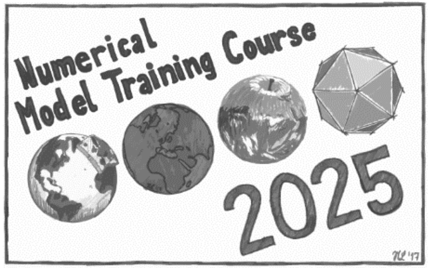
Getting Started with ICON

DWD ICON Course | July 2025 | Florian Prill, DWD

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Introduction



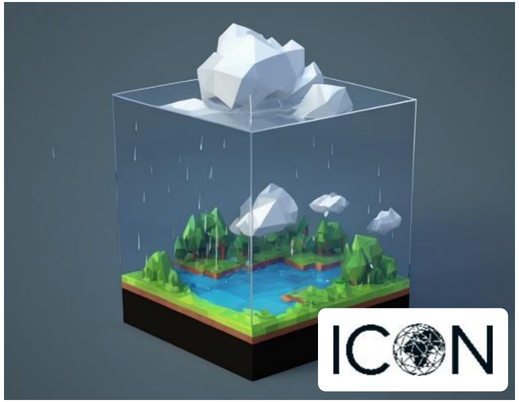
ICON

ICON-NWP

ICON is a modelling framework, consisting of various

- institutions: DWD, MPI-M, DKRZ, KIT, C2SM, MCH, ...
- communities: users and developers, MetServices, universities, CLM, natESM, ...

ICON is used with various computing platforms:
x86 clusters, GPUs, vector engines …



In this talk we move along a path called **ICON-NWP**, which is the numerical weather prediction (NWP) branch of the model, developed for operational and regional weather forecasting by the German Weather Service.

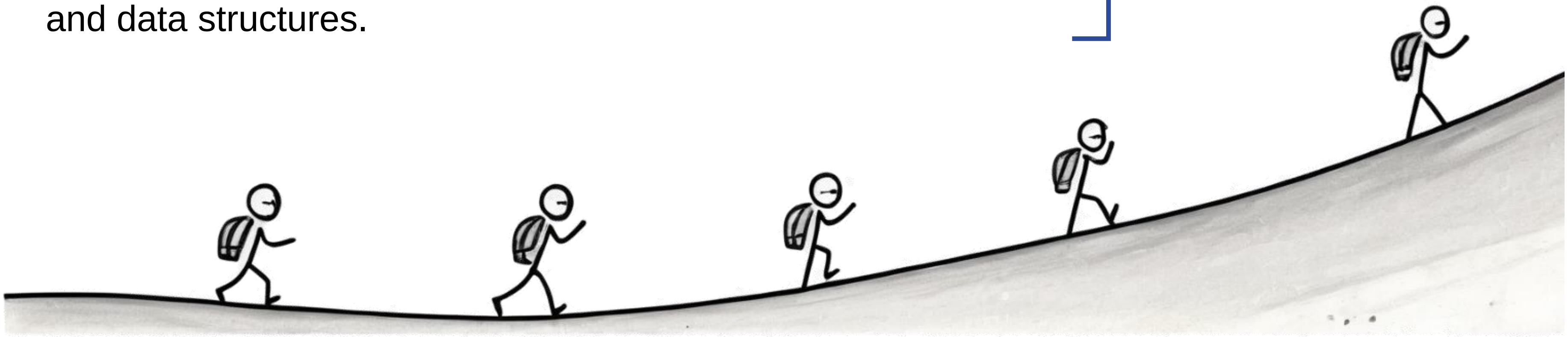
Unavoidable simplifications: we omit ICON's ocean model, wave model, …

Talk outline

We can get started with the ICON model in three different ways:

- We begin with the source code, the various subdirectories and libraries involved.
- We consider a model setup, and follow the flow of control of a typical model run.
- We have a closer look at some of the functional components and data structures.

~ 45 minutes



Building blocks of the ICON model

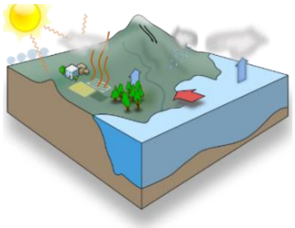
Many contents are dealt with in specific lectures…

`src/atm_dyn_iconam`

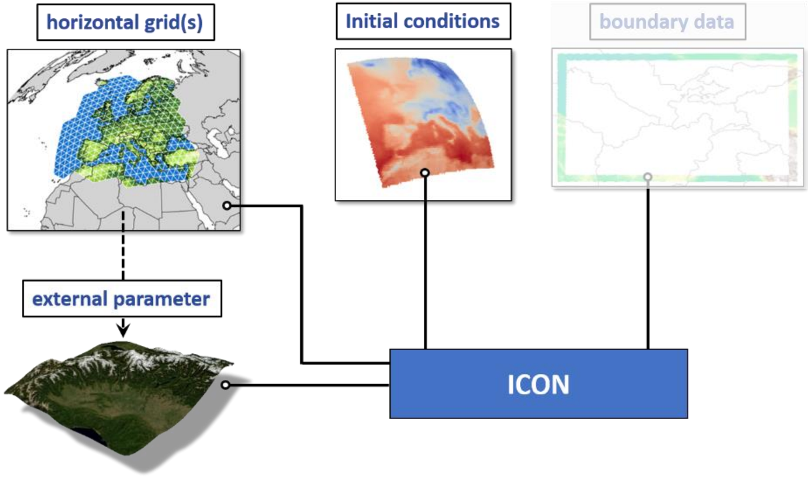
$$
\begin{aligned}
&\partial_t v_n \quad + (\zeta + f)\, v_t \quad + \ldots \\
&\partial_t w \quad + \mathbf{v}_n \cdot \nabla w \quad + \ldots \\
&\partial_t \rho \quad + \nabla \cdot (\mathbf{v}\rho) \quad = \ldots \\
&\partial_t (\rho\theta_v) + \nabla \cdot (\mathbf{v}\rho\theta_v) \quad = \ldots
\end{aligned}
$$

- **DyCore**
  see
  dedicated talk

`src/atm_phy_schemes`



- **Physics**
  see various
  talks this week



- **Input and boundary data**
  see dedicated talk

ICON Tutorial and further documentation

- **Users**: see https://docs.icon-model.org
and we also refer to the **Tutorial Book** (linked there)

⇨ chapters 0-2, 7-10

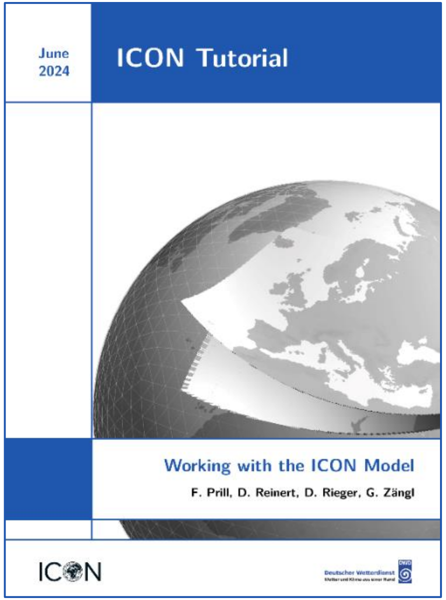
- **Scientific publications**, eg.,

  Zängl et al. (2015): The ICON modelling framework of DWD and MPI-M: Description of the nonhydrostatic dynamical core. Q. J. R. Meteorol. Soc., 141, 563-579.

  (see https://www.icon-model.org/publications/reference-publications for more)

- DWD's operational products are documented in the
DWD database manual

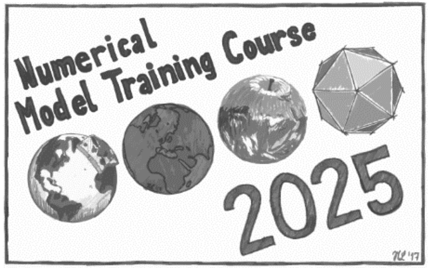
  www.dwd.de/SharedDocs/downloads/DE/modelldokumentationen/
  nwv/icon/icon_dbbeschr_aktuell.pdf



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

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Part 1: Source Code



ICON source code

- Components: **weather, climate, ocean, land.**
- **Model source code - Fortran code (***F2003 compliant***)**
- **Auxiliary libraries**: written in C99

Root directory: **icon**

config	*platform configuration*
externals	*external submodules (ART, calendar etc.)*
run	*test scripts*
src	*Fortran sources*
support	*C99 support library and utility routines*
build	*build directory with sources and binary*

ICON model code

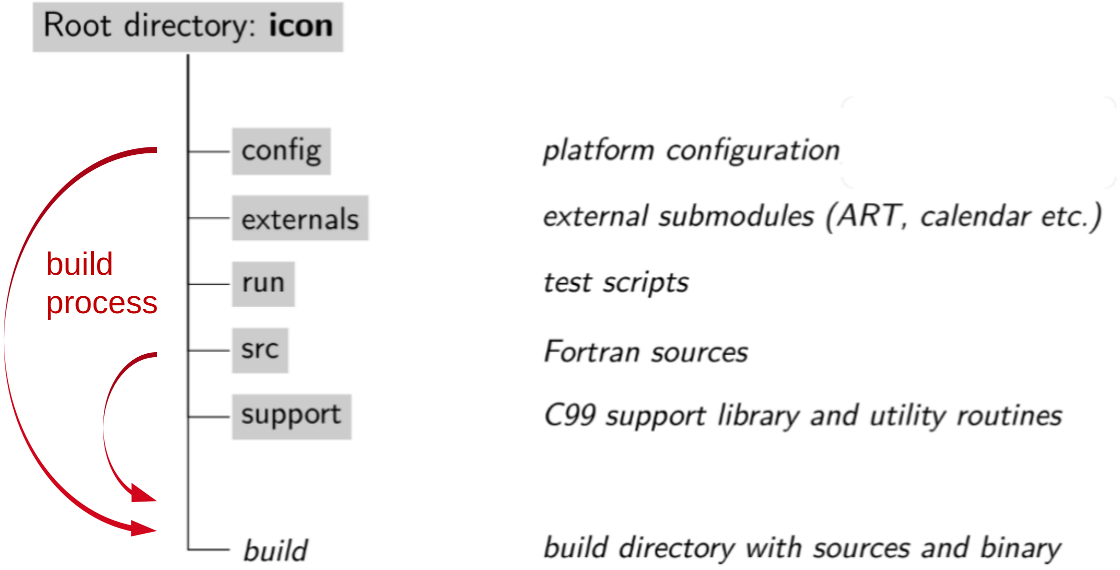
spring release 2025.04

Releases can be found under

gitlab.dkrz.de/icon/icon-model

ICON versions are named
using the scheme \`icon-yyyy.mm\`
where \`yyyy\` is the year
and \`mm\` is the month
of the release

Build process: configure & compile



Configure options:

`configure -h`

Parallel build:

`make -j4`

List of tested compilers
see Tutorial book, Section 1.3

GNU, Cray, Intel, NAG, NEC, NVIDIA

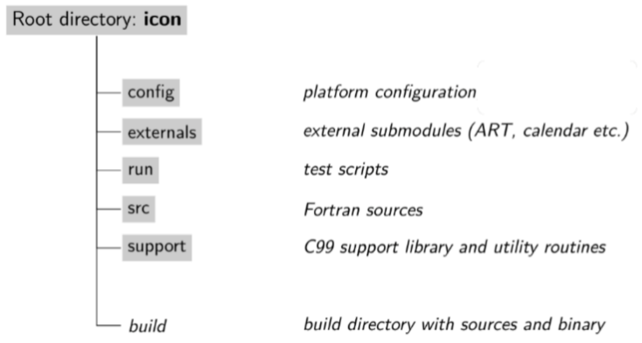
- see **`https://docs.icon-model.org/buildrun/buildrun_building.html`**:
how to configure, build and run ICON.

