beginning of the simulation. The validity time of the *first guess* and *analysis* must match the model's start date. Table 11.2 provides an overview of the fields contained in the *uninitialized analysis product* for 00 UTC. Columns 4 and 5 again indicate, whether a specific field is contained in the first guess file and/or analysis file. Fields which are optional for starting the model are highlighted in blue, with their scope being indicated in the description. If one or more of these fields are unavailable, a cold-start of these fields is performed given that the parameterizations for which they are needed are activated.

As already explained in the previous section, the analysis at times different from 00 UTC will only contain a subset of the fields provided at 00 UTC. Table 11.2 will differ for non-00 UTC runs in the way that the fields

FR ICE	H ICE	T ICE	T SEA	W SO
_	_	_	_	_

will not be available from the analysis file but from the first guess file.

Table 11.2.: Content of the uninitialized analysis product, separated into first guess(FG) and analysis (ANA). Optional fields for model initialization are markedin blue. Analysis fields highlighted in red are only available for 00 UTC.

shortName	Unit	Description	Soι	ırce
			$\mathbf{FG}$	ANA
DEN	${ m kgm^{-3}}$	air density	×	
Р	Pa	pressure		×
QC	${\rm kgkg^{-1}}$	cloud liquid water mass fraction	×	
QI	${ m kgkg^{-1}}$	cloud ice mass fraction	×	
QR	${ m kgkg^{-1}}$	rain water mass fraction	×	
QS	${ m kgkg^{-1}}$	snow mass fraction	×	
QV	${\rm kgkg^{-1}}$	water vapor mass fraction		×
Т	Κ	air temperature		×
THETA_V	Κ	virtual potential temperature	×	
TKE	$\mathrm{m}^2\mathrm{s}^{-2}$	turbulent kinetic energy	×	
U, V	${\rm ms^{-1}}$	horizontal velocity components		×
VN	${ m ms^{-1}}$	edge normal velocity component	×	
W	${\rm ms^{-1}}$	vertical velocity	×	
ALB_SEAICE	%	sea ice albedo	×	
		scope: lprog_albsi=.TRUE.		
a m t 17	1	(namenst ind_nmi)		
C_T_LK	1	shape factor w.r.t. temp. profile in the thermocline)	×	

Continued on next page

EVAP_PL	${\rm kgm^{-2}}$	evaporation of plants (integrated since "nightly reset")	×	
		<pre>scope: itype_trvg=3 (namelist lnd_nml)</pre>		
FRESHSNW	1	age of snow indicator		×
FR_ICE	1	sea/lake ice fraction		×
H_ICE	m	sea ice depth		×
H_ML_LK	m	mixed-layer thickness	×	
H_SNOW	m	snow depth		×
HSNOW_MAX	m	maximum snow depth reached within current snow-cover period	×	
		<pre>scope: itype_snowevap=3 (namelist lnd_nml)</pre>		
QV_S	${ m kgkg^{-1}}$	surface specific humidity	×	
RHO_SNOW	${\rm kg}{\rm m}^{-3}$	snow density	×	
SNOAG	d	duration of current snow-cover period	×	
		<pre>scope: itype_snowevap=3 (namelist lnd_nml)</pre>		
T_BOT_LK	Κ	temperature at water-bottom sediment interface	×	
T_G	Κ	surface temperature	×	
T_ICE	Κ	sea ice temperature		×
T_MNW_LK	Κ	mean temperature of the water column	×	
T_SEA	Κ	sea surface temperature		×
T_SNOW	Κ	snow temperature		×
T_WML_LK	Κ	mixed-layer temperature	×	
W_I	${\rm kg}{\rm m}^{-2}$	water content of interception layer	×	
W_SNOW	${\rm kg}{\rm m}^{-2}$	snow water equivalent	×	
ZO	m	surface roughness length	×	
T_SO	K	soil temperature	×	
W_SO	${ m kg}{ m m}^{-2}$	soil water content (liq. $+$ ice)		×
W_SO_ICE	${ m kg}{ m m}^{-2}$	soil ice content	×	

 Table 11.2.: Continued from previous page

# 11.3.3. Initialized Analysis

The *initialized analysis* is strongly related to the *uninitialized analysis for IAU*. It is a by-product of starting the model from the latter. E.g. the *initialized analysis* at 00 UTC is

generated by starting the model from the 22:30 UTC first guess, and adding the analysis increments over an asymmetric time window of 1.5 h width until 00 UTC. Therefore the noise level of the initialized analysis product is comparable to that of the *uninitialized analysis product for IAU*.

For model initialization the validity time of the *initialized analysis product* must match the model's start date. Table 11.3 provides an overview of the fields contained in the *initialized analysis product* for 00 UTC. Fields which are optional for starting the model are highlighted in blue. If one or more of these fields are unavailable, a cold-start of these fields is performed given that the parameterizations for which they are needed are switched on.

Note that this product is also suitable for initializing COSMO limited area simulations. To this end it contains the atmospheric fields T, P, U, V rather than ICON's set of prognostic atmospheric variables, i.e. THETA_V, RHO, VN. For ICON, the necessary transformation is performed automatically during startup.

For recent analysis dates (say August 2018 and later) this analysis product contains both SMI and W_SO. For previous analysis dates only W_SO is available. ICON can read any of them, however SMI is preferred and W_SO is the fallback. Whenever this analysis product needs to be remapped and SMI is not available, it is strongly recommended to manually convert W_SO to SMI beforehand, since interpolating W_SO can lead to strong numerical artifacts. See Section 2.2.3 for more details.

Table 11.3.:	Content of the initialized analysis product. Fields which are optional for
	model initialization are marked in blue.

shortName	$\mathbf{Unit}$	Description
Р	Pa	pressure
QC	${ m kgkg^{-1}}$	cloud liquid water mass fraction
QI	${ m kgkg^{-1}}$	cloud ice mass fraction
QR	${ m kgkg^{-1}}$	rain water mass fraction
QS	${ m kgkg^{-1}}$	snow mass fraction
QV	${ m kgkg^{-1}}$	water vapor mass fraction
Т	Κ	air temperature
TKE	$\mathrm{m}^2\mathrm{s}^{-2}$	turbulent kinetic energy
U, V	${\rm ms^{-1}}$	horizontal velocity components
W	${ m ms^{-1}}$	vertical velocity
ALB_SEAICE	%	sea ice albedo
		scope: lprog_albsi=.TRUE. (namelist
		lnd_nml)

Continued on next page

C_T_LK	1	shape factor w.r.t. temp. profile in the thermocline)
EVAP_PL	${\rm kgm^{-2}}$	evaporation of plants (integrated since "nightly reset")
		<pre>scope: itype_trvg=3 (namelist lnd_nml)</pre>
FRESHSNW	1	age of snow indicator
FR_ICE	1	sea/lake ice fraction
H_ICE	m	sea ice depth
H_ML_LK	m	mixed-layer thickness
H_SNOW	m	snow depth
HSNOW_MAX	m	maximum snow depth reached within current snow-cover period
		<pre>scope: itype_snowevap=3 (namelist lnd_nml)</pre>
QV_S	${ m kgkg^{-1}}$	surface specific humidity
RHO_SNOW	${\rm kg}{\rm m}^{-3}$	snow density
SNOAG	d	duration of current snow-cover period
		<pre>scope: itype_snowevap=3 (namelist lnd_nml)</pre>
T_BOT_LK	Κ	temperature at water-bottom sediment
		interface
T_G	К	surface temperature
T_ICE	К	sea ice temperature
T_MNW_LK	К	mean temperature of the water column
T_SNOW	Κ	snow temperature
T_WML_LK	Κ	mixed-layer temperature
W_I	${ m kg}{ m m}^{-2}$	water content of interception layer
W_SNOW	${ m kg}{ m m}^{-2}$	snow water equivalent
ZO	m	surface roughness length
SMI	1	soil moisture index
T_SO	Κ	soil temperature
W_SO	${\rm kg}{\rm m}^{-2}$	soil water content (liq. $+$ ice)
W_SO_ICE	${\rm kgm^{-2}}$	soil ice content
HHL	m	vertical coordinate half level heights

 Table 11.3.: Continued from previous page

Please note that in contrast to the Uninitialized Analysis for IAU product (Table 11.1) the Initialized Analysis product does not contain the skin temperature  $T_SKT$  and the time-filtered 2 m temperature bias  $T_2M_FILTBIAS$ . If the skin temperature parameterization is activated (itype_canopy=2) but an initial  $T_SKT$  is missing,  $T_SKT$  is initialized with the surface temperature  $T_S$ . It is nevertheless recommended to switch on the skin temperature

parameterization. The lack of initial conditions for  ${\tt T_SKT}$  does only have a marginal effect on the forecast quality.

Similarly, if the enhanced vegetation cycle parameterization is activated (itype_vegetation_cycle=3) but T_2M_FILTBIAS is missing, T_2M_FILTBIAS is initialized with zero (which effectively resembles itype_vegetation_cycle=2).

# A. The Computer System at DWD

# Available Platforms at DWD

The NEC SX-Aurora supercomputer system at DWD is one of our main platforms for the execution of the ICON-NWP model. This system consists of numerous compute nodes with corresponding cross-compilation nodes.

 rcl.dwd.de → rcn1100, rcn1101: These are the cross-compilation nodes (x86 AMD "Rome", 32 cores, 2.5 GHz), which run a Red Hat Enterprise Linux.

They are used for **compiling and linking**, preparation of data and basic editing work. Tools for the **visualization of meteorological fields** (CDO, NCL, ncview) are also available here.

The rcl.dwd.de nodes are not used for running parallel NWP simulations, but ICON jobs can be submitted to the NEC SX-Aurora compute nodes vhXXX (vector computer) or the rcnXXX (Linux cluster). Note that there are also big memory nodes rcbXXX available with a memory limit of 192G for each process.

To compile and link applications on the routine cluster use either the intel or the gnu compiler. To compile and link targeting the NEC SX-Aurora platform the nfort NEC compiler must be used.

## • NEC SX-Aurora:

The NEC SX-Aurora research cluster has 440 **x86 vector hosts**, where each node is equipped with 8 **NEC SX-Autora 1 TSUBASA Type 10AE vector engine CPUs**. The vector hosts are AMD EPYC "Rome" processors with 24 cores, 2.8GHz, and 256 GiB memory. ¹ Each vector engine achieves ca. 2.15 TFLOPS double precision peak performance and 8 computational cores. The vector nodes cannot be accessed interactively, but only by **PBS batch jobs** together with the queuing system NQSV from NEC.

Such jobs can use up to 24 GB of main memory per vector engine.

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

/hpc/uhome/username (\$HOME)
 Directory for storing source code and scripts to run the model. This is a GPFS file system suitable for many small files.

¹Note that 8 of these 24 x86 cores are reserved for the vector engine OS.

• /hpc/uwork/username (\$WORK)

Directory for storing larger amounts of data. For the \$WORK (Lustre) file system there is a common quota for every user of 7.9 TBytes.

• /hpc/gtmp/username (Lustre file system, contains \$TMPDIR) Temporary directory for storing larger amounts of data.

