COSMO-Model Tutorial

February 2013



Working with the COSMO-Model

Practical Exercises

for NWP and CLM Mode

For NWP Mode
Ulrich Schättler
Ulrich Blahak
Michael Baldauf
Alexander Smalla
Cosmin Barbu
Rodica Dumitrache

For CLM Mode
Susanne Brienen
Kristina Trusilova
Burkhardt Rockel
Merja Tölle
Andreas Will





Contents

1 Installation of the COSMO-Model Package **1**

1.1 Introduction . . . 1

1.1.1 The COSMO-Model Package . . . 1

1.1.2 External Libraries needed by the COSMO-Model Package . . . 2

1.1.3 Computer Platforms for the COSMO-Model Package . . . 3

1.1.4 Necessary Data to run the System . . . 3

1.1.5 Practical Informations for these Exercises . . . 4

1.2 Installation of the External Libraries . . . 5

1.2.1 GRIB . . . 5

1.2.2 NetCDF . . . 7

1.3 Installation of the Programs . . . 7

1.3.1 The source code . . . 7

1.3.2 Adaptation of `Fopts` and the `Makefile` . . . 8

1.3.3 Compiling and linking the binary . . . 8

1.4 Installation of the reference-data . . . 8

1.5 Installation of the starter package for the climate mode . . . 9

1.6 Exercises . . . 10

2 Preparing External, Initial and Boundary Data **11**

2.1 Namelist Input for the INT2LM . . . 11

2.2 External Data . . . 11

2.3 The Model Domain . . . 12

2.3.1 The Horizontal Grid . . . 12

2.3.2 Choosing the Vertical Grid . . . 12

2.3.3 The Reference Atmosphere . . . 13

2.3.4 Summary of Namelist Group `LMGRID` . . . 14

2.4 Coarse Grid Model Data . . . 15
2.4.1 Possible Driving Models . . . 15
2.4.2 Specifying Regular Grids . . . 15
2.4.3 Specifying the GME Grid . . . 16
2.5 Specifying Data Characteristics . . . 16
2.5.1 External Parameters . . . 17
2.5.2 External Parameters for the Coarse Grid Input Model . . . 17
2.5.3 Coarse Grid Input Data . . . 17
2.5.4 COSMO Grid Output Data . . . 17
2.6 Parameters to Control the INT2LM Run . . . 18
2.6.1 Forecast Time, Domain Decomposition and Driving Model . . . 18
2.6.2 Basic Control Parameters . . . 19
2.6.3 Main Switch to choose NWP- or CLM-Mode . . . 19
2.6.4 Additional Switches Interesting for the CLM-Mode . . . 19
2.7 Examples for INT2LM Namelist Input Settings . . . 20
2.8 Exercises . . . 21
2.8.1 NWP-Mode . . . 21
2.8.2 CLM-Mode . . . 21

3 Running the COSMO-Model 23

3.1 Namelist Input for the COSMO-Model . . . 23
3.1.1 The Model Domain . . . 24
3.1.2 Basic Control Variables . . . 24
3.1.3 Settings for the Dynamics . . . 24
3.1.4 Settings for the Physics . . . 26
3.1.5 Settings for the I/O . . . 26
3.1.6 Settings for the Diagnostics . . . 27
3.2 Exercises . . . 27
3.2.1 NWP-Mode . . . 27
3.2.2 CLM-Mode . . . 28