Build process: configure & compile

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

The configuration step is typically executed by running the **`configure`** script with command-line arguments[1].

Instead of running the generic **`configure`** script directly, it is recommended to use a platform-specific configuration wrapper.
The wrappers can be found in the **`config`** subdirectory.



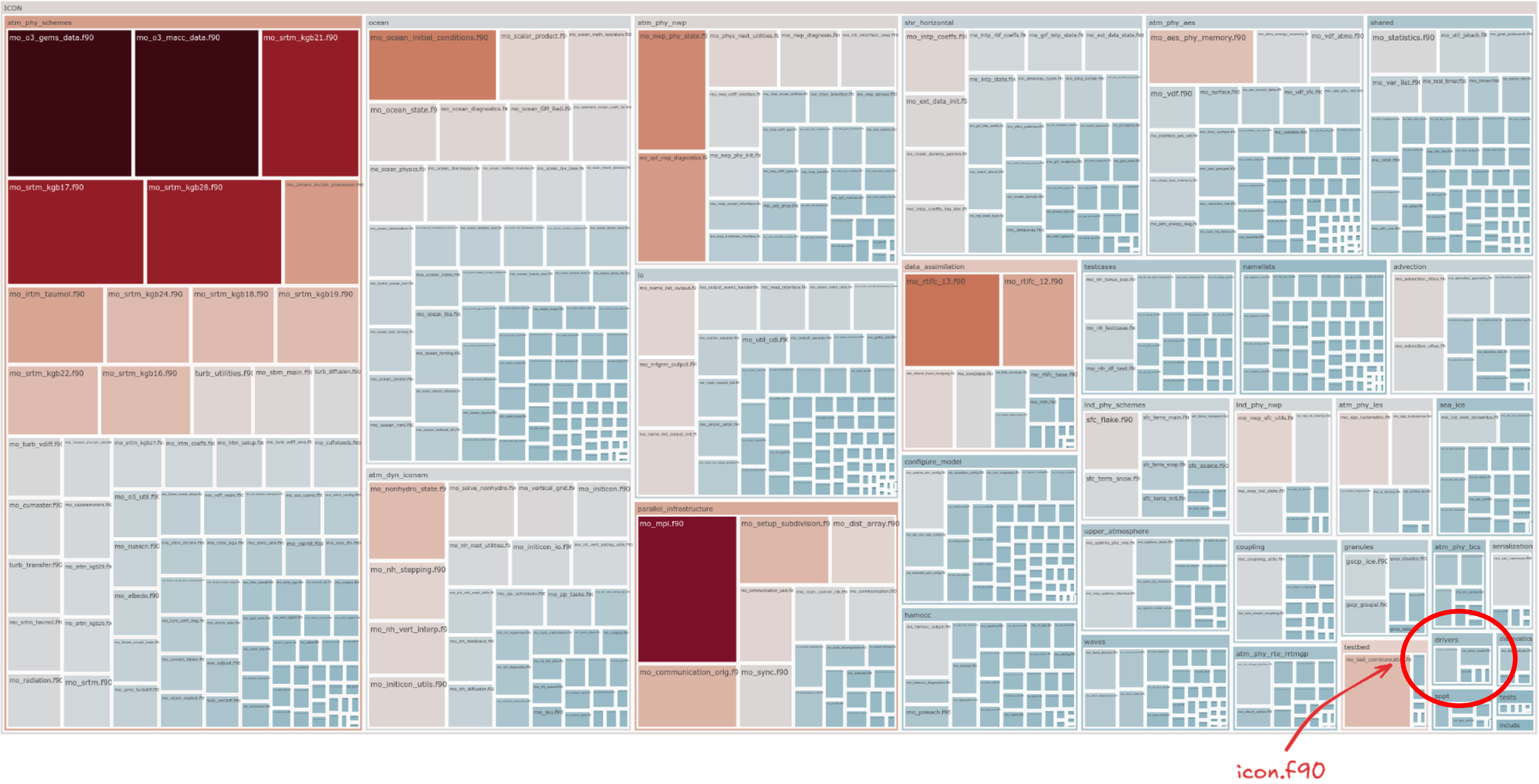
[1] … see Autoconf: https://www.gnu.org/software/autoconf/

ICON source code

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Release ICON 2025.04

Fortran and C code: 1,168,413 lines of which 523,853 are in `icon/src`[1]

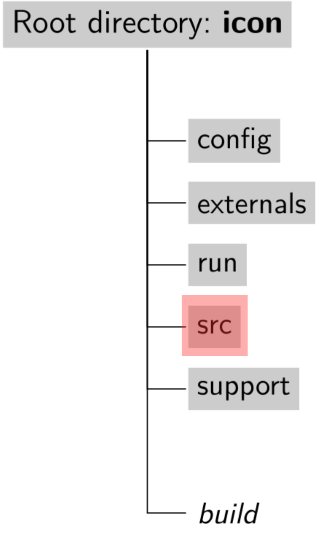


[1] SLOC count

Subdirectory **`icon/src`**

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

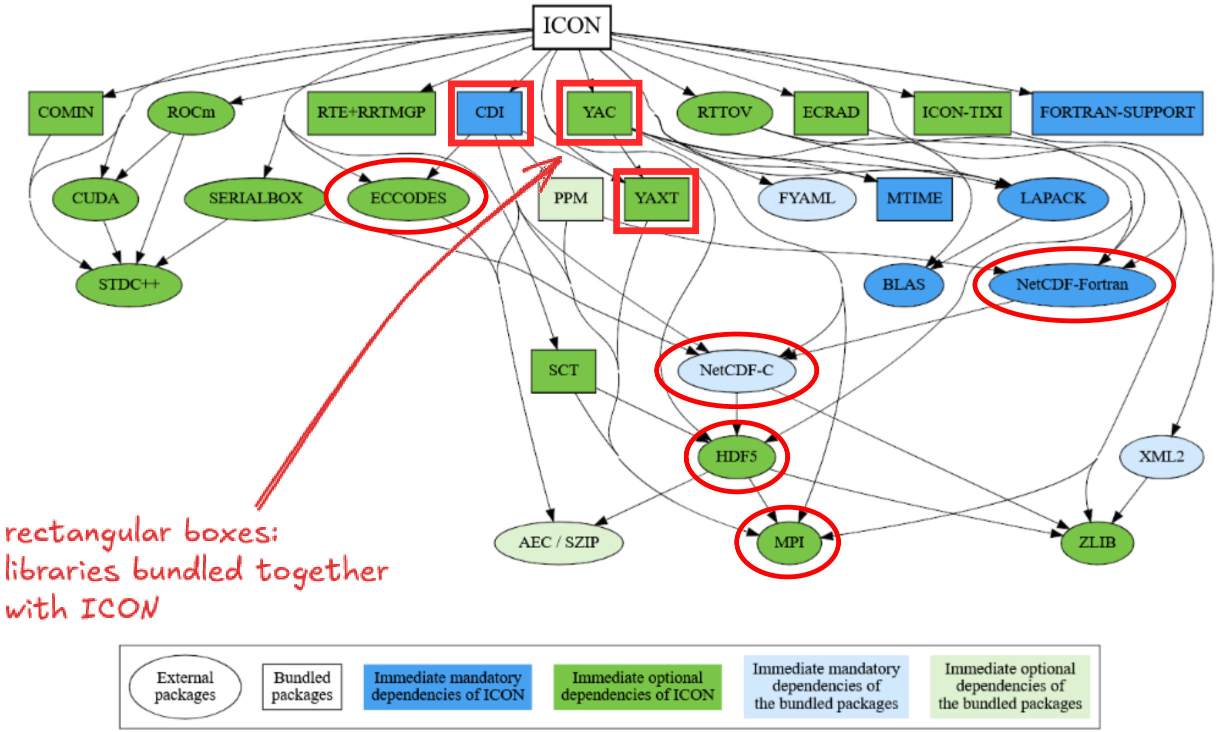
- Subdirectory **`src/drivers`**
  main program and nonhydrostatic setup
- Subdirectory **`src/atm_dyn_iconam`**
  dynamics implementation
- Subdirectory **`src/atm_phy_nwp`** and **`src/lnd_phy_nwp`**
  physical parameterizations
- Subdirectories **`src/configure_model`** and **`src/namelists`**
  configuration of run-time settings
- Subdirectories **`src/shared`** and **`src/shr_horizontal`**, and **`src/parallel_infrastructure`**
  shared infrastructure modules
- Subdirectory **`src/io`**
  Input and output modules



Software libraries

The ICON model package integrates a whole variety of external libraries.

Several of them are <u>mandatory</u> for parallelization, coupling, and model I/O!



Software libraries (cont'd)

- **MPI - Message Passing Interface**
  distributed-memory parallelization
- **OpenMP**
  multi-threading, part of your compiler
- **NetCDF and ecCodes/GRIB-API**
  read/write data

wrapper scripts with flags and library paths for various platforms

`icon/config/`

… append your additional options

Infrastructure libraries distributed together with the ICON model:

Climate Data Interface (CDI) - https://code.mpimet.mpg.de/projects/cdi

YAC (ICON coupler software, German Climate Computing Centre DKRZ)

and others

NEC SX-Aurora rack mount model

External packages - `icon/externals`

YAC is an example of a switchable external package.