# The Batch System for the NEC SX-Aurora

Jobs for the NEC SX-Aurora system have to be submitted from the login nodes rcl.dwd.de, rcnl100, rcnl101 with the batch system PBS. 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:

qsub job_name	submit a batch job to PBS.
qstat	query the status of all batch jobs on the NEC SX-Aurora. You can see whether jobs are $Q$ (queued) or $R$ (running). You have to submit jobs to the queue $sx_norm$ .
qstat -u <i>user</i>	query the status of all your batch jobs on the machine.
<pre>qdel job_nr@machine</pre>	cancel your job(s) from the batch queue of a machine. The $job_nr$ is given by qstat -w.

qcat -o -f job_nr@machine follow stdout and stderr interactively.

For the vector engines you have to use the queue **sx_norm**. Here are some important PBS options.

#PBS -q sx_norm	define the queue
#PBS -T necmpi_hydra	batch job topology
#PBS -1 cpunum_job=5	no. of Xeon cores used on VH
<b>#</b> PBS -1 elapstim_req= <i>00:40:00</i>	time limit
#PBSvenode=2	total number of VEs required
#PBSvenum-lhost=2	no. of VE per logical host; max is 8
#PBS -1 coresz_prc=0	limit on max. core file size
#PBSuse-hca=2	necessary (no. host channel adapters per logical host)
#PBS -j o	set "oe" to join STDOUT and STDERR streams
#PBS -o filename	Redirect STDOUT
#PBS -v N1=var1,N2=var2	define environment variables

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_0_WORKDIR** makes it easy to return to the directory from which you submitted the job:

cd \$PBS_0_WORKDIR

To start a parallel executable, the mpirun command has to be used:

```
mpirun -x -venode -node 0-$N1 -np $PPN \
    -env OMP_NUM_THREADS $OMP_NUM_THREADS /path/to/application
```

Here the options have the following meaning (non-hybrid execution):

OMP_NUM_THREADS	number of OpenMP threads per task
-np	total number of MPI procs per job.
N1	one less than number of vector engines per node (here: 2) $$

More detailed information and documentation to the NEC SX-Aurora and related software can be found on the NEC web pages: https://www.hpc.nec/documentation

A. DWD HPC System

# **B. Table of ICON Output Variables**

The following table contains the NWP variables available for  $output^1$ . Please note that the field names are following an ICON-internal nomenclature, see Section 7 for details. The list also contains tile-based fields (suffix  $t_1, t_2, \ldots$ ) which are summarized as  $t_*$ .

The left column lists the so called "ICON-internal" variable names, which denotes those field names that are provided as the string argument name to the subroutine calls CALL add_var(...) and CALL add_ref(...) inside the ICON source code. These subroutine calls have the purpose to register new variables, to allocate the necessary memory, and to set the meta-data for these variables.

Variable Name	GRIB2 Name	Description
acdnc	ndcloud	Cloud droplet number concentration
adrag_u_grid		Zonal resolved surface stress mean since model start
adrag_v_grid		Meridional resolved surface stress mean since model start
aer_bc	aer_bc	Black carbon aerosol
aer_du	aer_dust	Total soil dust aerosol
aer_or	aer_org	Organic aerosol
aer_ss	aer_ss	Sea salt aerosol
aer_su	aer_so4	Total sulfate aerosol
aercl_bc		Black carbon aerosol climatology
aercl_du		Total soil dust aerosol climatology
aercl_or		Organic aerosol climatology
aercl_ss		Sea salt aerosol climatology
aercl_su		Total sulfate aerosol climatology
alb_dif	alb_dif	Shortwave albedo for diffuse radiation
alb_si	alb_seaice	Sea ice albedo (diffuse)
albdif_t_*	alb_rad	Tile-based shortwave albedo for diffusive radiation
albdif	alb_rad	Shortwave albedo for diffuse radiation
albni_dif	alb_ni	Near IR albedo for diffuse radiation
albnirdif_t_*	alb_ni	Tile-based near IR albedo for diffuse radiation
albnirdif	alb_ni	Near IR albedo for diffuse radiation
albnirdir	alnip	Near IR albedo for direct radiation
albuv_dif	alb_uv	UV visible albedo for diffuse radiation
albvisdif_t_*	alb_uv	Tile-based UV visible albedo for diffusive radiation
albvisdif	alb_uv	UV visible albedo for diffuse radiation
albvisdir	aluvp	UV visible albedo for direct radiation
alhfl_bs	alhfl_bs	Latent heat flux from bare mean since model start
alhfl_pl	alhfl_pl	Latent heat flux from plantmean since model start
alhfl_s	alhfl_s	surface latent heat flux mean since model start

Continued on next page

¹The Table B.1 in this Appendix is based on the revision state 3c3fdf4a (2022-08-08).

Variable Name	GRIB2 Name	Description
aqhfl_s	evapt	surface moisture flux mean since model start
ashfl_s	ashfl_s	Surface sensible heat flux mean since model start
asob_s	asob_s	Surface net solar radiation mean since model start
asob_t	_ asob_t	TOA net solar radiation mean since model start
asobclr_s	_ asob_s_cs	Clear-sky surface net solar radiation mean since model
-		start
asod_s	asod_s	Surface down solar rad. mean since model start
asod_t	asod_t	Top down solar radiation mean since model start
asodifd_s	aswdifd_s	Surface down solar diff. rad. mean since model start
asodifu_s	aswdifu_s	Surface up solar diff. rad. mean since model start
asodird_s	aswdir_s	Surface down solar direct rad.mean since model start
asou_t	uswrf	Top up solar radiation mean since model start
astr_u_sso	lgws	Zonal sso surface stress mean since model start
astr_v_sso	mgws	Meridional sso surface stress mean since model start
aswflx_par_sfc	apab_s	Downward PAR flux mean since model start
athb_s	athb_s	Surface net thermal radiation mean since model start
athb_t	athb_t	TOA net thermal radiation mean since model start
athbclr_s	athb_s_cs	Clear-sky surface net thermal radiation mean since
		model start
athd_s	athd_s	Surface down thermal radiationmean since model start
athu_s	athu_s	Surface up thermal radiation mean since model start
aumfl_s	aumfl_s	U-momentum flux flux at sumean since model start
avmfl_s	avmfl_s	V-momentum flux flux at sumean since model start
bdy_halo_c_blk		Block lists for halo points belonging to the nest
		boundary region
bdy_halo_c_idx		Index lists for halo points belonging to the nest
c t lk	c t lk	Shape factor (temp. profile in lake thermocline)
cape ml	cape ml	Cape of mean surface layer parcel
cape	cape con	Conv avail pot energy
cin ml	cin ml	Convective inhibition of mean surface laver parcel
clch	clch	High level clouds
clcl	clcl	Low level clouds
clcm	clcm	Mid level clouds
clct_mod	clct_mod	Modified total cloud cover for media
clct	clct	Total cloud cover
clc	clc	Cloud cover
cldepth	cldepth	Modified cloud depth for media
cli_m	qi	Specific cloud ice content (time mean)
cloud_num		Cloud droplet number concentration
clw_m	qc	Specific cloud water content (time mean)
con_gust		Convective contribution to wind gust
condhf_ice		Conductive heat flux at sea-ice bottom
cosmu0	uvcossza	Cosine of solar zenith angle
ddqz_z_full_e	рр	Metrics functional determinant (edge)
ddqz_z_full		Metrics functional determinant
ddqz_z_half		Metrics functional determinant
ddt_adv_q*		Advective tracer tendency

## Table B.1 – Continued from previous page

Continued on next page
Variable Name	GRIB2 Name	Description
ddt_exner_phy		Exner pressure physical tendency
ddt_gri_q≁	dpadt	Surface process tondoney
ddt ac conv	apsat	Convective tendency of cloud water mass density
ddt_qc_conv		Microphysics tendency of specific cloud water
ddt ac turb		Turbulence tendency of specific cloud water
ddt ag gsch		Microphysics tendency of graupel
ddt ai conv		Convective tendency of cloud ice mass density
ddt ai gscn		Microphysics tendency of specific cloud ice
ddt ai turb		Turbulence tendency of specific cloud ice
ddt ar conv		Convective tendency of rain mass density
ddt ar gscp		Microphysics tendency of rain
ddt gs conv		Convective tendency of snow mass density
ddt as gscp		Microphysics tendency of snow
ddt_qv_conv		Convective tendency of absolute humidity
ddt_qv_gscp		Microphysics tendency of specific humidity
ddt_qv_turb	qtendt	Turbulence tendency of specific humidity
ddt_temp_clcov	-	Sgs condensation temperature tendency
ddt_temp_drag	ttends	Sso + gwdrag temperature tendency
ddt_temp_dyn		Dynamical temperature tendency
ddt_temp_gscp		Microphysical temperature tendency
ddt_temp_pconv	dt_con	Convective temperature tendency
ddt_temp_radlw	thhr_rad	Long wave radiative temperature tendency
ddt_temp_radsw	sohr_rad	Short wave radiative temperature tendency
ddt_temp_turb	ttendts	Turbulence temperature tendency
ddt_tke_hsh	dtke_hsh	TKE tendency horizonzal shear production
ddt_tke_pconv	dtke_con	TKE tendency due to sub-grid scale convection
ddt_tke	tketens	Tendency of turbulent velocity scale
ddt_u_gwd	ewgd	GWD tendency of zonal wind
ddt_u_pconv	du_con	Convective tendency of zonal wind
ddt_u_sso	du_sso	Sso tendency of zonal wind
ddt_u_turb	utendts	Turbulence tendency of zonal wind
ddt_v_gwd	nsgd	GWD tendency of meridional wind
ddt_v_pconv	dv_con	Convective tendency of meridional wind
ddt_v_sso	dv_sso	Sso tendency of meridional wind
ddt_v_turb	vtenats	Advective Leonicity of meridional wind
dat_vn_apc_pc		predictor/corrector
ddt_vn_phy		Normal wind physical tendency
ddt_w_adv_pc		Advective vertical wind tendency, predictor/corrector
ddxn_z_full		Terrain slope in normal direction
ddxt_z_full		Terrain slope in tangential direction
deepatmo_t1ifc		Missing description
deepatmo_t1mc		Metrical modification factors for the deep-atmosphere equations
deepatmo_t2mc		Missing description
depth_lk	depth_lk	Lake depth
dgeopot_mc	fi	Geopotential difference between half levels