4 Visualizing COSMO-Model Output using GrADS **29**

4.1 Introduction . . . 29

4.2 The data descriptor file . . . 29

4.3 The gribmap program . . . 31

4.4 A Sample GrADS Session . . . 31

4.4.1 Drawing a Map Background . . . 38

4.4.2 Use of GrADS in Batch mode . . . 40

4.5 Helpful Tools: `wgrib` and `grib2ctl` . . . 41

4.5.1 `wgrib` . . . 41

4.5.2 `grib2ctl` . . . 42

4.5.3 Exercise . . . 42

5 Postprocessing and visualization of NetCDF files **43**

5.1 Introduction . . . 43

5.2 ncdump . . . 43

5.3 CDO and NCO . . . 44

5.4 ncview . . . 44

5.5 Further tools for visualization and manipulation of NetCDF data . . . 45

5.6 ETOOL . . . 45

5.7 Exercises . . . 45

6 Running Idealized Test Cases **47**

6.1 General remarks . . . 47

6.1.1 Visualization . . . 48

6.1.2 gnuplot . . . 49

6.2 Exercises . . . 49

7 Troubleshooting for the COSMO-Model **53**

7.1 Compiling and Linking . . . 53

7.2 Troubleshooting in NWP-Mode . . . 54

7.2.1 Running INT2LM . . . 54

7.2.2 Running the COSMO-Model . . . 55

7.3 Troubleshooting in CLM-Mode . . . 56

8 Running COSMO-ART **57**

8.1 General remarks . . . 57
8.2 Source Code and Installation . . . 57
8.3 Exercises at DWD . . . 61
8.3.1 Dispersion of idealized area emissions . . . 62
8.3.2 Emission and dispersion of sea salt . . . 62
8.3.3 Dispersion of mineral dust particles . . . 62
8.3.4 Dispersion of gases and ozone formation or depletion . . . 63
8.3.5 Dispersion of aerosols and formation of secondary aerosol particles . . . 63
8.4 Exercises in Hamburg . . . 63
8.4.1 Dispersion of idealized area emissions . . . 63
8.4.2 Emission and dispersion of sea salt . . . 64
8.4.3 Dispersion of mineral dust particles . . . 64
8.4.4 Dispersion of gases and ozone formation or depletion . . . 64
8.4.5 Dispersion of aerosols and formation of secondary aerosol particles . . . 64
8.5 INT2LM-TRCR . . . 64

A The Computer System at DWD **1**

A.1 Available Plattforms at DWD . . . 1
A.2 The Batch System for the NEC . . . 2
A.3 NEC Run-Scripts for the Programs . . . 2

B The Computer System at DKRZ **1**

B.1 Available Plattforms at DKRZ . . . 1
B.2 The Batch System for the IBM . . . 1
B.3 Script environment for running the COSMO-CLM in a chain . . . 2
B.3.1 The scripts . . . 2
B.3.2 The environment variables . . . 4
B.3.3 Running the CCLM chain . . . 4
B.3.4 The directory structure . . . 5
B.3.5 Remarks . . . 5

C Necessary Files and Data for the NWP Mode **1**

C.1 The Source Codes . . . 1
C.2 External Parameters . . . 2
C.3 Driving Data . . . 4

D Necessary Files and Data for the Climate Mode **1**

D.1 The Source Codes 2

D.2 External Parameters 2

D.3 Initial and Boundary Data 2

D.4 Preprocessor and model system overview 3

Chapter 1

Installation of the COSMO-Model Package

1.1 Introduction

1.1.1 The COSMO-Model Package

The COSMO-Model Package is a regional numerical weather prediction system. It is based on the *COSMO-Model* (formerly known as LM), a nonhydrostatic limited-area atmospheric prediction model. Besides the forecast model itself, a number of additional components such as data assimilation, interpolation of boundary conditions from the driving model, and pre- and post-processing utilities are required to run the COSMO-Model in numerical weather prediction-mode (NWP-mode), in climate mode (CLM-mode) or for idealized case studies. The source code for the COSMO-Model Package consists of the following two programs:

- The interpolation program INT2LM
  The INT2LM interpolates data from different data sets to the rotated latitude-longitude grid of the COSMO-Model. Thus it provides the initial and / or boundary data necessary to run the COSMO-Model. Data from the global models GME (icosahedral grid point model of DWD), IFS (spectral model of ECMWF) and the regional COSMO-Model itself can be processed directly. For the climate mode, processing of a *common data format* from global data sets has been implemented. Thus, data from ECHAM5, ERA15, ERA40, NCEP and some other models can be processed. An extra pre-pre-processor is needed to convert these data to the *common data format*.