Further examples:

- ComIn
  plugin mechanism (e.g. Python scripts)
- ecRad
  atmospheric radiation scheme
- JSBACH
  ICON-LAND
- YAXT
  parallel communication library

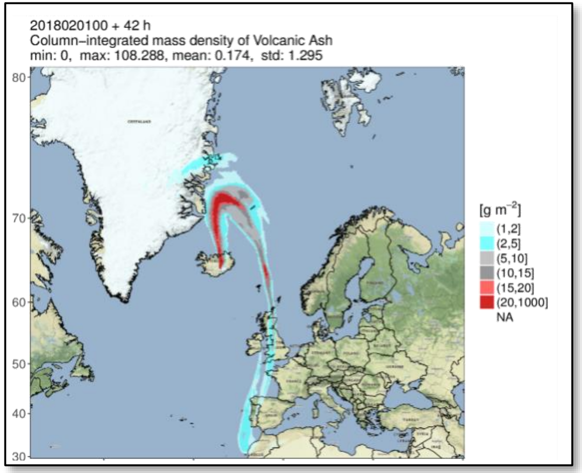
Note: Not all modules that can be switched via **`configure`** are available in the Open Source Release.

ICON-ART

The ICON-ART package enables ICON to simulate gases, aerosol particles and related feedback processes in the atmosphere.

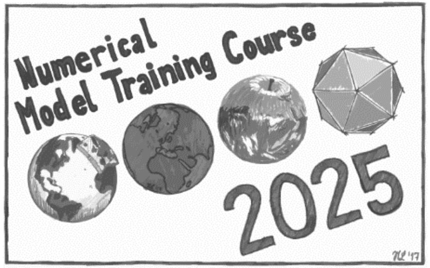
- "ART" stands for
  Aerosols and **R**eactive **T**race gases.
- ICON-ART requires additional source code, which is provided by the **Karlsruhe Institute of Technology**.
- ICON-ART is a good example for a model extension, which consists of the **adapter source code** (externals/art/interface) and the **module code** (externals/art) itself.

Karlsruhe Institute of Technology



Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Part 2: Typical model run



ICON

ICON’s namelists

In general, the ICON model is controlled by a parameter file which uses the Fortran NAMELIST syntax.

```
! --- parallel_nml: MPI parallelization ---
&parallel_nml
 nproma                       =              8         ! loop chunk length
 num_io_procs                 =              1         ! number of I/O processors
 num_restart_procs            =              0         ! number of restart processors
 iorder_sendrecv              =              3         ! sequence of MPI send/receive calls
/
```

- During the setup phase, the ICON model reads ~600 namelist parameters
  … most namelist parameters are not set explicitly, but use a default value!

Complete table with ICON namelist parameters:

icon/doc/Namelist_overview/Namelist_overview.pdf

most important namelist parameters see Tutorial book index!

`master_nml` and NWP namelists

For NWP simulations: Use the prepared exercises as “best practice settings”.

Bootstrap namelist file **`icon_master.nml`**

- initializes the ICON setup
- handles restart from saved model state: namelists are read from the **restart** file to override the default namelist settings, before reading new namelists from the run script
- specifies the “component namelist” for NWP in which the lion’s share of the parameters is defined **(often) named NAMELIST_NWP**

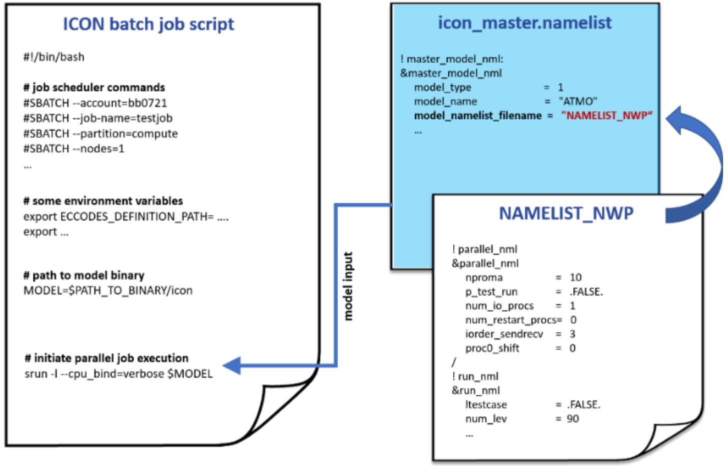
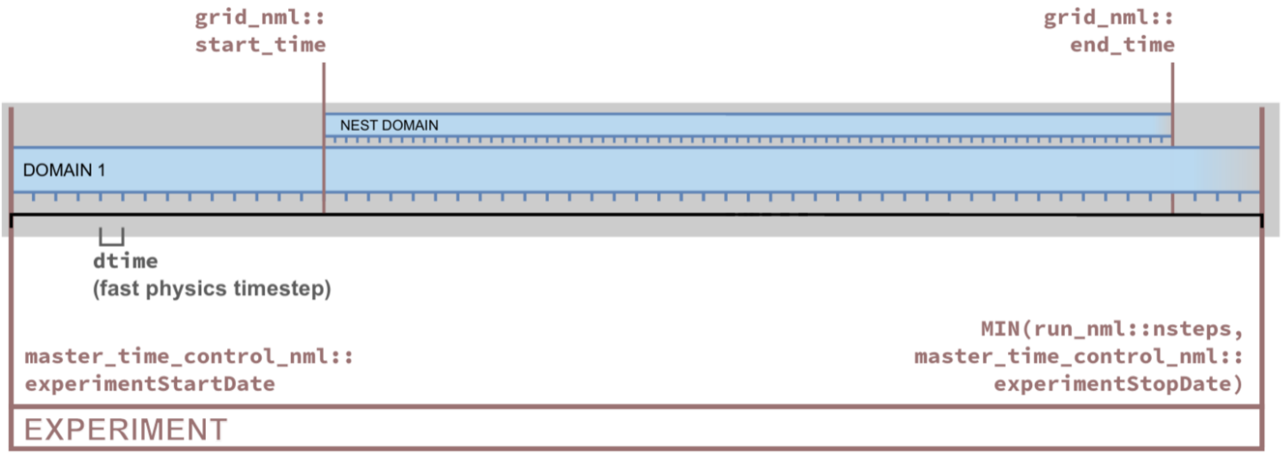


Illustration: D. Reinert, DWD

Namelist: setting start and end date

… and parameters for switching computational domains on and off during the simulation.

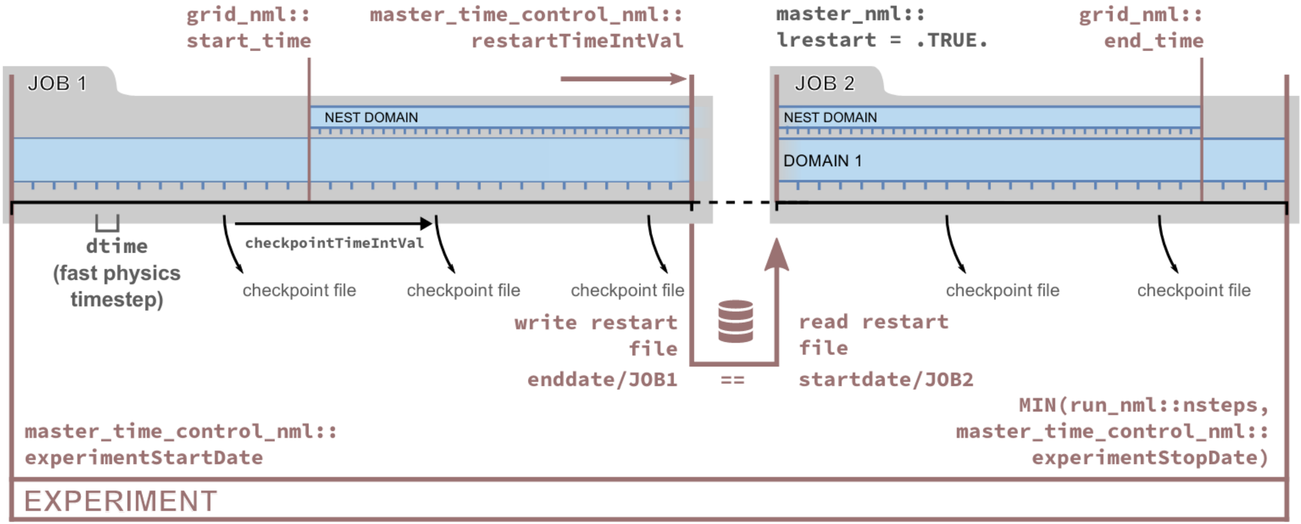


ICON Tutorial book - Section 5.1

Namelist: setting restart option

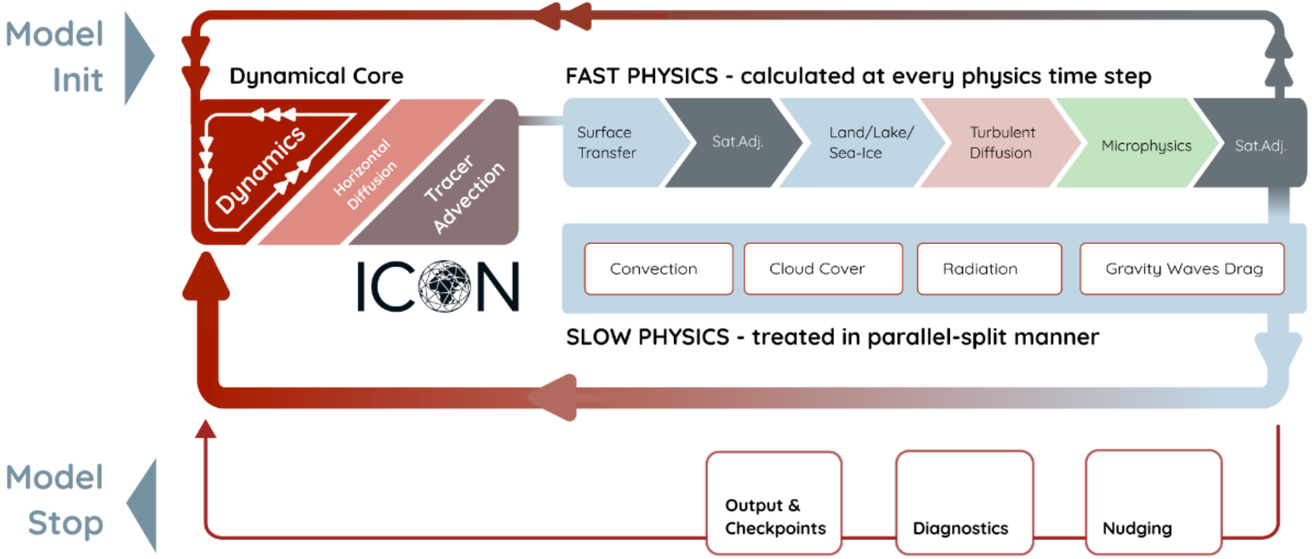
Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Restart option: allows to restart the execution from a pre-defined point using a checkpoint file.



ICON Tutorial book - Section 7.2

Flow of control



For efficiency reasons, different integration time steps are applied depending on the process.