Variable Name	GRIB2 Name	Description
div_ic div dp_bs_lk dpres_mc drag_u_grid drag_v_grid dvn_ie_int dvn_ie_ubc dwdx dwdy dyn_gust	rdiv	Divergence at half levels Divergence Depth of thermally active layer of bot. sediments. Pressure thickness Zonal resolved surface stress Meridional resolved surface stress Normal velocity at parent interface level Normal velocity at child upper boundary Zonal gradient of vertical wind Meridional gradient of vertical wind Dynamical gust
eai		(evaporative) earth area index
emis_rad enhfac_diffu	emis_rad relhum	Longwave surface emissivity Background nabla2 diffusion coefficient for upper sponge layer
exner_dyn_incr		Exner dynamics increment
exner_pr	exner	Exner perturbation pressure
exner_ref_mc		Reference atmosphere field exner
exner	exner	Exner pressure
fac_ccqc		Factor for cloud cover - cloud water relationship
fetch_lk	fetch_lk	Wind fetch over lake
fis	fis	Geopotential (s)
for_d	for_d	Fraction of deciduous forest
fr_glac		Fraction glacier
fr_lake	fr_lake	Fraction lake
fr_land	Ir_land	Fraction land
fr_par_sfc_diff		Diffuse fraction of downward hear-infrared flux at surface flux at surface
fr_seaice	fr_ice	Fraction of sea ice
fr_vis_sfc_diff	-	Diffuse fraction of downward visible flux at surface
frac_t_*	fr_luc	Tile point area fraction list
freshsnow_t_*	freshsnw	Indicator for age of snow in top of snow layer
freshsnow	freshsnw	Weighted indicator for age of snow in top of snow layer
gamso_lk		Attenuation coefficient of lake water with respect to sol. rad.
<pre>geopot_agl_ifc</pre>	fi	Geopotential above groundlevel at cell center
geopot_agl	fi	Geopotential above groundlevel at cell center
geopot	fi	Geopotential at full level cell centre
graupel_gsp_rate	prg_gsp	Gridscale graupel rate
graupel_gsp	grau_gsp	Gridscale graupel
grf_tend_mflx		Normal mass flux tendency (grid refinement)
grf_tend_rho	s_oro_max	Density tendency (grid refinement)
gr1_tend_thv		Virtual potential temperature tendency (grid refinement)
gri_tend_vn		Normal wind tendency (grid refinement)
grr_tena_w		vertical whild tendency (grid rennement)

 $Continued \ on \ next \ page$ 

Table B.1 – Continued from previous $p$	age
-----------------------------------------	-----

Variable Name	GRIB2 Name	Description
gust10	vmax_10m	Gust at 10 m since end of previous full 01H since model start
gz0_t_*	z0	Tile-based roughness length times gravity
gz0	z0	Roughness length
h_b1_lk	h_b1_lk	Thickness of the upper layer of the sediments
h_ice	h_ice	Sea/lake-ice depth
h_ml_lk	h_ml_lk	Mixed-layer thickness
h_snow_lk		Depth of snow on lake ice
h_snow_si		Depth of snow on sea ice
h_snow_t_*	h_snow	Snow height
h_snow	h_snow	Weighted snow depth
hbas_con	hbas_con	Height of convective cloud base
hdef_ic		Deformation
hfl_q*		Horizontal tracer flux
hmo3		Height of O3 maximum (Pa)
htop_con	htop_con	Height of convective cloud top
htop_dc	htop_dc	Height of top of dry convection
hus_m	qv bassa 2	Specific humidity (time mean)
hzerocl	hzerocl	Height of 0 deg C level
<pre>ice_gsp_rate</pre>	iprate	Gridscale ice rate
ice_gsp	iprate	Gridscale ice
k400		Level index corresponding to the HAG of the 400hPa level
k650		Level index corresponding to the HAG of the $650\mathrm{hPa}$ level
k700		Level index corresponding to the HAG of the 700hPa level
k800		Level index corresponding to the HAG of the 800hPa level
k850		Level index corresponding to the HAG of the 850hPa level
k950		Level index corresponding to the HAG of the 950hPa level
ktype		Type of convection
l_pat		Effective length scale of circulation patterns
lai	lai	Leaf Area Index
lc_class_t_*	luc	Tile point land cover class
lhfl_bs_t_*		Tile-based latent heat flux from bare soil
lhfl_bs		Latent heat flux from bare soil
lhfl_pl_t_*		Tile-based latent heat flux from plants
lhfl_pl		Latent heat flux from plants
lhfl_s_t_*	lhfl_s	Tile-based surface latent heat flux
lhfl_s	lhfl_s	Surface latent heat flux
Liqfl_turb		Vertical turbulent liquid water flux
lsm_ctr_c	wmb	Ocean model land-sea-mask
⊥sm_switch		Land-sea-mask switched by ocean
IW_emiss	emis_rad	Longwave surface emissivity

Variable Name	GRIB2 Name	Description
<pre>lwflx_dn_clr lwflx_dn lwflx_up_clr lwflx_up lwflxall</pre>	nlwrf	Longwave downward clear-sky flux Longwave downward flux Longwave upward clear-sky flux Longwave upward flux Longwave net flux
<pre>mask_mtnpoints_g mask_mtnpoints mass_fl_e_sv mass_fl_e mflx_ic_ubc</pre>		Mask field for mountain points Mask field for mountain points Storage field for horizontal mass flux at edges Horizontal mass flux at edges Mass flux and tendency at child upper boundary
ndvi_max		NDVI yearly maximum
ndviratio	ndviratio	(monthly) proportion of actual value/maximum NDVI (at init time)
o3	o3	Ozone mixing ratio
omega_z	relv	Vertical vorticity
omega	omega	Vertical velocity
pat_len		Length scale of sub-grid scale roughness elements
pfull_m	р	Pressure at full level (time mean)
phalf_m	р	Pressure at half level (time mean)
plcov_t_*	plcov	Plant covering degree in the vegetation phase
plcov	plcov	Plant covering degree in the vegetation phase
prec_con_d		Convective precip since end of previous full 01H since model start
prec_con_rate_avg	cprat	Convective precip rate, time average
prec_con	prec_con	Convective precip
prec_gsp_d		Gridscale precip since end of previous full 01H since model start
<pre>prec_gsp_rate_avg</pre>	pr_gsp	Gridscale precip rate, time average
<pre>prec_gsp_rate</pre>	pr_gsp	Gridscale precipitation rate
prec_gsp	prec_gsp	Gridscale precip
pref_aerdis		Reference pressure used for vertical distribution of aerosol optical depths
pres_ifc	р	Pressure at half level
pres_msl	pmsl	Mean sea level pressure
pres_sfc	ps	Surface pressure
pres	р	Pressure
ps_m	ps	Surface pressure (time mean)
psl_m	pmsl	Mean sea level pressure (time mean)
q_int1		Q at parent interface level
q_int2		Q at parent interface level
q_int3		Q at parent interface level
q_int4		Q at parent interface level
q_int5		Q at parent interface level
q_int6		Q at parent interface level
q_ubc1		Q at child upper boundary
q_ubc2		Q at child upper boundary
q_ubc3		Q at child upper boundary

Variable Name	GRIB2 Name	Description
q_ubc4 q_ubc5		Q at child upper boundary Q at child upper boundary
q_ubc6		Q at child upper boundary
qc_sgs		Subgrid-scale cloud water
dcil's		Surface cloud water deposition flux due to diffusion
qc	qc	Specific cloud water content
qg	qg	Tile hand surface weithing from
qnil_s_t_*	evapt	Curfe as an electron form
qnii_s	evapt	Surface moisture flux
q111_s		Surface cloud ice deposition nux due to diffusion
dī	qı	Specific cloud ice content
qr	qr	Specific rain content
qs	qs	Specific snow content
qv_2m	qv_2m	Specific water vapor content in 2m
qv_s_t_*	qv_s	Specific humidity at the surface
qv_s	qv_s	Specific humidity at the surface
da	da	Specific humidity
r_bsmin		Minimal bare soil evaporation resistance
rain_con_rate_3d	prr_con	3d convective rain rate
rain_con_rate	prr_con	Convective rain rate
rain_con	rain_con	Convective rain
rain_gsp_rate	prr_gsp	Gridscale rain rate
rain_gsp	rain_gsp	Gridscale rain
rain_upd		Rain in updroughts
rayleigh_vn		Rayleigh damping coefficient for vn
rayleigh_w		Rayleigh damping coefficient for w
rcld		Standard deviation of the saturation deficit
rh_2m_land	relhum_2m_l	Relative humidity in 2m over land fraction
rh_2m	relhum_2m	Relative humidity in 2m
rho_ic_ubc		Density and tendency at child upper boundary
rho_ic	den	Density at half level
rho_m	den	Density (time mean)
rho_ref_mc		Reference atmosphere field density
rho_ref_me		Reference atmosphere field density
rho_snow_t_*	rho_snow	Snow density
rho_snow	rho_snow	Weighted snow density
rho	den	Density
rlamh_fac_t_*		Scaling factor for rlam_heat
rootdp	rootdp	Root depth of vegetation
rsmin	rsmin	Minimal stomata resistence
rstom	rstom	Stomatal resistance
runoff_g_t_*	watr	Soil water runoff
runoff_g	runoff_g	Weighted soil water runoff
runoff_s_t_*	watr	Surface water runoff
runoff_s	runoff_s	Weighted surface water runoff
sai		Surface area index
scalfac_dd3d		Scaling factor for 3D divergence damping terms

B. Output Variables

 $Continued \ on \ next \ page$ 

Variable Name	GRIB2 Name	Description
sfcfric_fac		Tuning factor for surface friction
shfl_s_t_*	shfl_s	Tile-based surface sensible heat flux
shfl_s	shfl_s	Surface sensible heat flux
skinc	skc	Skin conductivity
<pre>slope_angle</pre>		Slpe angle
slope_azimuth		Slpe azimuth
<pre>snow_con_rate_3d</pre>	prs_con	3d convective snow rate
<pre>snow_con_rate</pre>	prs_con	Convective snow rate
snow_con	snow_con	Convective snow
<pre>snow_gsp_rate</pre>	prs_gsp	Gridscale snow rate
snow_gsp	snow_gsp	Gridscale snow
<pre>snowfrac_lc_t_*</pre>	snowc	Tile-based snow-cover fraction
<pre>snowfrac_lcu_t_*</pre>		Tile-based snow-cover fraction
<pre>snowfrac_lc</pre>	snowc	Snow-cover fraction
<pre>snowfrac_t_*</pre>		Local tile-based snow-cover fraction
snowfrac		Snow-cover fraction
snowlmt	snowlmt	Height of snow fall limit above MSL
sob_s_t_*	sobs_rad	Tile-based shortwave net flux at surface
sob_s	sobs_rad	Shortwave net flux at surface
sob_t	sobt_rad	Shortwave net flux at TOA
sobclr_s	sobs_rad_cs	Net shortwave clear-sky flux at surface
sod_t	<pre>sodt_rad</pre>	Downward shortwave flux at TOA
sodifd_s	swdifds_rad	Shortwave diffuse downward flux at surface
soiltyp	soiltyp	Soil type
sou_s	swdifus_rad	Shortwave upward flux at surface
sou_t	uswrf	Shortwave upward flux at TOA
sp_10m	sp_10m	Wind speed in 10m
sso_gamma	sso_gamma	Anisotropy of sub-gridscale orography
sso_sigma	sso_sigma	Slope of sub-gridscale orography
sso_stdh_raw		Standard deviation of sub-grid scale orography
sso_stdh	sso_stdh	Standard deviation of sub-grid scale orography
sso_theta	sso_theta	Angle of sub-gridscale orography
str_u_sso	lgws	Zonal sso surface stress
str_v_sso	mgws	Meridional sso surface stress
swflx_dn_clr		Shortave downward clear-sky flux
swflx_dn		Shortwave downward flux
swflx_nir_sfc		Downward near-infrared flux at surface
swflx_par_sfc	pabs_rad	Downward photosynthetically active flux at surface
swflx_up_clr		Shortave upward clear-sky flux
swflx_up		Shortwave upward flux
swflx_vis_sfc		Downward visible flux at surface
t_2m_land	t_2m_1	Temperature in 2m over land fraction
t_2m	t_2m	Temperature in 2m
t_b1_lk	t_b1_lk	Temperature at the bottom of the upper layer of the sediments
t_bot_lk	t_bot_lk	Temperature at the water-bottom sediment interface
t_bs_lk		Clim. temp. at bottom of thermally active layer of sediments