- The nonhydrostatic limited-area COSMO-Model
  The COSMO-Model is a nonhydrostatic limited-area atmospheric prediction model. It has been designed for both operational numerical weather prediction (NWP) and various scientific applications on the meso-β and meso-γ scale. In the last years a climate mode has been implemented which also allows for long-term simulations and regional climate calculations. From Version 4.0 onwards, the NWP-mode and the climate mode share the same source code. The COSMO-Model is based on the primitive thermo-hydrodynamical equations describing compressible flow in a moist atmosphere. The model equations are formulated in rotated geographical coordinates and a generalized terrain-following height coordinate. A variety of physical processes are taken into account by parameterization schemes. Also included in the model is a continuous data assimilation scheme for the atmospheric variables based on the nudging method.

The purpose of this tutorial is to give you some practical experience in installing and running the COSMO-Model Package. Exercises can be carried out on the supercomputers at DWD and / or DKRZ, resp. The principal steps of the installation can directly be transferred to other systems.

1.1.2 External Libraries needed by the COSMO-Model Package

There are some tasks, for which the COSMO-Model package needs external libraries:

- Data input and output:
  Reading and writing data from or to disk: Two data formats are implemented in the package: GRIB1 and NetCDF. Implementation of the new GRIB standard, GRIB2, is underway.

  · The GRIB1 library `libgrib1.a`
  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 consist of a continuous bit-stream made of a sequence of octets (1 octet = 8 bits). Currently, Version 1 of the GRIB is used. A special library, the grib1-library, is necessary to pack / unpack data in / from grib code. This library also containes the C-routines for performing the input and output for the programs.
  The grib1-library is provided by DWD. Because the COSMO-Model package uses some DWD-specific extensions to the GRIB1-Code, only the GRIB1 library provided by DWD will work with the models.

  · The NetCDF library `libnetcdf.a`
  NetCDF (Network Common Data Form) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. NetCDF files contain the complete information about the dependent variables, the history and the fields themselves. A special library, the NetCDF-library is necessary to pack / unpack data in / from NetCDF. This library also contains routines for manipulating and visualizing the data (nco-routines). Additionally, CDO-routines written at the Max-Planck-Institute for Meteorology in Hamburg may be used. These routines provide additional features like the harmonics or the vorticity of the wind field. For more information on NetCDF see `http://www.unidata.ucar.edu`.
  If the NetCDF library is not available, a file `dummy_netcdf.f90` is provided, that can be compiled and linked with the model, to satisfy the external references. I/O using the NetCDF data format is not possible then.

- Processing of observations and additional computations in the data assimilation mode (the nudging): `libmisc.a` and `libnetcdf.a`
  From Version 4.5 on, a new interface is used to read and pre-process observation data from NetCDF files instead of an AOF (Analysis Observation Format) file. Therefore, the `libnetcdf.a` is also necessary for running the nudging with observation processing.

  For Versions 4.21 and less, another library, the `libmisc.a` was necessary to do some standard-atmosphere computations. From Version 4.22 on, these computations have been included in the model. Therefore this library is not needed any more.

  To activate the nudging, the COSMO-Model has to be compiled with the preprocessor directive `NUDGING`. If this directive is not specified, the source code for the data assimilation mode is not compiled. To run the nudging is not possible then.

- Computing synthetic satellite images (radiances and brightness temperatures for several satellites / instruments):

- RTTOV7: `librttov7.a`
  Since Version 3.7 the COSMO-Model contains an interface to the RTTOV7-library (Radiative Transfer Model). This interface has been developed at the DLR Institute for Atmospheric Physics in Oberpfaffenhofen. Together with the RTTOV7-library it is possible to compute *synthetic* satellite images (brightness temperatures and radiances) derived from model variables for Meteosat5-7 and Meteosat Second Generation. The RTTOV7-library has been developed by UKMO et.al. in the framework of the ESA NWP-SAF. It has been modified at DWD to be used in parallel programs.

  To activate the interface for RTTOV7, the model has to be compiled with the preprocessor directive `RTTOV7`.