Δt : basic time step (tracer, diffusion, fast physics) ; $\Delta\tau$: DyCore time step ; $\Delta t_{i,slow}$: slower processes

Grids and nested domains

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Many multi-domain namelist parameters:

```
&run_nml
  num_lev = 90, 60
```
number of vertical levels

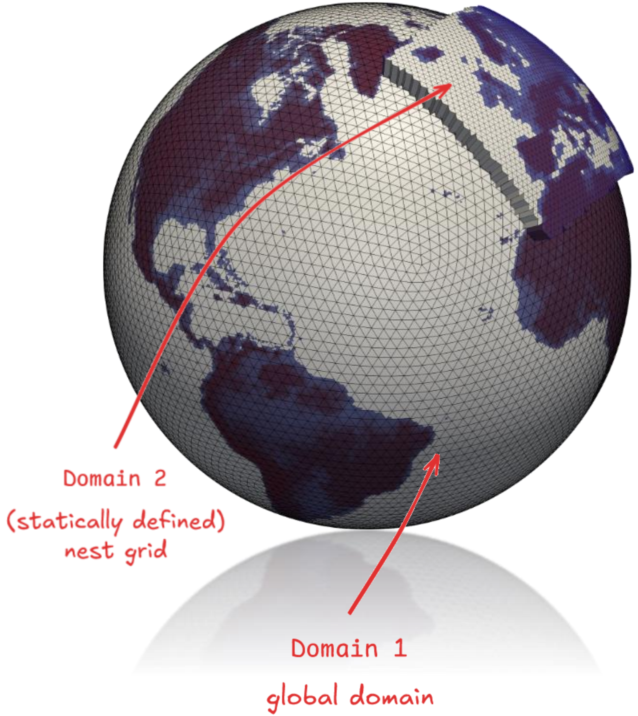
ICON has the capability for running

- global simulations on a single global grid
- global simulations with "nests" (= refined domains)
- regional simulations (ICON-LAM)

Nomenclature for icosahedral grids: RnBk

initial subdivision of icosahedral edges

subsequent edge bisections



Namelist specifications for the output

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

```
! output_nml: specifies an output stream --------------------------------------
!
&output_nml
 filetype                  =                      4         ! output format: 2=GRIB2, 4=NETCDFv2
 dom                       =                     -1         ! write all domains
 output_bounds             =   0., 10000000., 10800.        ! output: start, end, increment
 steps_per_file            =                      2         ! number of output steps in one file
 mode                      =                      1         ! 1: forecast mode (relative t-axis)
 output_filename           =                  'NWP'         ! file name base
 output_grid               =                 .TRUE.         ! flag: grid information in output?
 !
 ml_varlist                =  'u', 'v', 'w', 'temp', 'pres','topography_c', 'pres_msl', &
                              'qv', 'qc', 'qi', 'qr', 'qs', 'tke',                      &
                              'tkvm', 'tkvh', 'group:pbl_vars',                         &
                              'group:precip_vars',  'group:additional_precip_vars',     &
                              'group:land_vars', 'group:land_tile_vars',                &
                              'group:multisnow_vars,
/
```

- Multiple namelists **`output_nml`** are allowed.
- Model prints a detailed tabular summary at start-up.

List of output fields and variable groups

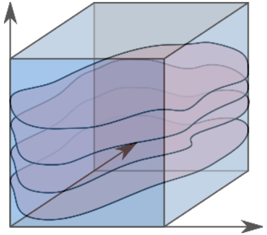
Variable Name	GRIB2 Name	Description
acdnc	ndcloud	Cloud droplet number concentration
adrag_u_grid		Zonal resolved surface stress mean since model start
adrag_v_grid		Meridional resolved surface stress mean since model start
aer_bc	aer_bc	Black carbon aerosol
aer_du	aer_dust	Total soil dust aerosol
aer_or	aer_org	Organic aerosol
aer_ss	aer_ss	Sea salt aerosol
aer_su	aer_so4	Total sulfate aerosol
aercl_bc		Black carbon aerosol c
aercl_du		Total soil dust aerosol
aercl_or		Organic aerosol climat
aercl_ss		Sea salt aerosol climat
aercl_su		Total sulfate aerosol cl
alb_dif	alb_dif	Shortwave albedo for d
alb_si	alb_seaice	Sea ice albedo (diffuse

List of available output fields and variable groups

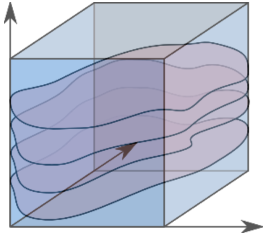
See Tutorial book, Appendix A

Internal post-processing

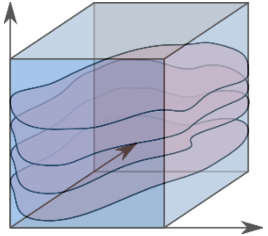
Vertical Grid



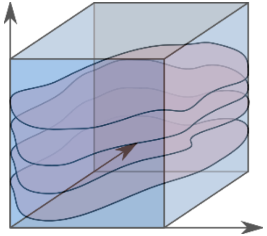
model levels



pressure levels



height levels

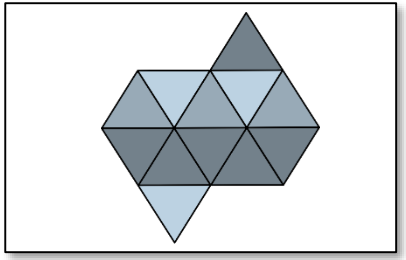


isentropic levels

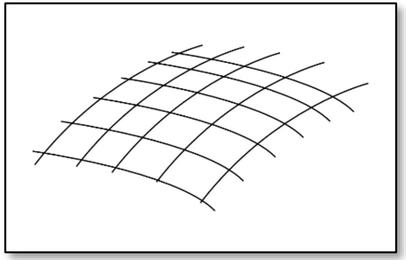
`output_nml::hl_varlist & h_levels`

X

Horizontal Grid



triangular grid



regular grids

`output_nml::remap`

GRIB2 data format

- GRIB Data Format: Defined by the World Meteorological Organization (WMO).
- The European Centre for Medium-Range Weather Forecasts (ECMWF) has developed a GRIB application programmers interface: ecCodes library (previously: GRIB-API).
- ICON supports only GRIB edition 2.

Model output in ICON accesses the ecCodes API through the CDI library.

DWD ecCodes Definition Files

ecCodes consists of two parts: The library and a definitions directory.

In particular, the variable name (string) is stored in the definitions but **not** in the data files!

To recognize GRIB2 variable names, you need to install DWD definition files

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

NetCDF data format

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

The NetCDF storage format can also be used for model output:

- storage format for data arrays and attributes, structured and unstructured
- self-describing, machine-independent

Many tools interfacing NetCDF are available:

dataset info/modification:	**`ncdump`, `nco`, `cdo infov`**
visualization:	**`ncview`**, NCL, Python

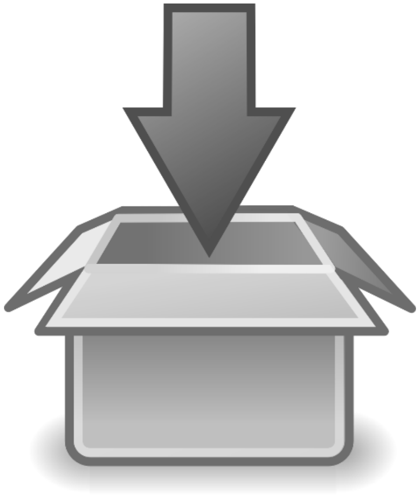
NetCDF-4/HDF5: permits storing files as large as the file system supports.

Typical file sizes

Example: global R2B6 grid with
327 680 triangular cells

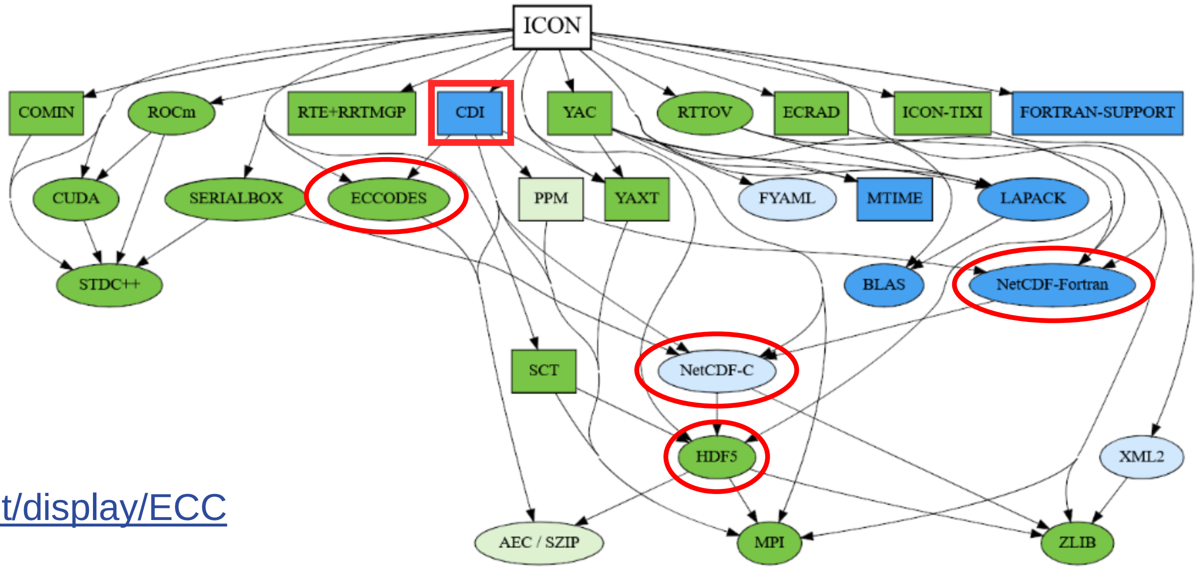
40 km grid spacing, 90 vertical levels

- Storage size of a single 2D layer:
  - NetCDF: 1.38 MB
  - GRIB2: 641 KB
- GRIB2 storage format:
  ~ 50% file size compared to NetCDF
- additional 22 MB grid topology data
  in NetCDF file



I/O Libraries

Deutscher Wetterdienst
Wetter und Klima aus einer Hand



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

- NetCDF
  https://www.unidata.ucar.edu/software/netcdf/

Complicated details: file name keywords

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

File name parameters for init data & output & boundary often use placeholders.

… the user specifies a list of file names through wildcards (“keywords”).