 $Continued \ on \ next \ page$ 

Variable Name	GRIB2 Name	Description
t_cl	t_2m_cl	CRU near surface temperature climatology
t_g_t_*	t_g	Weighted surface temperature
t_g	t_g	Weighted surface temperature
t_ice	t_ice	Sea/lake-ice temperature
t_mnw_lk	t_mnw_lk	Mean temperature of the water column
t_s_t_*	t_s	Temperature of ground surface
t_seasfc	t_sea	Sea surface temperature
t_sk_t_*	skt	Skin temperature
t_sk	skt	Skin temperature
t_snow_lk		Temperature of snow on lake ice
t_snow_si		Temperature of snow on sea ice
t_snow_t_*	t_snow	Temperature of the snow-surface
t_snow	t_snow	Weighted temperature of the snow-surface
t_so_t_*	t_so	Soil temperature (main level)
t_so	- t_so	Weighted soil temperature (main level)
t_s	t_s	Weighted temperature of ground surface
t_tilemax_inst_2m	- t_2m	Instantaneous temperature in 2m, maximum over tiles
t_tilemin_inst_2m	- t_2m	Instantaneous temperature in 2m, minimum over tiles
t_wml_lk	_ t_wml_lk	Mixed-layer temperature
ta m	 t	Temperature
tai		Transpiration area index
tch t *	tch	Tile-based turbulent transfer coefficients for heat
tch	tch	Turbulent transfer coefficients for heat
tcm t *	tcm	Tile-based turbulent transfer coefficients for momentum
tcm	tcm	Turbulent transfer coefficients for momentum
td 2m land	td 2m l	Dew-point in 2m over land fraction
td 2m	td 2m	Dew-point in 2m
temp ifc	t	Temperature at half level
tempv	vtmp	Virtual temperature
temp	t	Temperature
tetfl turb		Vertical turbulent theta flux
tfh		Factor of laminar transfer of scalars
t.fm		Factor of laminar transfer of momentum
tfv t *	nsurs	Tile-based laminar reduction factor for evaporation
tfv	IIGWI D	Laminar reduction factor for evaporation
thhst *	thbs rad	Tile-based longwave net flux at surface
thb s	thbs rad	Longwave net flux at surface
thb t	thbt rad	Thermal net flux at TOA
thbclr s	thbs rad cs	Net longwave clear-sky flux at surface
theta ref ic	unop_rud_co	Reference atmosphere field theta
theta_ref_mc		Reference atmosphere field theta
theta_ref_me		Reference atmosphere field theta
theta v ic ubc		Potential temperature and tendency at child upper
		boundary
theta_v_ic	theta_v	Virtual potential temperature at half levels
theta_v	theta_v	Virtual potential temperature
thu_s	thus_rad	Longwave upward flux at surface
tke	tke	Turbulent kinetic energy

Variable Name	GRIB2 Name	Description
tkr_t_* tkred_sfc		Tile-based turbulent reference surface diffusion coefficient Reduction factor for minimum diffusion coefficients
	+ levels	turbulent diffusion coefficients for heat
	t k v II	turbulent diffusion coefficients for momentum
tmox Om	tror Om	Max 2m temperature
tmax_2m	tmax_2m	Min 2m temperature
tenegrophy a	baurf	Competition height of the conthe surface shows see level
topography_c	lisuri	Total procinging and of provide full 01H gines model
tot_prec_a	tot_prec_d	start
tot_prec_rate_avg	tot_pr	Total precip rate, time average
tot_prec_rate	tot_pr	Total precipitation rate
tot_prec	tot_prec	Total precip
tot_qc_dia	qc_dia	Total specific cloud water content (diagnostic)
tot_qi_dia	qi_dia	Total specific cloud ice content (diagnostic)
tot_qv_dia	qv_dia	Total specific humidity (diagnostic)
tqc_dia	tqc_dia	Total column integrated cloud water (diagnostic)
tqc	tqc	Total column integrated cloud water
tqg	tqg	Total column integrated graupel
tqi_dia	tqi_dia	Total column integrated cloud ice (diagnostic)
tqi	tqi	Total column integrated cloud ice
tqr	tqr	Total column integrated rain
tqs	tqs	Total column integrated snow
tqv_dia	tqv_dia	Total column integrated water vapour (diagnostic)
tqv	tqv	Total column integrated water vapour
trsolall		Shortwave net tranmissivity
tsic_rei		Reference surface temperature
tsictrad		Surface temperature at trad
tvh_t_*		The-based turbulent transfer velocity for heat
tvh		Turbulent transfer velocity for heat
tvm_t_* tvm		The-based turbulent transfer velocity for momentum Turbulent transfer velocity for momentum
u_10m_t_*	u_10m	Tile-based zonal wind in 2m
u_10m	u_10m	Zonal wind in 10m
ua_m	u	Zonal wind (time mean)
umfl_s_t_*	umfl_s	U-momentum flux at the surface
umfl_s	umfl_s	U-momentum flux at the surface
u	u	Zonal wind
v_10m_t_*	v_10m	Tile-based meridional wind in 2m
v_10m	v_10m	Meridional wind in 10m
va_m	v	Meridional wind (time mean)
vapfl_turb		Vertical turbulent water vapour flux
vfl_q*		Vertical tracer flux
vio3	tozne	Vertically integrated ozone amount
vmfl_s_t_*	vmfl_s	V-momentum flux at the surface
vmfl_s	vmfl_s	V-momentum flux at the surface
vn_ie	vn	Normal wind at half level

Variable Name	GRIB2 Name	Description
vn	vn	Velocity normal to edge
vor	relv	Vorticity
vt	vt	Tangential-component of wind
vwind_expl_wgt		Explicit weight in vertical wind solver
vwind_impl_wgt		Implicit weight in vertical wind solver
v	v	Meridional wind
w_concorr_c		Contravariant vertical correction
w_i_t_*	w_i	Weighted water content of interception water
w_i	w_i	Weighted water content of interception water
w_snow_t_*	w_snow	Water equivalent of snow
w_snow	w_snow	Weighted water equivalent of snow
w_so_ice_t_*	w_so_ice	Ice content
w_so_ice	w_so_ice	Ice content
w_so_t_*	w_so	Total water content (ice + liquid water)
w_so	w_so	Total water content (ice + liquid water)
w_ubc		Vertical velocity and tendency at child upper boundary
wa_m	W	Vertical velocity (time mean)
wap_m	omega	Vertical velocity (time mean)
ww	WW	Significant weather
W	W	Vertical velocity
z_ifc	hhl	Geometric height at half level center
z_mc	h	Geometric height at full level center
zd_blklist		Missing description
zd_diffcoef		Missing description
zd_e2cell		Missing description
zd_edgeblk		Missing description
zd_edgeidx		Missing description
zd_geofac		Missing description
zd_indlist		Missing description
zd_intcoef		Missing description
zd_vertidx		Missing description

Table B.1 – Continued from previous page

## List of ICON Variable Groups

The "group: " keyword for the namelist parameters ml_varlist, hl_varlist, pl_varlist (namelist output_nml) can be used to activate a set of common variables for output at once. The following lists contain the variables for each of these groups (empty groups and groups with only a single entry are omitted).

#### Group ADDITIONAL_PRECIP_VARS

cape, clct, clct_mod, prec_con_rate_avg, prec_gsp_rate_avg, tqc_dia, tqi_dia, tqv_dia

## Group ATMO_DERIVED_VARS

 ${\tt div},\,{\tt omega},\,{\tt omega}_z,\,{\tt vor}$ 

#### Group ATMO_ML_VARS

cli_m, clw_m, hus_m, pres, qc, qg, qi, qr, qs, qv, temp, tke, u, v, w

#### Group ATMO_PL_VARS

cli_m, clw_m, hus_m, qc, qg, qi, qr, qs, qv, temp, tke, u, v, w

#### Group ATMO_TIMEMEAN

cli_m, clw_m, hus_m, pfull_m, phalf_m, ps_m, psl_m, rho_m, ta_m, ua_m, va_m, wa_m, wap_m

#### Group ATMO_ZL_VARS

cli_m, clw_m, hus_m, pres, qc, qg, qi, qr, qs, qv, temp, tke, u, v, w

#### Group CLOUD_DIAG

clc, qc_sgs, tot_qc_dia, tot_qi_dia, tot_qv_dia

#### Group DWD_FG_ATM_VARS

pres, pres_sfc, qc, qg, qi, qr, qs, qv, rho, t_2m, td_2m, temp, theta_v, tke, u, u_10m, v, v_10m, vn, w, z_ifc

#### Group DWD_FG_SFC_VARS

alb_si, c_t_lk, fr_land, fr_seaice, freshsnow, gz0, h_ice, h_ml_lk, h_snow, qv_s, rho_snow, snowfrac_lc, t_bot_lk, t_g, t_ice, t_mnw_lk, t_seasfc, t_sk, t_snow, t_so, t_wml_lk, w_i, w_snow, w_so, w_so_ice

#### Group DWD_FG_SFC_VARS_T

freshsnow_t_*, h_snow_t_*, qv_s_t_*, rho_snow_t_*, snowfrac_lc_t_*, t_g_t_*, t_sk_t_*, t_snow_t_*, t_so_t_*, w_i_t_*, w_snow_t_*, w_so_ice_t_*, w_so_t_*

#### Group ICON_LBC_VARS

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

#### Group LAND_TILE_VARS

h_snow_t_*, qv_s_t_*, rho_snow_t_*, snowfrac_lc_t_*, snowfrac_t_*, t_g_t_*, t_s_t_*, t_sk_t_*, t_snow_t_*, t_so_t_*, w_i_t_*, w_snow_t_*, w_so_ice_t_*, w_so_t_*

#### Group LAND_VARS

qv_s, rho_snow, snowfrac, snowfrac_lc, t_g, t_snow, t_so, w_i, w_snow, w_so, w_so_ice

#### Group LATBC_PREFETCH_VARS

pres, pres_sfc, qc, qg, qi, qr, qs, qv, rho, temp, theta_v, u, v, vn, w, z_ifc

#### Group MODE_COMBINED_IN

alb_si, fr_seaice, freshsnow, h_ice, h_snow, qv_s, rho_snow, t_g, t_ice, t_sk, t_snow, t_so, w_i, w_snow, w_so