- RTTOV9: `librttov_ifc.a`, `librttov9.3.a`, `librttov9.3_parallel.a`
  In Version 4.18 also an interface to the newer RTTOV9-library has been installed. Besides the model-specific interface there is an additional model-independent interface, `mo_rttov_ifc`, which is also available from an external library `librttov_ifc.a`. For using RTTOV9, two additional external libraries `librttov9.3_parallel.a` and `librttov9.3.a` are also necessary.

  To activate the interfaces for RTTOV9, the model has to be compiled with the preprocessor directive `RTTOV9`.

- RTTOV10:
  Work is going on currently to install an interface to RTTOV10. This will be finally released in COSMO-Model 5.0.

To use the RTTOV-libraries, a license is necessary.
For getting this license, please contact `nwpsaf@metoffice.gov.uk`.

If none of the directives `RTTOV7` or `RTTOV9` is specified, the RTTOV-libraries are not necessary. The computation of the synthetic satellite images is not possible then.

1.1.3 Computer Platforms for the COSMO-Model Package

For practical work, the libraries and programs have to be installed on a computer system. You should have some knowledge about Unix / Linux, Compilers and Makefiles for these exercises.

INT2LM and the COSMO-Model are implemented for distributed memory parallel computers using the *Message Passing Interface* (MPI), but can also be installed on sequential computers, where MPI is not available. For that purpose, a file `dummy_mpi.f90` has to be compiled and linked with the programs. A `Makefile` is provided with the source codes, where the compiler call, the options and the necessary libraries can be specified.

All the codes have already been ported to different computer platforms and may easily be adapted to other systems. If you want to port the libraries and programs, you have to adapt the compiler and linker options in the Makefiles (see Sections 1.2.1 and 1.3.2) and the machine dependent parts in the run-scripts (see Appendices A-3 and / or B-3, resp.)

1.1.4 Necessary Data to run the System

Besides the source codes of the COSMO-package and the libraries, data are needed to perform runs of the COSMO-Model. There are three categories of necessary data:

External Parameter Files

External parameters are used to describe the surface of the earth. These data include the orography and the land-sea-mask. Also, several parameters are needed to specify the dominant land use of a grid box like the soil type or the plant cover fraction.

For fixed domain sizes and resolutions some external parameter files for the COSMO-Model are available. For the NWP community these data are usually provided in GRIB-format, while the CLM community prefers the NetCDF format.

External parameter data sets can be generated for any domain on the earth. Depending on the raw data sets available up to now, the highest resolution possible is about 2 km (0.02 degrees). At DWD there is a tool `EXTPAR` for creating the external parameters. The CLM Community has a slightly different tool `PEP` (Preparation of External Parameters), which can use additional raw data sets. More information on this issue can be found in the COSMO-Model documentation.

Boundary Conditions

Because the COSMO-Model is a regional system, it needs boundary data from a driving model. For the NWP-mode it is possible to use the GME and the IFS as driving models. Also, the COSMO-Model data can be used as boundary data for a higher resolution COSMO run. The boundary data are interpolated to the COSMO-Model grid with the INT2LM.

For the climate mode, several driving models are possible. A pre-pre-processor transforms their output into a common format, which can be processed by the INT2LM. See additional documentation from the CLM community for this mode.

Initial Conditions

In an operational NWP-mode, initial conditions for the COSMO-Model are produced by Data Assimilation. Included in the COSMO-Model is the nudging procedure, that *nudges* the model state towards available observations. For normal test runs and case studies, the initial data can also be derived from the driving model using the INT2LM. This is generally also the way it is done for climate simulations since data assimilation is not used then.

1.1.5 Practical Informations for these Exercises

For practical work during this week, you need informations about the use of the computer systems and from where you can access the necessary files and data. Depending on whether you run the NWP- or the CLM-mode, these informations are differing. Please take a look to the following Appendices:

A: The Computer System at DWD
To get informations on how to use DWD's supercomputer and run test jobs using the NWP-runscripts.

B: The Computer System at DKRZ
To get informations on how to use DKRZ's supercomputer and run test jobs using the CLM-runscripts.

C: Necessary Files and Data for the NWP-Mode
To get informations from where you can access the source code and the data for the NWP-exercises.