Example: Lateral boundary conditions, **`latbc_filename`** may contain:

<nroot>	grid root division Rx (single digit)
<nroot0>	grid root division Rxx (two digits)
<jlev>	grid bisection level Byy (two digits)
<dom>	domain number (two digits)
<y>	year (four digits)
<m>	month (two digits)
<d>	day in month (two digits)
<h>	hour (UTC) (two digits)
<min>	minutes (UTC) (two digits)
<sec>	seconds (UTC) (two digits)
<ddhhmmss>	elapsed days, hours, minutes and seconds since `ini_datetime_string` or `experimentStartDate` (each two digits)
<dddhh>	elapsed days and hours since `ini_datetime_string` or `experimentStartDate` (three digits day, two digits hours).

Namelist setting (example):

```
extpar_filename = 'extpar_DOM<idom>.nc'
```

Complicated details: dictionaries

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Field names that are internally used by the ICON model are not necessarily identical to the GRIB2 field names or the names in your NetCDF input file.

Dictionary files: two-column text files; translate between ICON variable names and GRIB2 or other external name schemes.

```
# internal name      GRIB2 shortName
#
temp                 T
pres                 P
pres_sfc             LNPS
rho                  DEN
…
```

Namelist parameters: **ana_varnames_map_file**, **latbc_varnames_map_file**, **var_names_map_file**, …

Error handling

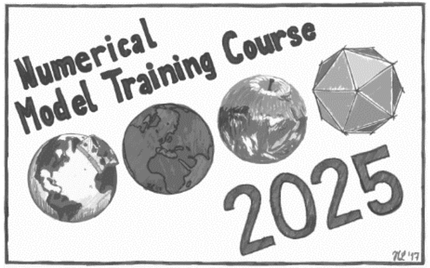
Fortran does not offer built-in support for exceptions and exception handling.

- **Exception handling in software: not implemented**
  The functions and subroutines in the ICON model do not return special error codes.
- **On error, the ICON model simply aborts:**
  When
  - an input argument is invalid
    (e.g. value is outside of the domain of a function), or
  - when a resource it relies on is unavailable
    (like a missing file),

  routines simply raise an exception with a hopefully meaningful message:

```
CALL finish(subroutine,"my_process_component is unknown")
```

Runscript and job execution



Runscript and job execution

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

To run ICON it is necessary to create a runscript that sets required environment variables and calls the executable.

Slurm scheduler basics: The Slurm workload manager mediates between the ICON software and the HPC cluster, ie. the management of a job queue and the allocation of cluster nodes.

Slurm directives (may) appear as header lines in a runscript:

```
#SBATCH --partition=compute
#SBATCH --job-name=icon-lam
#SBATCH --nodes=16
#SBATCH --ntasks-per-node=48
#SBATCH --cpus-per-task=2
```

Most of your jobs will be submitted this way:

- sbatch <your_batch_script>

- srun <executable> is the task launcher for slurm
- squeue is used to show the queue
- scancel is used to cancel (i.e. kill) a job.

DWD's NEC: PBS batch system, qstat, qsub, …

Distributed-memory parallelization: MPI

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Model performance is determined by balance of workload, communication and memory consumption.

Domain decomposition

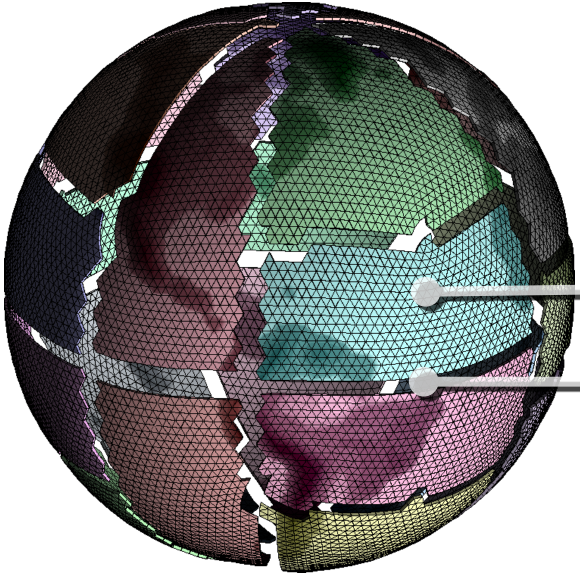
Each MPI task operates on a separate partition

Static load balancing

Partitioning fixed at start-up

Geometric subdivision

Recursive latitude/longitude bisection



Interior of partition

Halo region

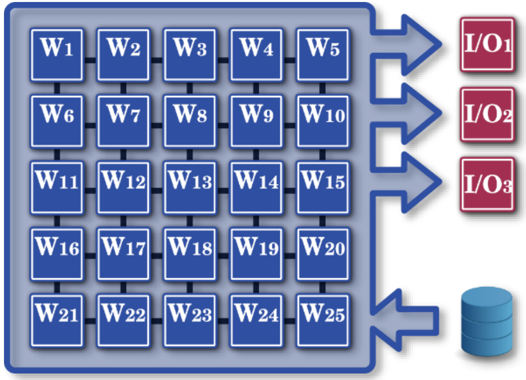
Finite difference stencils require communication

ICON

Splitting of the MPI process pool

Processors (PEs) are divided into

- **Worker PEs** majority of MPI tasks, doing the actual work
- **Output PEs** dedicated I/O server tasks
- **Restart PEs** for asynchronous restart writing
- **Prefetch PE** for ICON-LAM boundary data



ICON namelist settings for MPI

`num_io_procs`, `num_restart_procs` `num_prefetch_proc`

Asynchronous output: Some MPI processes run exclusively for writing data. Computation and output overlap.

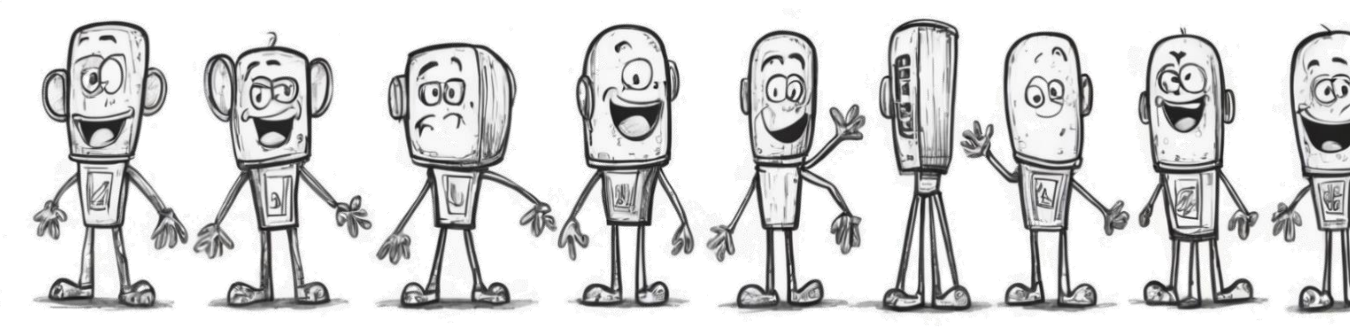
ICON-LAM: One MPI process runs exclusively for reading boundary data. Boundary data is linearly interpolated between two boundary data samples.

Levels of parallelism, target platforms

Besides distributed-memory computing other modes of parallel execution are possible:

- multi-core processors - OpenMPI — shared memory, multiple register sets
- classical vector computers — operate efficiently of large one-dimensional arrays
- Graphics Processing Units (GPUs) — compute kernels run on thousands of cores.

Not mutually exclusive; hybrid approaches are possible.



ICON on GPUs (Graphics Processing Units)

OpenACC is an API for offloading programs from a host CPU to an accelerator device.
ICON-Atmosphere has been gradually ported to GPUs using OpenACC.

ICON-CH on GPUs operational since May 28, 2024.

Critical settings:

- **`nproma`** ≫ number of arithmetic units on the accelerator
  (explanation will follow later)
- data transfers between CPU and GPU must be avoided.

Tutorial Book
Section 8.5

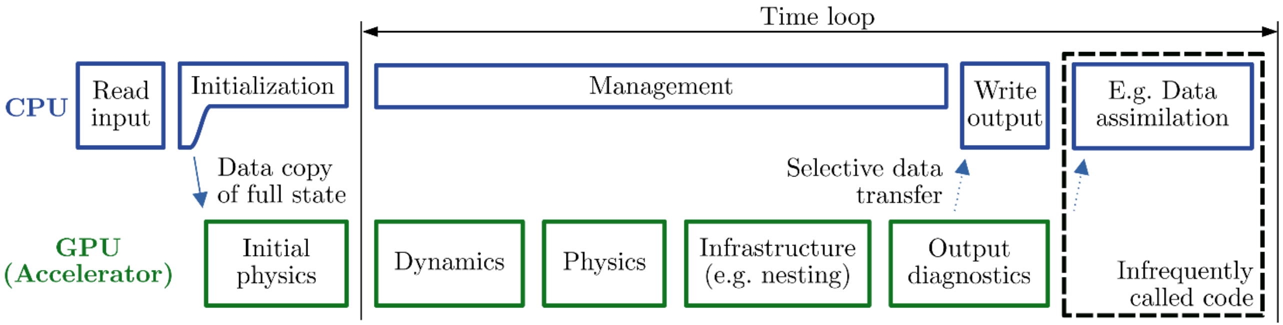


Illustration: M. Jacob, DWD

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.

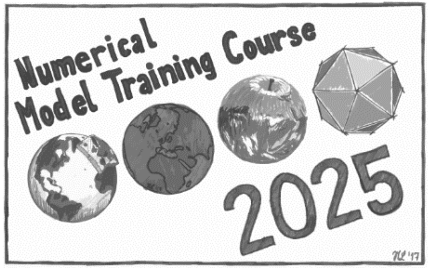
```
----------------------------  -------             -------------  -------------
name                          # calls             total min (s)  total max (s)
----------------------------  -------             -------------  -------------
total                         118         ...          272.564        272.637
 L integrate_nh               247800      ...          255.208        270.596
    L transport               49560       ...           33.409         35.791
       L adv_horiz            49560       ...           22.849         24.663
       L adv_vert             49560       ...            6.280          7.698
physics                       49678       ...          103.107        104.759
 L nwp_radiation              10030       ...           40.402         42.985
    L radiation               220674      ...           31.845         34.963
…
```

Caveat: Sometimes defined inconsistently by module developers!
For example, “total” does not contain ICON’s initialization.