#### Group MODE_COSMO_IN

alb_si, freshsnow, h_ice, qv_s, rho_snow, t_g, t_ice, t_snow, t_so, w_i, w_snow, w_so

#### Group MODE_DWD_ANA_IN

fr_seaice, freshsnow, h_ice, h_snow, pres, qv, t_ice, t_seasfc, t_snow, t_so, temp, u,
v, w_so

#### $Group \; \texttt{MODE_DWD_FG_IN}$

alb_si, c_t_lk, gz0, h_ml_lk, qc, qg, qi, qr, qs, qv_s, rho, rho_snow, t_bot_lk, t_g, t_mnw_lk, t_sk, t_so, t_wml_lk, theta_v, tke, vn, w, w_i, w_snow, w_so_ice, z_ifc

#### Group MODE_IAU_ANAATM_IN

pres, qc, qg, qi, qr, qs, qv, temp, u, v

#### Group MODE_IAU_ANA_IN

fr_seaice, freshsnow, h_snow, pres, qc, qg, qi, qr, qs, qv, t_seasfc, t_so, temp, u, v,
w_so

#### Group MODE_IAU_FG_IN

alb_si, c_t_lk, freshsnow, gz0, h_ice, h_ml_lk, h_snow, qc, qg, qi, qr, qs, qv, qv_s, rho, rho_snow, snowfrac_lc, t_bot_lk, t_g, t_ice, t_mnw_lk, t_sk, t_snow, t_so, t_wml_lk, theta_v, tke, vn, w, w_i, w_so, w_so_ice

#### Group MODE_IAU_OLD_ANA_IN

fr_seaice, freshsnow, h_snow, pres, qv, rho_snow, t_seasfc, t_so, temp, u, v, w_snow,
w_so

#### Group MODE_IAU_OLD_FG_IN

alb_si, c_t_lk, gz0, h_ice, h_ml_lk, qc, qg, qi, qr, qs, qv, qv_s, rho, t_bot_lk, t_g, t_ice, t_mnw_lk, t_snow, t_so, t_wml_lk, theta_v, tke, vn, w, w_i, w_so, w_so_ice

#### Group MODE_INIANA

alb_si, c_t_lk, fr_land, fr_seaice, freshsnow, gz0, h_ice, h_ml_lk, h_snow, pres, qc, qg, qi, qr, qs, qv, qv_s, rho_snow, t_bot_lk, t_g, t_ice, t_mnw_lk, t_snow, t_so, t_wml_lk, temp, tke, u, v, w, w_i, w_snow, w_so_ice, z_ifc

#### Group NH_PROG_VARS

exner, rho, theta_v, vn

#### Group PBL_VARS

alhfl_s, aqhfl_s, ashfl_s, gust10, lhfl_bs, lhfl_s, qhfl_s, qv_2m, shfl_s, t_2m, t_2m_land, tch, tcm, td_2m, td_2m_land, tkr, tkvh, tkvm, tvh, tvm, u_10m, v_10m

#### Group PHYS_TENDENCIES

ddt_qc_conv, ddt_qc_gscp, ddt_qc_turb, ddt_qg_gscp, ddt_qi_conv, ddt_qi_gscp, ddt_qi_turb, ddt_qr_conv, ddt_qr_gscp, ddt_qs_conv, ddt_qs_gscp, ddt_qv_conv, ddt_qv_gscp, ddt_qv_turb, ddt_temp_clcov, ddt_temp_drag, ddt_temp_gscp, ddt_temp_pconv, ddt_temp_radlw, ddt_temp_radsw, ddt_temp_turb, ddt_tke, ddt_tke_hsh, ddt_tke_pconv, ddt_u_gwd, ddt_u_pconv, ddt_u_sso, ddt_u_turb, ddt_v_gwd, ddt_v_pconv, ddt_v_sso, ddt_v_turb

#### Group PRECIP_VARS

#### Group PROG_TIMEMEAN

pfull_m, phalf_m, ps_m, psl_m, rho_m, ta_m, ua_m, va_m, wa_m, wap_m

#### Group RAD_VARS

albdif, albnirdif, albvisdif, asob_s, asob_t, asod_t, asodifd_s, asodifu_s, asodird_s, asou_t, aswflx_par_sfc, athb_s, athb_t, athd_s, athu_s, cosmu0, fr_nir_sfc_diff, fr_par_sfc_diff, fr_vis_sfc_diff, sob_s, sob_s_t_*, sob_t, sod_t, sodifd_s, sou_s, sou_t, swflx_nir_sfc, swflx_par_sfc, swflx_vis_sfc, thb_s, thb_s_t_*, thb_t, thu_s ICON Model Tutorial

## $\mathbf{Group} \ \mathtt{SNOW}_\mathtt{VARS}$

rho_snow, t_snow

# Group TRACER_TIMEMEAN cli_m, clw_m, hus_m

# **Bibliography**

- Asensio, H., and M. Messmer, 2014: External Parameters for Numerical Weather Prediction and Climate Application: EXTPAR v2.0.2 User and Implementation Guide. Consortium for Small-scale Modeling (COSMO), URL http://www.cosmo-model.org/content/ model/modules/Extpar_201408_user_and_implementation_manual.pdf.
- Avissar, R., and R. Pielke, 1989: A parameterization of heterogeneous land surfaces for atmospheric numerical models and its impact on regional meteorology. *Mon. Weather Rev.*, 117, 2113–2136.
- Baines, P., and T. Palmer, 1990: Rationale for a new physically based parameterization of sub-grid scale orographic effects. Tech. Rep. 169, European Centre for Medium-Range Weather Forecasts, 11 pp. URL http://www.ecmwf.int.
- Baldauf, M., A. Seifert, J. Förstner, D. Majewski, M. Raschendorfer, and T. Reinhardt, 2011: Operational Convective-Scale Numerical Weather Prediction with the COSMO Model: Description and Sensitivities. *Mon. Weather Rev.*, **139** (12), 3887–3905, doi: 10.1175/MWR-D-10-05013.1.
- Barker, H. W., G. L. Stephens, P. T. Partain, and Coauthors, 2003: Assessing 1D atmospheric solar radiative transfer models: Interpretation and handling of unresolved clouds. J. Clim., 16 (16), 2676–2699, doi:10.1175/1520-0442(2003)016<2676:ADASRT>2.0. CO;2.
- Bechtold, P., 2017: Atmospheric moist convection. *Meteorological Training Course Lecture Series*, ECMWF, 1–78, URL https://www.ecmwf.int/sites/default/files/elibrary/2017/16953-atmospheric-moist-convection.pdf.
- Bechtold, P., M. Köhler, T. Jung, F. Doblas-Reyes, M. Leutbecher, M. J. Rodwell, F. Vitart, and G. Balsamo, 2008: Advances in simulating atmospheric variability with the ECMWF model: From synoptic to decadal time-scales. Q. J. R. Meteorol. Soc., 134 (634), 1337–1351, doi:10.1002/qj.289.
- Bechtold, P., N. Semane, P. Lopez, J.-P. Chaboureau, A. Beljaars, and N. Bormann, 2014: Representing equilibrium and nonequilibrium convection in large-scale models. J. Atmos. Sci., 71 (2), 734–753, doi:10.1175/JAS-D-13-0163.1.
- Blackadar, A. K., 1962: The vertical distribution of wind and turbulent exchange in a neutral atmosphere. J. Geophys. Res., 67, 3095–3102.
- Bloom, S. C., L. L. Takacs, A. M. D. Silva, and D. Ledvina, 1996: Data assimilation using incremental analysis updates. *Mon. Weather Rev.*, **124**, 1256–1270.

- Choulga, M., E. Kourzeneva, E. Zakharova, and A. Doganovsky, 2014: Estimation of the mean depth of boreal lakes for use in numerical weather prediction and climate modelling. *Tellus A*, 66, 21 295, doi:10.3402/tellusa.v66.21295.
- Colella, P., and P. R. Woodward, 1984: The piecewise parabolic method (ppm) for gasdynamical simulations. J. Comput. Phys., 54, 174–201.
- Crueger, T., and Coauthors, 2018: ICON-A, the atmosphere component of the ICON earth system model: II. model evaluation. J. Adv. Model Earth Sy., 10 (7), 1638–1662, doi:10.1029/2017MS001233.
- Davies, H., 1976: A lateral boundary formulation for multi-level prediction models. Q. J. R. Meteorol. Soc., 102, 405–418, doi:10.1002/qj.49710243210.
- Davies, H., 1983: Limitations of some common lateral boundary schemes used in regional NWP models. Mon. Weather Rev., 111, 1002–1012, doi:10.1175/1520-0493(1983) 111<1002:LOSCLB>2.0.CO;2.
- Dipankar, A., B. Stevens, R. Heinze, C. Moseley, G. Zängl, M. Giorgetta, and S. Brdar, 2015: Large eddy simulation using the general circulation model ICON. J. Adv. Model Earth Sy., 7 (3), 963–986, doi:10.1002/2015MS000431.
- Doms, G., and M. Baldauf, 2018: A Description of the Nonhydrostatic Regional COSMO-Model. Part I: Dynamics and Numerics. Consortium for Small-Scale Modelling, URL http://www.cosmo-model.org/public/documentation.htm.
- Doms, G., U. Schättler, and J.-P. Schulz, 2003: *Kurze Beschreibung des Lokal-Modells LM und seiner Datenbanken auf dem Datenserver des DWD*. Deutscher Wetterdienst (DWD), URL https://www.dwd.de/EN/ourservices/reports_on_icon/reports_ on_icon.html.
- Doms, G., and Coauthors, 2011: A Description of the Nonhydrostatic Regional COSMO Model. Part II: Physical Parameterization. Consortium for Small-Scale Modelling, URL http://www.cosmo-model.org.
- Easter, R. C., 1993: Two modified versions of Bott's positive-definite numerical advection scheme. Mon. Weather Rev., 121, 297–304.
- ECMWF, 2017: PART IV: PHYSICAL PROCESSES, chap. Convection, 75–95. IFS Documentation, ECMWF.
- ECMWF, 2018a: *PART IV: PHYSICAL PROCESSES*, chap. Subgrid-scale orographic drag, 59–67. IFS Documentation, ECMWF.
- ECMWF, 2018b: *PART IV: PHYSICAL PROCESSES*, chap. Non-ogographic gravity wave drag, 69–74. IFS Documentation, ECMWF.
- Ern, M., P. Preusse, and C. D. Warner, 2006: Some experimental constraints for spectral parameters used in the warner and mcintyre gravity wave parameterization scheme. *Atmos. Chem. Phys.*, 6 (12), 4361–4381, doi:10.5194/acp-6-4361-2006.
- Gal-Chen, T., and R. Somerville, 1975: On the use of a coordinate transformation for the solution of the Navier-Stokes equations. J. Comput. Phys., 17, 209–228.

- Gallus, W. A., and M. Ranĉić, 1996: A non-hydrostatic version of the NMC's regional Eta model. Q. J. R. Meteorol. Soc., 122, 495–513.
- Gassmann, A., 2013: A global hexagonal C-grid non-hydrostatic dynamical core (ICON-IAP) designed for energetic consistency. Q. J. R. Meteorol. Soc., 139 (670), 152–175, doi:10.1002/qj.1960.
- Gassmann, A., and H.-J. Herzog, 2008: Towards a consistent numerical compressible non-hydrostatic model using generalized Hamiltonian tools. Q. J. R. Meteorol. Soc., 134 (635), 1597–1613.
- Giorgetta, M., and Coauthors, 2018: ICON-A, the Atmosphere Component of the ICON Earth System Model: I. Model Description. J. Adv. Model Earth Sy., 10 (7), 1613–1637, doi:10.1029/2017MS001242.
- Giorgetta, M. A., and Coauthors, 2022: The icon-a model for direct qbo simulations on gpus (version icon-cscs:baf28a514). Geosci. Model Dev., 15 (18), 6985–7016, doi:10. 5194/gmd-15-6985-2022.
- GLOBE-Task-Team, 1999: The Global Land One-kilometer Base Elevation (GLOBE) Digital Elevation Model, version 1.0. Tech. rep., National Oceanic and Atmospheric Administration. URL http://www.ngdc.noaa.gov/mgg/topo/globe.html.
- Grell, G. A., J. Dudhia, and D. Stauffer, 1994: A description of the fifth-generation Penn State/NCAR Mesoscale Model (MM5). Tech. Rep. NCAR/TN-398+STR, University Corporation for Atmospheric Research. doi:10.5065/D60Z716B.
- Guerra, J. E., and P. A. Ullrich, 2016: A high-order staggered finite-element vertical discretization for non-hydrostatic atmospheric models. *Geosci. Model Dev.*, 9 (5), 2007– 2029, doi:10.5194/gmd-9-2007-2016, URL https://gmd.copernicus.org/articles/9/2007/ 2016/.
- Harris, L. M., and P. H. Lauritzen, 2010: A flux-form version of the conservative semilagrangian multi-tracer transport scheme (CSLAM) on the cubed sphere grid. J. Comput. Phys., 230, 1215–1237.
- Heinze, R., and Coauthors, 2017: Large-eddy simulations over Germany using ICON: a comprehensive evaluation. Q. J. R. Meteorol. Soc., 143 (702), 69–100.
- Heise, E., B. Ritter, and R. Schrodin, 2006: Operational implementation of the multilayer soil model. COSMO Technical Reports No. 9, Consortium for Small-Scale Modelling, 19pp, URL http://www.cosmo-model.org.
- Hesselberg, A., 1925: Die Gesetze der ausgeglichenen atmosphärischen Bewegungen. Beitr. Phys. Atmos., 12, 141–160.
- Heymsfield, A. J., and L. J. Donner, 1990: A scheme for parameterizing ice-cloud water content in general circulation models. J. Atmos. Sci., 47 (15), 1865–1877.
- Hogan, R. J., and A. Bozzo, 2018: A flexible and efficient radiation scheme for the ecmwf model. J. Adv. Model Earth Sy., 10 (8), 1990–2008.

- Hogan, R. J., and A. J. Illingworth, 2000: Deriving cloud overlap statistics from radar. Q. J. R. Meteorol. Soc., 126 (569), 2903–2909.
- Hohenegger, C., and Coauthors, 2023: ICON-Sapphire: simulating the components of the Earth system and their interactions at kilometer and subkilometer scales. *Geosci. Model Dev.*, 16 (2), 779–811, doi:10.5194/gmd-16-779-2023.
- Hunt, B. R., E. Kostelich, and I. Szunyogh, 2007: Efficient data assimilation for spatiotemporal chaos: A Local Ensemble Transform Kalman Filter. *Physica D*, 230, 112–126.
- Jablonowski, C., P. Lauritzen, R. Nair, and M. Taylor, 2008: Idealized test cases for the dynamical cores of Atmospheric General Circulation Models: A proposal for the NCAR ASP 2008 summer colloquium. National Center for Atmospheric Research (NCAR).
- Jablonowski, C., and D. L. Williamson, 2006: A baroclinic instability test case for atmospheric model dynamical cores. Q. J. R. Meteorol. Soc., 132, 2943–2975.
- Klemp, J., 2011: A terrain-following coordinate with smoothed coordinate surfaces. Mon. Weather Rev., 139, 2163–2169.
- Klemp, J., J. Dudhia, and A. Hassiotis, 2008: An upper gravity-wave absorbing layer for NWP applications. *Mon. Weather Rev.*, **136** (10), 3987–4004.
- Klemp, J. B., W. C. Skamarock, and J. Dudhia, 2007: Conservative split-explicit time integration methods for the compressible nonhydrostatic equations. *Mon. Weather Rev.*, 135 (8), 2897 – 2913.
- Köhler, M., M. Ahlgrimm, and A. Beljaars, 2011: Unified treatment of dry convective and stratocumulus-topped boundary layers in the ECMWF model. Q. J. R. Meteorol. Soc., 137 (654), 43–57.
- Kolmogorov, A. N., 1968: Local structure of turbulence in an incompressible viscous fluid at very high reynolds numbers. *Sov. Phys. Usp.*, **10** (6), 734.
- Korn, P., 2017: Formulation of an unstructured grid model for global ocean dynamics. J. Comput. Phys., 339 (C), 525–552, doi:10.1016/j.jcp.2017.03.009.
- Korn, P., and S. Danilov, 2017: Elementary dispersion analysis of some mimetic discretizations on triangular C-grids. J. Comput. Phys., 330, 156 – 172, doi:https://doi.org/10. 1016/j.jcp.2016.10.059.
- Korn, P., and Coauthors, 2022: ICON-O: The Ocean Component of the ICON Earth System Model–Global Simulation Characteristics and Local Telescoping Capability. J. Adv. Model Earth Sy., 14 (10), e2021MS002952, doi:10.1029/2021MS002952.
- Kourzeneva, E., 2010: External data for lake parameterization in Numerical Weather Prediction and climate modeling. *Boreal Env. Res.*, 15, 165–177.
- Kourzeneva, E., H. Asensio, E. Martin, and S. Faroux, 2012: Global gridded dataset of lake coverage and lake depth for use in numerical weather prediction and climate modelling. *Tellus A*, 64, 15640, doi:10.3402/tellusa.v64i0.15640.
- Lauritzen, P. H., C. Erath, and R. Mittal, 2011a: On simplifying 'incremental remap'-based transport schemes. J. Comput. Phys., 230, 7957–7963.

- Lauritzen, P. H., C. Jablonowski, M. A. Taylor, and R. D. Nair, 2011b: Numerical Techniques for Global Atmospheric Models. 1st ed., Springer, 556 pp.
- Lauritzen, P. H., R. D. Nair, and P. A. Ullrich, 2010: A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid. J. Comput. Phys., 229, 1401–1424.
- Leuenberger, D., M. Koller, and C. Schär, 2010: A generalization of the SLEVE vertical coordinate. Mon. Weather Rev., 138, 3683–3689.
- Lilly, D. K., 1962: On the numerical simulation of buoyant convection. *Tellus*, **14** (2), 148–172, doi:10.1111/j.2153-3490.1962.tb00128.x.
- Lin, S.-J., and R. B. Rood, 1996: Multidimensional flux-form semi-lagrangian transport schemes. Mon. Weather Rev., 124 (9), 2046–2070, doi:10.1175/1520-0493(1996) 124<2046:MFFSLT>2.0.CO;2.
- Lipscomb, W. H., and T. D. Ringler, 2005: An incremental remapping transport scheme on a spherical geodesic grid. *Mon. Weather Rev.*, **133**, 2335–2350.
- Lott, F., and M. J. Miller, 1997: A new subgrid-scale orographic drag parametrization: Its formulation and testing. Q. J. R. Meteorol. Soc., 123 (537), 101–127, doi:10.1002/qj. 49712353704.
- Matsuno, T., 1966: Numerical integrations of the primitive equations by a simulated backward difference method. J. Meteorolog. Soc. Jpn., 44 (1), 76–84, doi:10.2151/jmsj1965. 44.1_76.
- McLandress, C., and J. F. Scinocca, 2005: The GCM response to current parameterizations of nonorographic gravity wave drag. J. Atmos. Sci., 62 (7), 2394–2413, doi:10.1175/ JAS3483.1.
- Mellor, G. L., and T. Yamada, 1982: Development of a turbulence closure model for geophysical fluid problems. *Rev. Geophys.*, **20** (4), 851–875, doi:10.1029/RG020i004p00851.
- Mironov, D., E. Heise, E. Kourzeneva, B. Ritter, N. Schneider, and A. Terzhevik, 2010: Implementation of the lake parameterisation scheme FLake into the numerical weather prediction model COSMO. *Boreal Env. Res.*, 15, 218–230.
- Mironov, D., B. Ritter, J.-P. Schulz, M. Buchhold, M. Lange, and E. Machulskaya, 2012: Parameterisation of sea and lake ice in numerical weather prediction models of the German weather service. *Tellus A*, **64** (0), doi:10.3402/tellusa.v64i0.17330.
- Mironov, D. V., 2008: Parameterization of lakes in numerical weather prediction. Description of a lake model. COSMO Technical Report, Consortium for Small-Scale Modelling, 41 pp. URL http://www.cosmo-model.org.
- Miura, H., 2007: An upwind-biased conservative advection scheme for spherical hexagonalpentagonal grids. Mon. Weather Rev., 135, 4038–4044.
- Mlawer, E. J., S. J. Taubman, P. D. Brown, M. J. Iacono, and S. A. Clough, 1997: Radiative transfer for inhomogeneous atmospheres: RRTM, a validated correlated-k model for the longwave. J. Geophys. Res.: Atmos., 102 (D14), 16663–16682, doi:10.1029/97JD00237.