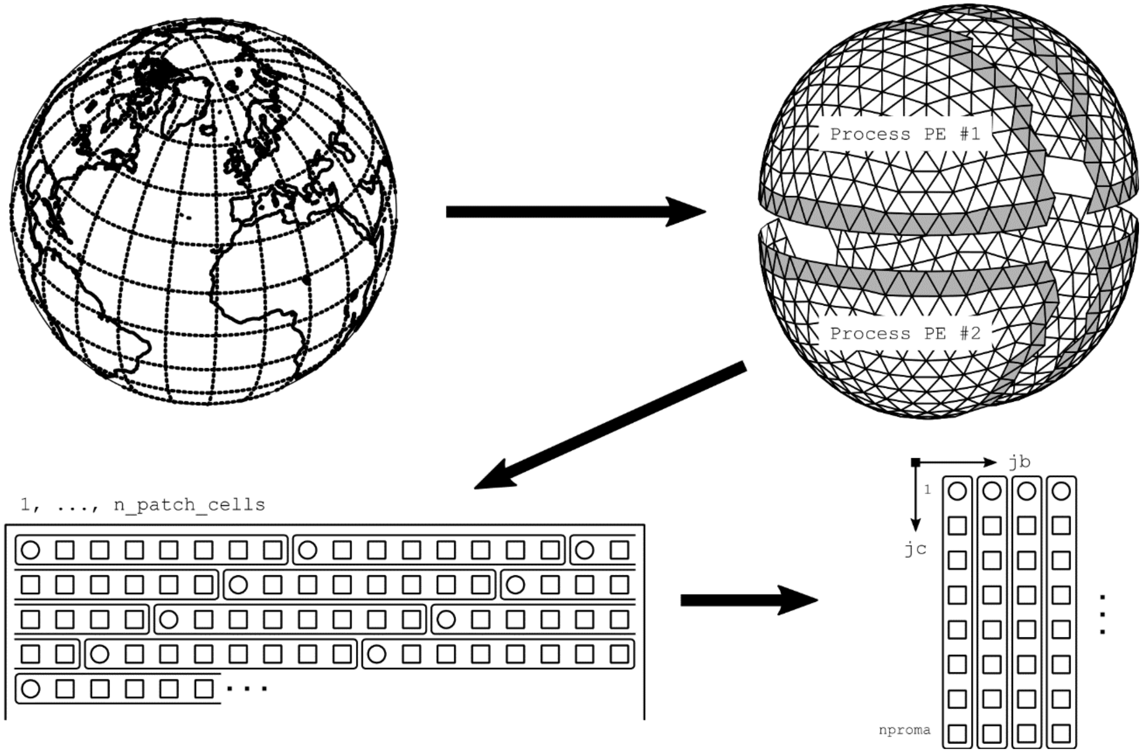
ICON namelist parameters

`run_nml::ltimer, timers_level`

Part 3: Code parts



Representation of 2D and 3D fields



Loop tiling: Arrays are logically two-dimensional in ICON

ICON namelist
`parallel_nml`
`nproma`

abbreviation **`nproma`** (probably) stands for
<u>n</u>ombre de <u>pro</u>fondeurs <u>ma</u>ximal
- maximum of maximal depths.

Physics and dynamics variables

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Prognostic and diagnostic fields are collected in **`t_nh_prog`** and **`t_nh_diag`**.
Elements of **`t_nh_prog`** are allocated for each time slice.

`t_nh_prog`

w	orthogonal vertical wind $[\mathrm{m\,s^{-1}}]$
vn	orthogonal normal wind $[\mathrm{m\,s^{-1}}]$
rho	density $[\mathrm{kg\,m^{-3}}]$
exner	Exner pressure
tke	turbulent kinetic energy $[\mathrm{m^2\,s^{-2}}]$
tracer	tracer concentration $[\mathrm{kg\,kg^{-1}}]$
	$\vdots$

Array dimensions:

[nproma, nlev, nblks_c]

`t_nh_diag`

u	zonal wind $[\mathrm{m\,s^{-1}}]$
v	meridional wind $[\mathrm{m\,s^{-1}}]$
temp	temperature $[\mathrm{K}]$
pres	pressure $[\mathrm{Pa}]$
	$\vdots$

src/atm_dyn_iconam/
mo_nonhydro_types.f90

Data structures: model domain

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

The **t_patch** data structure contains all information about grid coordinates and topology as well as parallel communication patterns and decomposition info.

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	lower-level data structures, see below
comm_pat_c/e/v	halo communication patterns, see Section 9.2.4
	⋮

src/shr_horizontal/
mo_model_domain.f90

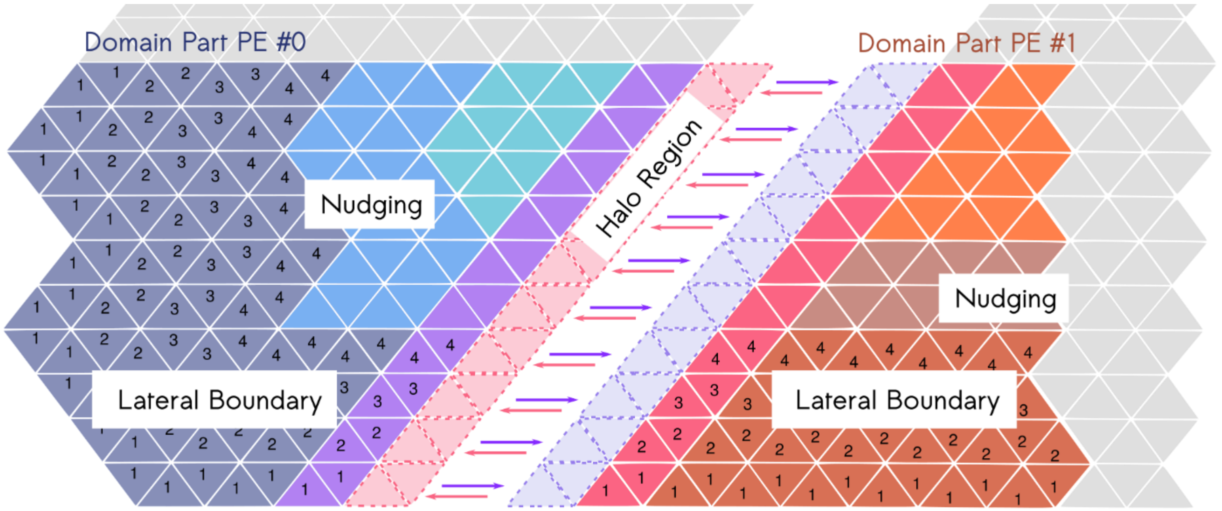
Index ordering and loop blocking

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Each PE performs a sorting of its local cells.

array portions for

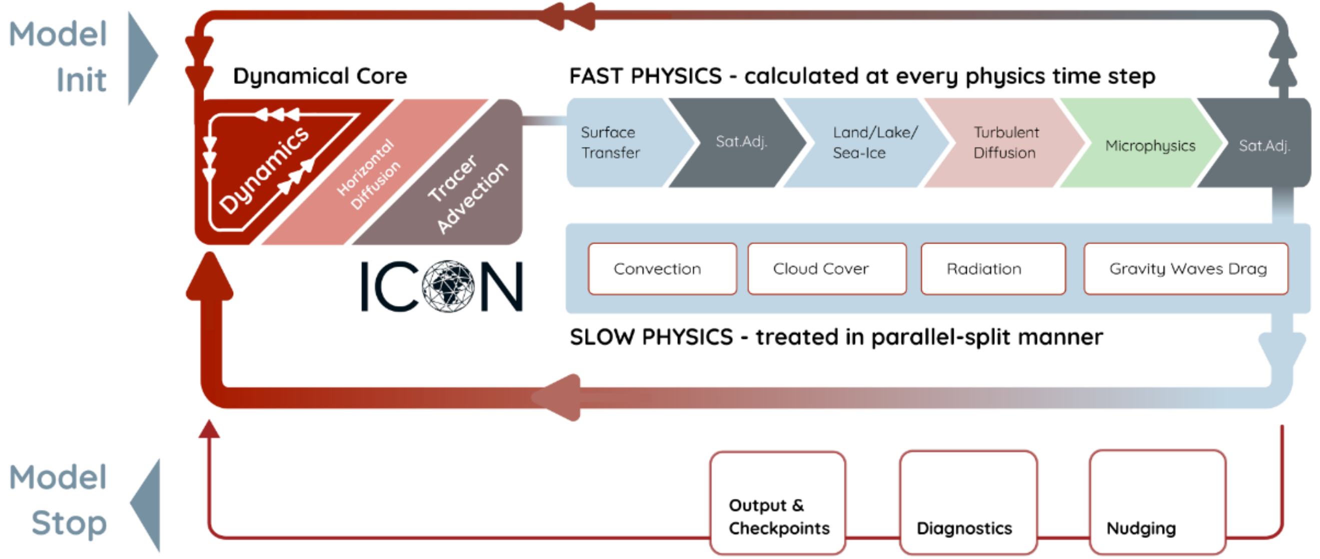
- lateral boundary and
- prognostic cells



`refin_c_ctrl(cell)` array:

Used to identify the nest boundary zone: cell rows are numbered starting from the grid boundary

Flow of control (revisited)



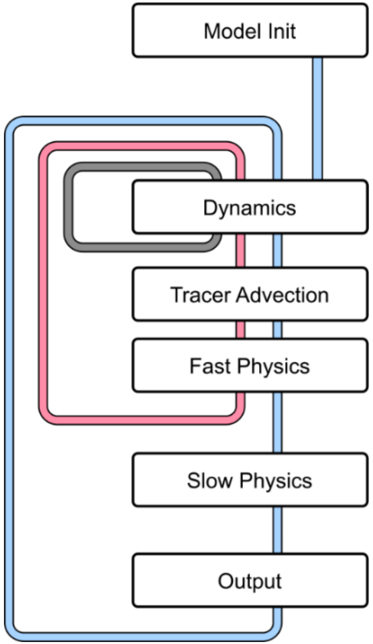
For efficiency reasons, different integration time steps are applied depending on the process.

Δt : basic time step (tracer, diffusion, fast physics) ; $\Delta\tau$: DyCore time step ; $\Delta t_{i,slow}$: slower processes

Physics-dynamics coupling

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

schematic



different integration
time steps applied:

dry NSE

act sequentially on
the atmospheric state

calculation of
independent tendencies

actual implementation

Physical parameterizations

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Process	Scheme	Settings
Radiation	RRTM (Rapid Radiative Transfer Model) Mlawer et al. (1997), Barker et al. (2003)	inwp_radiation=1
	ecRad Hogan and Bozzo (2018)	inwp_radiation=4
Non-orographic gravity wave drag	Wave dissipation at critical level Orr et al. (2010)	inwp_gwd=1
Sub-grid scale orographic drag	Lott and Miller scheme Lott and Miller (1997)	inwp_sso=1
Cloud cover	Diagnostic PDF *M. Köhler et al. (DWD)*	inwp_cldcover=1
	All-or-nothing scheme (grid-scale clouds)	inwp_cldcover=5

Process	Scheme	Settings
Microphysics	Single-moment scheme Doms et al. (2011), Seifert (2008)	inwp_gscp=1, 2
	Double-moment scheme Seifert and Beheng (2006)	inwp_gscp=4
Convection	Mass-flux shallow and deep Tiedtke (1989), Bechtold et al. (2008)	inwp_convection=1
Turbulent transfer	Prognostic TKE (COSMO) Raschendorfer (2001)	inwp_turb=1
	EDMF-DualM (Eddy-Diffusivity/Mass-Flux) Köhler et al. (2011), Neggers et al. (2009)	inwp_turb=3
	3D Smagorinsky diffusion (for LES)	inwp_turb=5
Land	Tiled TERRA Schrodin and Heise (2001), Schulz et al. (2016)	inwp_surface=1
	Flake: Mironov (2008)	llake=.TRUE.
	Sea-ice: Mironov et al. (2012)	lseaice=.TRUE.