- Narcowich, F. J., and J. D. Ward, 1994: Generalized Hermite interpolation via matrixvalued conditionally positive definite functions. *Math. Comp.*, 63, 661–687, doi:10.1090/ S0025-5718-1994-1254147-6.
- Neggers, R. A. J., M. Köhler, and A. C. M. Beljaars, 2009: A dual mass flux framework for boundary layer convection. Part I: Transport. J. Atmos. Sci., 66 (6), 1465–1487, doi:10.1175/2008JAS2635.1.
- Orr, A., P. Bechtold, J. Scinocca, M. Ern, and M. Janiskova, 2010: Improved middle atmosphere climate and forecasts in the ECMWF model through a nonorographic gravity wave drag parameterization. J. Clim., 23 (22), 5905–5926, doi:10.1175/2010JCLI3490.1.
- Pincus, R., H. W. Barker, and J.-J. Morcrette, 2003: A fast, flexible, approximate technique for computing radiative transfer in inhomogeneous cloud fields. J. Geophys. Res.: Atmos., 108 (D13).
- Polavarapu, S., S. Ren, A. M. Clayton, D. Sankey, and Y. Rochon, 2004: On the relationship between incremental analysis updating and incremental digital filtering. *Mon. Weather Rev.*, 132, 2495–2502.
- Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, 2007: Numerical Recipes. 3rd ed., Cambridge University Press, 1235 pp.
- Prill, F., 2020: *DWD ICON Tools Documentation*. Deutscher Wetterdienst (DWD), dwd_icon_tools/doc/icontools_doc.pdf.
- Randall, D. A., 2017: A survey of time-differencing schemes for the oscillation and decay equations. An Introduction to Numerical Modelling of the Atmosphere, Colorado State University, 63–104, URL http://kiwi.atmos.colostate.edu/group/dave/at604pdf/ AT604_LaTeX_Book.pdf.
- Raschendorfer, M., 2001: The new turbulence parameterization of LM. COSMO News Letter No. 1, Consortium for Small-Scale Modelling, 89–97, URL http://www.cosmo-model. org.
- Rast, S., 2017: Using and programming ICON a first introduction. Max Planck Institute for Meteorology, URL https://code.mpimet.mpg.de/attachments/download/15898/ icon_lecture_2016.pdf, course at Hamburg University 2017.
- Reinert, D., 2020: The Tracer Transport Module Part I: A Mass Consistent Finite Volume Approach with Fractional Steps. Reports on icon, Deutscher Wetterdienst, 19 pp. doi:10.5676/DWD_pub/nwv/icon_005, URL https://www.dwd.de/DE/leistungen/ reports_on_icon/reports_on_icon.html.
- Reinert, D., 2021: The Tracer Transport Module Part II: Description and Validation of the Vertical Transport Operator. Reports on icon, Deutscher Wetterdienst, 42 pp. doi:10.5676/DWD_pub/nwv/icon_007, URL https://www.dwd.de/DE/leistungen/reports_on_icon/reports_on_icon.html.
- Reinert, D., F. Prill, H. Frank, M. Denhardt, and G. Zängl, 2020: *Database Reference Manual for ICON and ICON-EPS*. Deutscher Wetterdienst (DWD), URL https://www.dwd.de/EN/ourservices/reports_on_icon/reports_on_icon.html.

- Rieger, D., M. Köhler, R. J. Hogan, S. A. K. Schäfer, A. Seifert, A. de Lozar, and G. Zängl, 2019: ecRad in ICON - Details on the Implementation and First Results. (4), doi: 10.5676/DWD_pub/nwv/icon_004.
- Rieger, D., and Coauthors, 2015: ICON–ART 1.0 a new online-coupled model system from the global to regional scale. *Geosci. Model Dev.*, 8 (6), 1659–1676, doi:10.5194/ gmd-8-1659-2015.
- Rotta, J., 1951a: Statistische Theorie nichthomogener Turbulenz. J. Z. Physik, **129** (6), 547–572, doi:10.1007/BF01330059.
- Rotta, J., 1951b: Statistische Theorie nichthomogener Turbulenz. J. Z. Physik, 131 (1), 51–77, doi:10.1007/BF01329645.
- Ruppert, T., 2007: Diplomarbeit: Vector field reconstruction by radial basis functions. M.S. thesis, Department of Mathematics, Technical University Darmstadt.
- Sadourny, R., A. Arakawa, and Y. Mintz, 1968: Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere. *Mon. Weather Rev.*, 96 (6), 351–356, doi:10.1175/1520-0493(1968)096<0351:IOTNBV>2.0.CO;2.
- Satoh, M., 2002: Conservative scheme for the compressible nonhydrostatic models with the horizontally explicit and vertically implicit time integration scheme. *Mon. Weather Rev.*, **130**, 1227–1245.
- Schär, C., D. Leuenberger, O. Fuhrer, D. Lüthi, and C. Girard, 2002: A new terrainfollowing vertical coordinate formulation for atmospheric prediction models. *Mon. Weather Rev.*, 130, 2459–2480.
- Schrodin, R., and E. Heise, 2001: The Multi-Layer Version of the DWD Soil Model TERRA_LM. Tech. Rep. 2, Consortium for Small-Scale Modelling, 1–16 pp. URL http://www.cosmo-model.org.
- Schröter, J., and Coauthors, 2018: ICON-ART 2.1: a flexible tracer framework and its application for composition studies in numerical weather forecasting and climate simulations. *Geosci. Model Dev.*, **11 (10)**, 4043–4068, doi:10.5194/gmd-11-4043-2018.
- Schulz, J.-P., 2006: The new Lokal-Model LME of the German Weather Service. COSMO News Letter No. 6, Consortium for Small-Scale Modelling, 210–212, URL http://www. cosmo-model.org.
- Schulz, J.-P., 2008: Introducing sub-grid scale orographic effects in the cosmo model. COSMO News Letter No. 9, Consortium for Small-Scale Modelling, 29–36, URL http: //www.cosmo-model.org.
- Schulz, J.-P., G. Vogel, C. Becker, S. Kothe, U. Rummel, and B. Ahrens, 2016: Evaluation of the ground heat flux simulated by a multi-layer land surface scheme using highquality observations at grass land and bare soil. *Meteorol. Z.*, 25 (5), 607–620, doi: 10.1127/metz/2016/0537.
- Scinocca, J. F., 2003: An accurate spectral nonorographic gravity wave drag parameterization for general circulation models. J. Atmos. Sci., 60 (4), 667–682, doi:10.1175/ 1520-0469(2003)060<0667:AASNGW>2.0.CO;2.

- Seifert, A., 2008: A revised cloud microphysical parameterization for COSMO-LME. COSMO News Letter No. 7, Proceedings from the 8th COSMO General Meeting in Bucharest, 2006, Consortium for Small-Scale Modelling, 25–28, URL http://www. cosmo-model.org.
- Seifert, A., and K. D. Beheng, 2006: A two-moment cloud microphysics parameterization for mixed-phase clouds. Part 1: Model description. *Meteorol. Atmos. Phys.*, **92** (1), 45–66, doi:10.1007/s00703-005-0112-4.
- Siebesma, A. P., and J. W. M. Cuijpers, 1995: Evaluation of parametric assumptions for shallow cumulus convection. J. Atmos. Sci., 52, 650–666.
- Simmons, A., and D. Burridge, 1981: An energy and angular-momentum conserving finitedifference scheme and hybrid vertical coordinates. *Mon. Weather Rev.*, 109, 758–766.
- Skamarock, W., J. B. Klemp, M. G. Duda, L. D. Fowler, S. Park, and T. D. Ringler, 2012: A Multiscale Nonhydrostatic Atmospheric Model Using Centroidal Voronoi Tesselations and C-Grid Staggering. *Mon. Weather Rev.*, 140, 3090–3105, doi:10.1175/ MWR-D-11-00215.1.
- Skamarock, W. C., and J. B. Klemp, 2008: A time-split nonhydrostatic atmospheric model for weather research and forecasting applications. J. Comput. Phys., 227, 3465–3485.
- Skamarock, W. C., and M. Menchaca, 2010: Conservative transport schemes for spherical geodesic grids: High-order reconstructions for forward-in-time schemes. *Mon. Weather Rev.*, 138, 4497–4508.
- Skamarock, W. C., and Coauthors, 2019: A description of the advanced research WRF model version 4. No. ncar/tn-556+str, National Center For Atmospheric Research, Boulder, CO. doi:10.5065/1dfh-6p97.
- Smagorinsky, J., 1963: General Circulation Experiments with the Primitive Equations. Mon. Weather Rev., 91, 99, doi:10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO; 2.
- Smiatek, G., J. Helmert, and E.-M. Gerstner, 2016: Impact of land use and soil data specifications on COSMO-CLM simulations in the CORDEX-MED area. *Meteorol. Z.*, 25 (2), 215–230, doi:10.1127/metz/2015/0594.
- Smiatek, G., B. Rockel, and U. Schättler, 2008: Time invariant data preprocessor for the climate version of the COSMO model (COSMO-CLM). *Meteorol. Z.*, **17** (4), 395–405, doi:10.1127/0941-2948/2008/0302.
- Snyder, J., 1987: *Map Projections a Working Manual*. No. 1395, U.S. Geological Survey professional paper, U.S. Government Printing Office.
- Sommeria, G., and J. W. Deardorff, 1977: Subgrid-Scale Condensation in Models of Nonprecipitating Clouds. J. Atmos. Sci., 34 (2), 344–355, doi:10.1175/1520-0469(1977) 034<0344:SSCIMO>2.0.CO;2.
- Straka, J. M., R. B. Wilhelmson, and K. K. Droegemeier, 1993: Numerical solutions of a non-linear density current: A benchmark solution and comparisons. Int. J. Numer. Methods Fluids, 17, 1–22.

- Strang, G., 1968: On the construction and comparison of difference schemes. SIAM J. Numer. Anal., 5 (3), 506–517, doi:10.1137/0705041.
- Tibaldi, S., 1986: Envelope orography and maintenance of quasi-stationary waves in the ECMWF model. Adv. Geophys., 29, 339–374.
- Tiedtke, M., 1989: A comprehensive mass flux scheme for cumulus parameterization in large-scale models. Mon. Weather Rev., 117 (8), 1779–1800, doi:10.1175/ 1520-0493(1989)117<1779:ACMFSF>2.0.CO;2.
- Tomita, H., M. Satoh, and K. Goto, 2002: An optimization of the icosahedral grid modified by spring dynamics. J. Comput. Phys., 183 (1), 307–331, doi:10.1006/jcph.2002.7193.
- Wacker, U., and F. Herbert, 2003: Continuity equations as expressions for local balances of masses in cloudy air. *Tellus A*, 55, 247–254.
- Wallace, J., S. Tibaldi, and A. Simmons, 1983: Reduction of systematic forecast errors in the ECMWF model through the introduction of an envelope orography. Q. J. R. Meteorol. Soc., 109, 683–717.
- Wan, H., and Coauthors, 2013: The ICON-1.2 hydrostatic atmospheric dynamical core on triangular grids – Part 1: Formulation and performance of the baseline version. *Geosci. Model Dev.*, 6 (3), 735–763, doi:10.5194/gmd-6-735-2013.
- Warner, C. D., and M. E. McIntyre, 1996: On the propagation and dissipation of gravity wave spectra through a realistic middle atmosphere. J. Atmos. Sci., 53 (22), 3213–3235, doi:10.1175/1520-0469(1996)053<3213:OTPADO>2.0.CO;2.
- Yeh, K.-S., 2007: The streamline subgrid integration method: I. quasi-monotonic second order transport schemes. J. Comput. Phys., 225, 1632–1652.
- Zalesak, S. T., 1979: Fully multidimensional flux-corrected transport algorithms for fluid. J. Comput. Phys., 31, 335–362.
- Zängl, G., 2012: Extending the Numerical Stability Limit of Terrain-Following Coordinate Models over Steep Slopes. Mon. Weather Rev., 140, 3722–3733, doi:10.1175/ MWR-D-12-00049.1.
- Zängl, G., D. Reinert, and F. Prill, 2022: Grid refinement in icon v2.6.4. Geosci. Model Dev. Disc., 2022, 1–32, doi:10.5194/gmd-2022-120.
- 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 nonhydrostatic dynamical core. Q. J. R. Meteorol. Soc., 141, 563–579.
- Zarzycki, C., M. N. Levy, C. Jablonowski, J. R. Overfelt, M. A. Taylor, and P. Ullrich, 2014: Aquaplanet experiments using CAM's variable resolution dynamical core. J. Clim., 27, 5481–5503, doi:10.1175/JCLI-D-14-00004.1.
- Zdunkowski, W., and A. Bott, 2003: Dynamics of the atmosphere: A course in theoretical meteorology. 1st ed., Cambridge University Press, 719 pp.