- see physics lectures this week

ICON

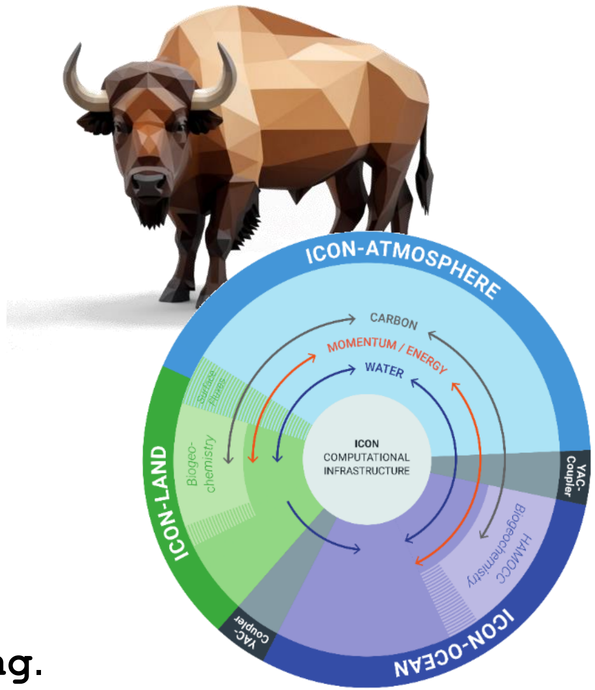
YAC coupler library

YAC (Yet Another Coupler) is used internally to exchange data between ICON's components, e.g.

- atmospheric model
- ocean model
- wave model
- hydrological discharge model
- CLEO microphysics
- alternative output writing

Implementation: see **`externals/yac`** and **`src/coupling`**.

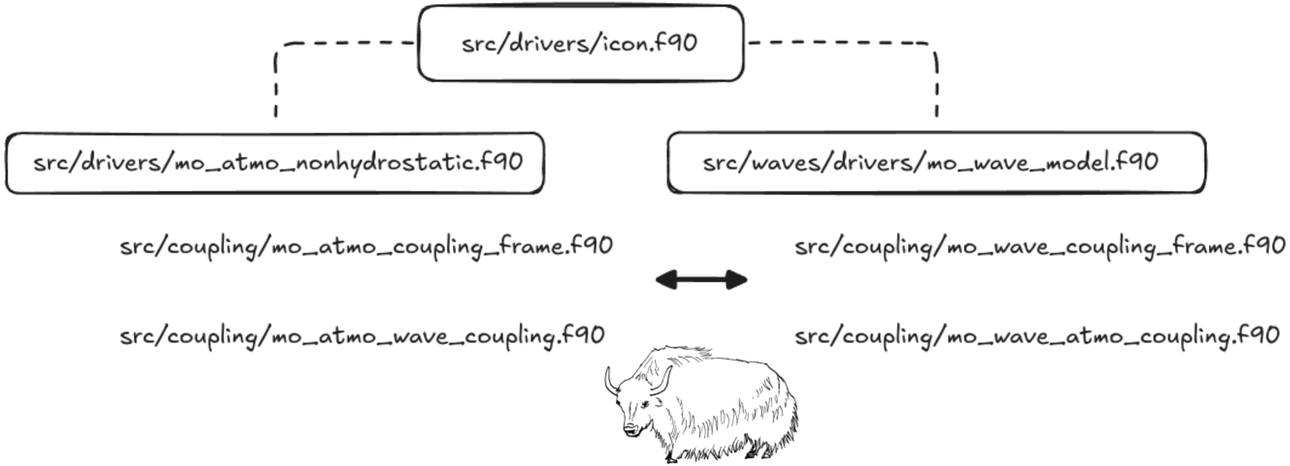
Developed and maintained by DKRZ, currently v3.6.2.



Source: D. Klocke, MPI-M

Coupler implementation

Example: Interface between ocean surface waves and atmosphere, through a coupler.
Coupling implemented in pairs of modules:



See also: Dobrynin, M., & co-authors. (2025, April 28). ICON-waves, a new ocean surface waves component of the ICON modeling framework [presentation]. EGU General Assembly 2025, Vienna, Austria.

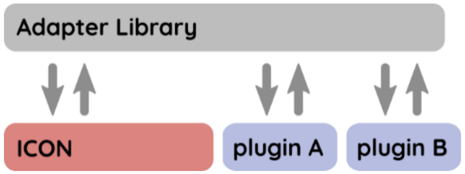
ComIn: ICON Community Interface

A plugin mechanism which

- connects 3rd party modules to the ICON host model
- plugin functions are called at pre-defined events
- access and creation of model variables

Shared libraries for Fortran or C/C++ plugins.

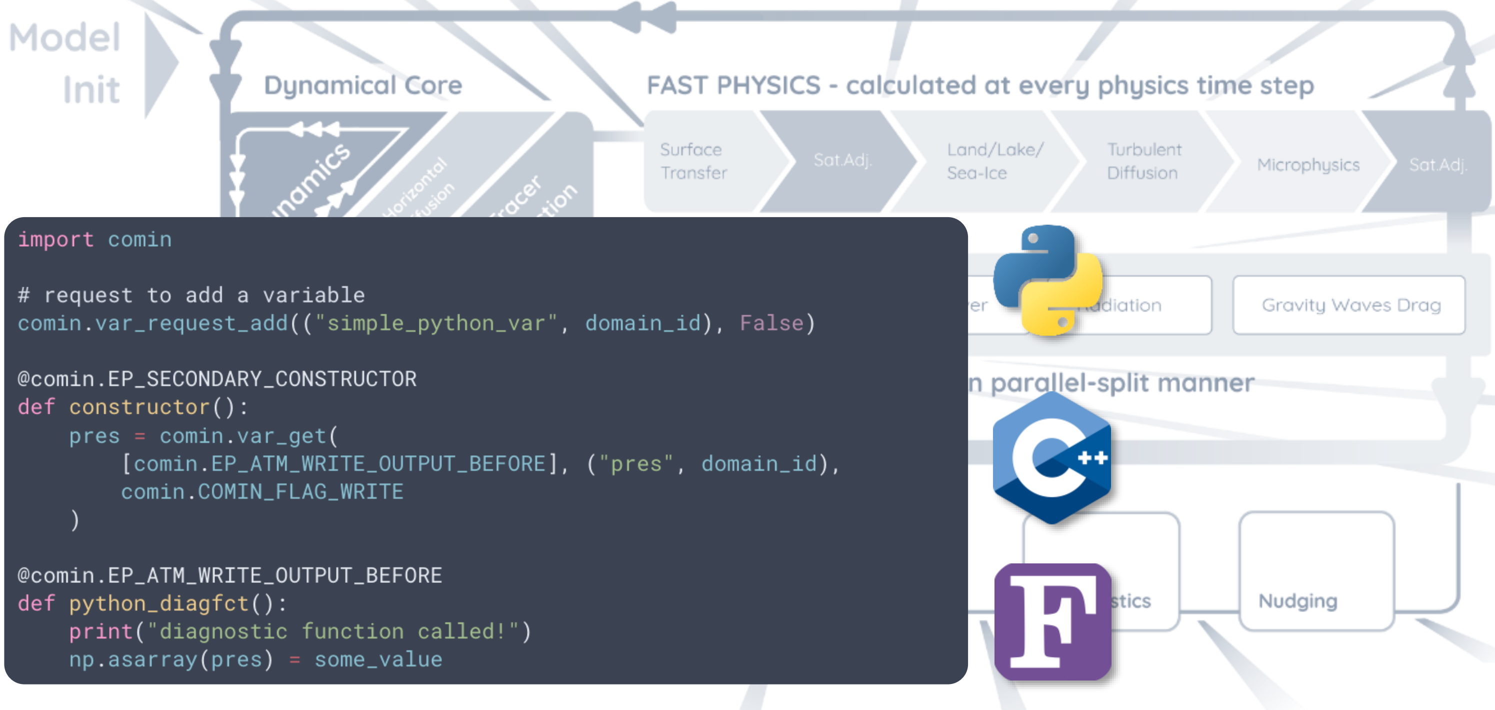
Python plugins do not need any compilation process at all.



Project started 2022 as a collaboration between
DWD, **DLR-IPA** and **DKRZ**.

ComIn is now part of the Open Source Release of ICON

ComIn: ICON Community Interface

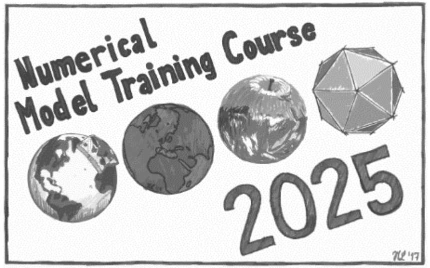



```python
import comin

# request to add a variable
comin.var_request_add(("simple_python_var", domain_id), False)

@comin.EP_SECONDARY_CONSTRUCTOR
def constructor():
    pres = comin.var_get(
        [comin.EP_ATM_WRITE_OUTPUT_BEFORE], ("pres", domain_id),
        comin.COMIN_FLAG_WRITE
    )

@comin.EP_ATM_WRITE_OUTPUT_BEFORE
def python_diagfct():
    print("diagnostic function called!")
    np.asarray(pres) = some_value
```

Example: Create and access model variables.

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Visualization and data processing



ICON

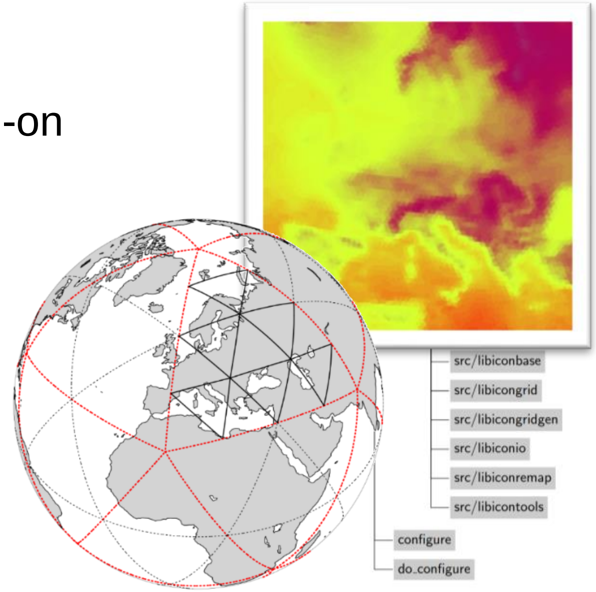
DWD ICON Tools

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

The ICON Tools are a set of command-line utilities
for remapping, extracting and querying data files.

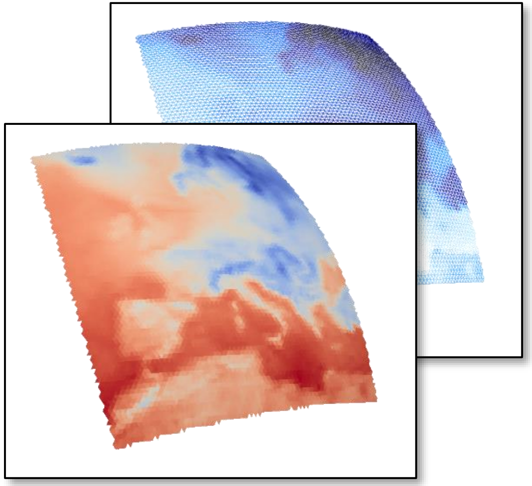
- not part of the open source release; developed as an add-on
- based on a Fortran 2003 library “libicontools.a”
  that is also used by the fieldextra COSMO software

- ICONREMAP utility: horizontal remapping,
  widely used to pre-process initial and boundary data.
- consider the Climate Data Operators (CDOs) as a
  powerful alternative!
  … see later slide …



ICONSUB utility (DWD ICON Tools)

- **SUB**grid extraction:
  cut subregions out of given data set.
- GRIB2 and NetCDF support;
  writes full topology of sub-grid.



Subregion specification

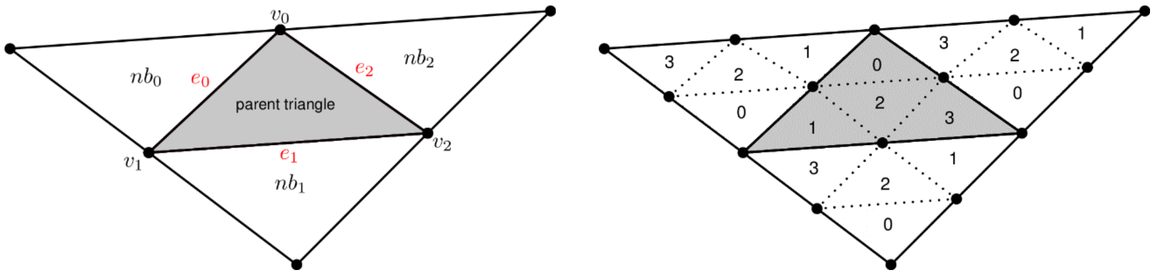
- south-west/north-east corner & rotation pole
- or: boundary strip (driving ICON-LAM)

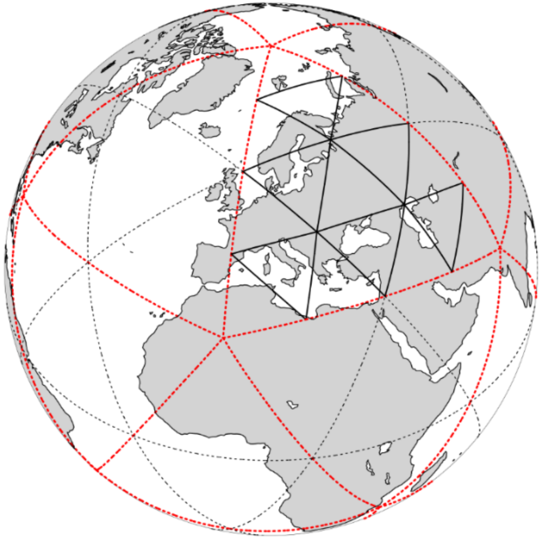
ICONSUB runs behind the scenes on DWD's HPC to provide datasets to NHMSs.

ICONGRIDGEN utility (DWD ICON Tools)

Simple grid generator: creates/refines ICON input grids

- generation of grid topology and child-to-parent relations
- generation process without storage of global grids





ICONGRIDGEN is primarily used as an online tool (web service).

Climate data operators (CDO)

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

CDOs are a collection of tools for NetCDF and GRIB data.

- based on CDI, developed and maintained at MPI-M
- source code and documentation available from
  https://code.mpimet.mpg.de/projects/cdo

Example: Dataset Information

```
> cdo sinfov test.nc

File format: netCDF2
 -1 : Institut Source   Ttype    Levels Num  Gridsize Num Dtype : Parameter name
  1 : ECMWF    unknown  instant      90   1     11324   1  F32  : T
  2 : ECMWF    unknown  instant       1   3     11324   1  F32  : PMSL
Grid coordinates :
…
```

Operator description can be obtained with `cdo -h [operator]`

Remapping with the CDOs

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

The CDO are also capable of remapping data to regular grids and triangular grids.

- recipes for ICON: https://github.com/deutscherwetterdienst/regrid

```
module load cdo
cdo -f nc2 remapcon,$LOCALGRID:2 \
     -selname,$FIELDS \
     -setgrid,$INPUTGRID:2 $DATAFILE output.nc
```

1st order conservative interpolation

Caveat: GRIB2 metadata sometimes mangled by CDO.

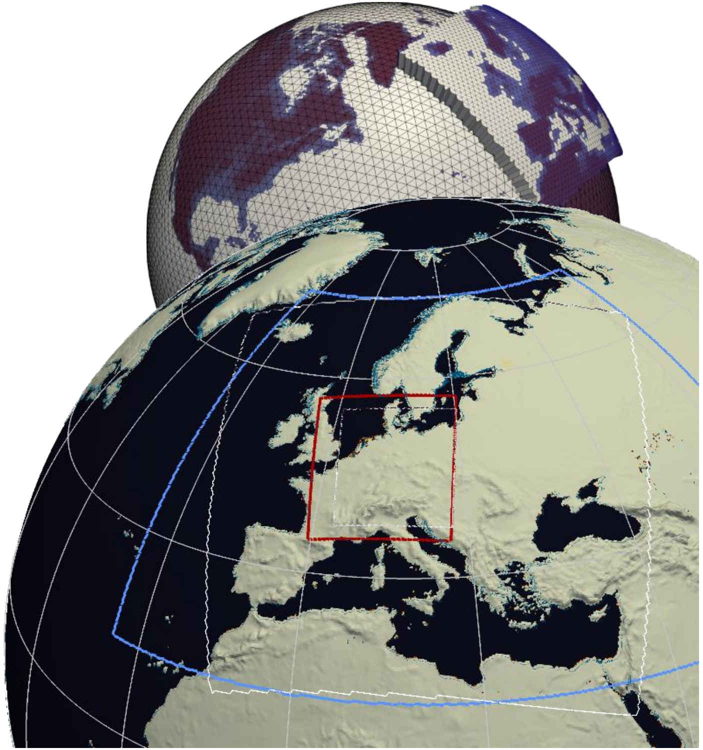
However, compared to, e.g., the DWD ICON Tools:
CDOs are freely available, well-maintained, feature-rich.

Visualization tools

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

The ICON Tutorial book (Chapter 10) presents quick start examples for

- NCAR Command Language (NCL)
- R (statistical computing environment)
- "Generic Mapping Tools" (GMT)
- the Python programming language …



[Visualization with ParaView, another open viz package]

Plotting with Python

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

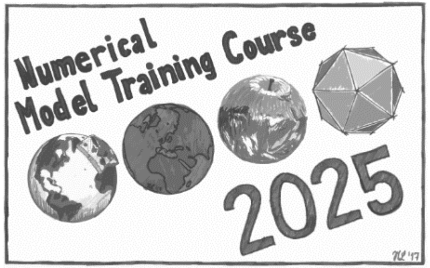
Introductory example using **matplotlib**, a plotting library for the Python programming language.

The **cartopy** package extends the Matplotlib functionality and offers map projection definitions.

```
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())
```

draw filled contours on unstructured grid

Wrap-Up



Wrap-Up

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

ICON

Introduction → Configure & Building → Source code overview → Namelist setup → Model output → Parallel execution → Data structures in ICON → Data processing & visualization

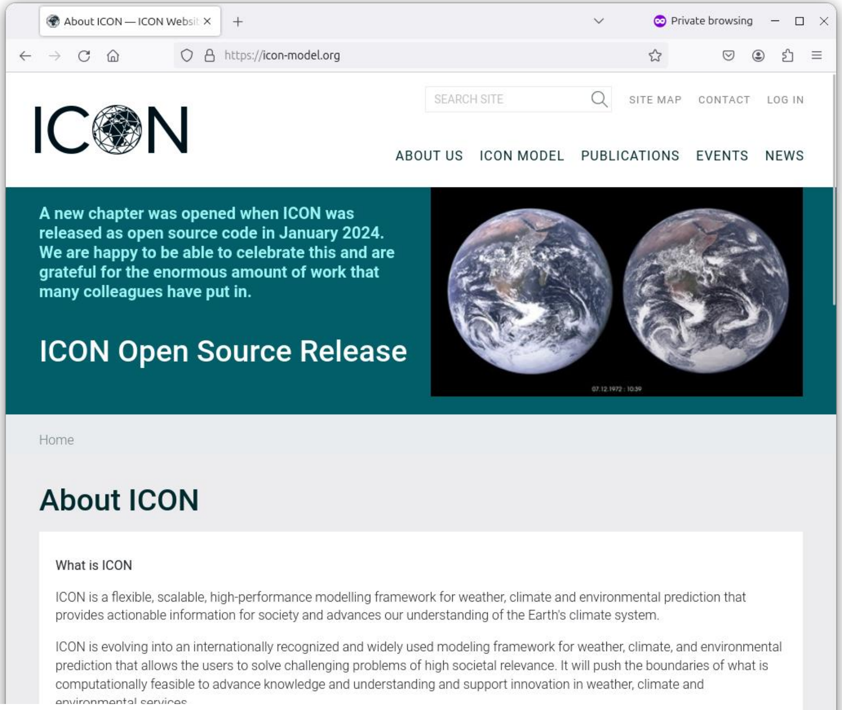
ICON Open Source Release

Since January 31, 2024, the source code of the ICON model has been released to the public under **the BSD-3C Open Source License**.
https://www.icon-model.org/

Moreover:

Special ICON-COSMO support license
for national weather services
icon-support@cosmo-model.org

ICON development takes place in Git repositories (restricted access).



https://icon-model.org – 2024-05-28

Questions?

Florian Prill

Met. Analyse und Modellierung
Deutscher Wetterdienst

e-mail: Florian.Prill@dwd.de