- Zerroukat, M., N. Wood, and A. Staniforth, 2002: SLICE: A Semi–Lagrangian Inherently Conserving and Efficient scheme for transport problems. Q. J. R. Meteorol. Soc., 128, 2801–2820, doi:10.1256/qj.02.69.
- Zerroukat, M., N. Wood, and A. Staniforth, 2005: A monotonic and positive-definite filter for Semi-Lagrangian Inherently Conserving and Efficient (SLICE) scheme. Q. J. R. Meteorol. Soc., 131, 2923–2936.
- Zerroukat, M., N. Wood, and A. Staniforth, 2006: The Parabolic Spline (PSM) Method for conservative transport problems. *Int. J. Numer. Meth. Fluids*, **51**, 1297–1318.

## **Index of Namelist Parameters**

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

icon/doc/Namelist_overview.pdf

for a complete list of available namelist parameters for the ICON model.

initicon_nml (Namelist), 148

albedo_type, 54 ana_varnames_map_file, **147** 

bdy_indexing_depth, **27**, 27 bub_amp, **141**, 142 bub_hor_width, **140**, 142 bub_ver_width, **141**, 142 bubctr_z, **141**, 142

checkpointTimeIntVal, **174**, 174, **175** cldopt_filename, **57** const_z0, 142

damp_height, 136, 157, 163, 164, 165 diffusion nml (Namelist), 132, 136, 142 dom, 168, 169 dt checkpoint, 174, 175, 175 dt conv, 90, 90 dt gwd, 90 dt iau, 149, 149–151 dt_rad, 90, 90, 96 dt restart, 175, 175 dt shift, **149**, 149–151, **152** dt sso, 90dtime, 89, 133, 136, 142, 144, 156, 174 dtime latbc, 161 dwdana filename, 148, 148, 149, 152 dwdfg_filename, 148, 148, 149, 151, 152, 158, 159 dynamics grid filename, 24, 118, 133, 136, 142, 145, 146, 193 dynamics nml (Namelist), 132, 136, 142

dynamics parent grid id, 23, 24, 24 ecrad data path, 57, 93 ecrad isolver, 187, 187 end datetime string, 144, 144 end_time, 144, 152, 152, 174 exner_expol, 75, 136 experimentStartDate, 144, 152, 162, 169, 174experimentStopDate, 144, 174 extpar filename, 145, 145–147 extpar nml (Namelist), 54-56, 132, 136, 142, 145, 147, 235  $extpar_varnames_map_file, 147$ fbk relax timescale, 124 filename_format, 168, 168, 169 filetype, 168, 168 flat_height, 69, 126 frlake_thrhld, 110 grid nml, 68 grid_nml (Namelist), 24, 70, 118, 122, 129, 133, 136, 142, 144, 145, 152, 153, 174gridgen_nml (Namelist), 27, 28 gridref_nml (Namelist), 124

h_levels, **170** hbot_qvsubstep, 86, **115**, 115, 116 hdiff_efdt_ratio, 136, 142 hdiff_order, 136, 142 hdiff_smag_fac, 142 hl varlist, **170**, 170 htop moist proc, 88, 115, 115, 116 i levels, 170 icapdcycle, **100**, 100 icld overlap, 95 iforcing, 132, 132, 136, 142, 145 ifs2icon filename, 148, 152, 158, 159 igradp method, 64 ihadv_tracer, 86, 86-88, 115 iice scat, 95 il varlist, **170**, 170 iliquid scat, 95 in filename, 53 in grid filename, 53 in mask below, 47 in_mask_threshold, 47 in type, 53 inextra 2d, 200 inextra 3d, 200 ini datetime string, **144**, 144, 148, 149, 151, 152, 162, 169 init mode, 43, **146**, 146, 148, 149, 151, 152, 158, 158, 159 initicon_nml (Namelist), 110, 114, 146–149, 151, 152, 158, 159 input field nml (Namelist), 43–47, 53 interpol nml (Namelist), 86, 126, 155, 171 inwp cldcover, 94 inwp convection, 94, 99, 163, 164 inwp_gscp, 85, 94, 94, 98, 98, 99 inwp_gwd, 94 inwp radiation, 93, 94, 187 inwp sso, 94 inwp surface, 94 inwp turb, 94, 103, 142 io nml (Namelist), 147, 168, 175, 176, 200 irad_aero, 54 is_dry_cbl, 142 is plane torus, 142isrfc type, 142 iterative iau, 151 itopo, 132, 132, 136, 142, 145 itype canopy, 234, 239 itype_hlimit, 86, 88 itype_latbc, 154, 161 itype_lwemiss, 54 itype sher, 103 itype snowevap, 234, 236–239

itype t diffu, 136 itype trvg, 234, 236, 238 itype vegetation cycle, 55, 235, 239, 239 itype vlimit, 83, 86, 88 itype vn diffu, 136 itype z0, 55ivadv tracer, 86, 87, 88 ivctype, 68, 69, 163 ivlimit selective, 87, 88 jw temp0, **134**, 136 jw u0, **134**, 136 jw up, **134**, 136 Kh ext, 142 Km ext, 142 l limited area, 153, 156 latbc boundary grid, 157, 162 latbc filename, 161, 161 latbc path, **161**, 161 latbc varnames map file, 147, 161, 162 latm above top, 157 layer thickness, 68 lconst z0, 142lcoriolis, **132**, 136, 142 ldynamics, 131, 136, 142, 145 les nml (Namelist), 104, 142 lfeedback, 122 limarea_nml (Namelist), 147, 154, 155, 157, 161, 162 linside domain test, 47 linvert dict, 147 llake, 94, 109, 110 lmeteogram enabled, 176 lnd nml (Namelist), 54–56, 109, 110, 112, 113, 234, 236-239 lprog_albsi, 234, 236, 238 lread_ana, 148, 148, 149, 159, 159 lredgrid phys, 129 lrestart, 174, 175 lrtm_filename, 57 lseaice, 94, 112 lshallowconv only, 99, 163, 164 lsnowtile, 113 lsq_high_ord, 86 ltestcase, 68, 131, 136, 142, 144 ltile coldstart, 114 ltile init, 114

ltimer, 183 ltkeshs, **103**, 103 ltransport, 85, 132, 136, 142, 145 ltransport=.TRUE., 135 lvert nest, 67, 126 lwrite parent, 28, 130

m levels, 168 master nml (Namelist), 146, 148, 169, 174, 175master time control nml (Namelist), 144, 174.175 max_nudge_coeff_thermdyn, 155 max_nudge_coeff_vn, 155 max refin c ctrl, 51, 51 max time stamps, 177 merge domain, 129 meteogram_output_nml (Namelist), 176, nwp tuning nml (Namelist), 106, 108, 163 177 min lay thckn, **69**, 69, 142 min refin c ctrl, 51 ml varlist, 168, 168, 170 model base dir, 146, 148, 169 modelTimeStep, 144 msg_level, 132 n0 mtgrm, 176  $n_{flat}ve v, 68$ n iter smooth topo, 56 nadv substeps, 86 nblocks_c, 186, 186 nblocks_sub, 186, 186  $ncstorage_file, 53$ ndvn substeps, 89, 89, 90, 120 netcdf dict, 147 nh brunt vais, 141, 142 nh_test_name, 134, 136, 142 nh testcase nml (Namelist), 131, 134, 136, 140 - 142nh testcases nml, 68 nh u0, **141** ninc mtgrm, 176 nonhydrostatic nml (Namelist), 64, 68, 69,

72, 73, 75, 86, 88, 89, 115, 120, 132, 136, 157, 163 north pole, 172, 173 nproma, 182, 186, 186, 189, 190 nproma sub, 186, 186, 187

nsteps, 133, 136, 142, 144, 174 ntiles, 54, 55, 56, 113, 113–115 ntracer, 85, 136 nudge efold width, 126, 155 nudge hydro pres, 155 nudge max coeff, **126**, 155 nudge start height, 155 nudge type, 51, **154**, 156, 157 nudge zone width, **126**, 155 nudging_nml (Namelist), 51, 154–157 num_io_procs, 173, 174, 180 num_lev, 67, 69, 126, 133, 136, 142, 165 num prefetch proc, 162, 162 num restart proc, 176 num restart procs, 175, 180 nwp phy nml (Namelist), 55, 57, 90, 93, 94, 98-100, 103, 132, 142, 157, 163 out grid filename, 53 out mask below, 47 out mask filename, 47 out_mask_threshold, 47 out_type, 53 output, 167, 176 output bounds, 168 output filename, 167, 168 output nml (Namelist), 133, 136, 167–170, 172-174, 255 output nml dict, 147, 168 p levels, **170** parallel_nml (Namelist), 162, 174–176, 180, 182, 186, 189 pe placement ml, 174 pl varlist, **170**, 170 proc0 shift, 182, 182, 187 radiation_grid_filename, 129, 145 radiation nml (Namelist), 54, 57, 93, 95, 187 rat_sea, 163, 163 rayleigh coeff, 136 rbf scale, 172 reg_lat_def, 169  $reg_lon_def, 169$ remap, 168, 169

remap nml (Namelist), 47, 53 restart filename, 175, 175

271

restart write mode, **176**, 176 restartTimeIntVal, 174, 175 rhotheta offctr, 73 run nml (Namelist), 67, 68, 85, 126, 131-133, 135, 136, 142, 144, 145, 167, 174, 175, 183 sleve nml, 70 sleve_nml (Namelist), 69, 126, 136, 142, 155, 163 smag_coeff_type, 142 smag constant, 104 sstice mode, 55 start time, 144, 152, 152, 174  $stationlist_tot, 176$ steps_per_file, 169 stream_partitions_ml, 173, 174, 174 support_baryctr_intp, 171 time_nml (Namelist), 144, 175 timers_level, 183top_height, 69, 136, 142, 155, 163, 164, 165 tracer inidist list, 136, 137 tracer names, 137, 137 transport_nml (Namelist), 83, 86, 87, 115, 132, 137, 145  $tune_gfluxlaun, 108$ tune_gfrcrit, 106 tune_gkdrag, **106**, 106 tune_gkwake, **106**, 106 tune_grcrit, 106 tune_zvz0i, 163, 163 turb prandtl, 104 turbdiff_nml (Namelist), 103, 142, 163 ufric, 142use lakeiceana, 110 var_in_mask, 46, 47, 47 var_list, 177 var_out_mask, 47 vct_filename, 68, 70 veladv offctr, 73 vwind offctr, 72, 72, 136 zprefix, 177