D: Necessary Files and Data for the CLM-Mode
To get informations from where you can access the source code and the data for the CLM-exercises.

1.2 Installation of the External Libraries

1.2.1 GRIB

Introduction

The GRIB library has two major tasks:

- First, it has to pack the meteorological fields into GRIB code and vice versa. Because GRIB coded data consists of a continuous bit stream, the GRIB code library has to deal with the manipulation of bits and bytes, which is not straightforward in Fortran.
- The second task is the reading and writing of these bit streams. The corresponding routines are written in C, because this usually has a better performance than Fortran-I/O.

The bit manipulation and the Fortran-C interface (which is not standardized across different platforms) makes the GRIB library the most critical part of the COSMO-Model Package, because its implementation is machine dependent.

Source code of the grib library

The source code of the grib library and a Makefile are contained in the compressed tar-file `DWD-libgrib1_<date>.tar.gz` (or `.bz2`). By detaring (`tar xvf DWD-libgrib1_<date>.tar`) a directory `DWD-libgrib1_<date>` is created that contains a README and subdirectories `include` and `source`. In the `include`-subdirectory several header files are contained, that are included during the compilation to the source files. The `source`-subdirectory contains the source code and a Makefile.

For a correct treatment of the Fortran-C-Interface and for properly working with bits and bytes, the Makefile has to be adapted to the machine used. For the compiler-calls and -flags, several variables are defined which can be commented / uncommented as needed. For the changes in the source code, implementations are provided for some machines with `ifdef`s. If there is no suitable definition / implementation, this has to be added.

Adaptation of the Makefile

In the `Makefile`, the following variables and commands have to be adjusted according to your needs:

`LIBPATH`	Directory, to which the library is written (this directory again is needed in the Makefiles of INT2LM and COSMO-Model).

`INCDIR`	`../include`.
`AR`	This is the standard Unix archiving utility. If compiling and archiving on one computer, but for another platform (like working on a front-end and compiling code for a NEC SX machine), this also should be adapted.
`FTN`	Call to the Fortran compiler.
`FCFLAGS`	Options for compiling the Fortran routines in *fixed form*. Here the `-Dxxx` must be set.
`F90FLAGS`	Options for compiling the Fortran routines in *free form*.
`CC`	Call to the C-Compilers (should be `cc` most everywhere).
`CCFLAGS`	Options for compiling the C routines. Here the `-Dxxx` must be set to choose one of the `ifdef` variants.

Big Endian / Little Endian

Unix computers, which are available today, use so-called *Big Endian* processors, which means that they number the bits in a byte from 0 to 7. Typical PC-processors (Intel et al.) do it the other way round and number the bits from 7 to 0. Because of this, the byte-order has to be swapped when using a PC or a Linux machine. This byte-swapping is implemented in the grib library and can be switched on by defining `-D__linux__`.

Compiling and creating the library

`make` or also `make all` compiles the routines and creates the library `libgrib1.a` in `LIBPATH`. 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 (typically 8).

Running programs using the GRIB library

All programs that are doing GRIB I/O, have to be linked with the library `libgrib1.a`. There are two additional issues that have to be considered when running programs using this GRIB library:

- DWD still uses a GRIB file format, where all records are starting and ending with additional bytes, the so-called *controlwords*. An implementation of the GRIB library is prepared that also deals with pure GRIB files, that do not have these controlwords. But still we guarantee correct execution only, if these *controlwords* are used. To ensure this you have to set the environment variable

  ```
  export LIBDWD_FORCE_CONTROLWORDS=1.
  ```

  in all your run-scripts.

- Another environment variable has to be set, if INT2LM is interpolating GME data that are using ASCII bitmap files:

  ```
  export LIBDWD_BITMAP_TYPE=ASCII.
  ```

  (because this library can also deal with binary bitmap files, which is the default).

1.2.2 NetCDF

NetCDF is a freely available library for reading and writing scientific data sets. If the library is not yet installed on your system, you can get the source code and documentation (this includes a description how to install the library on different platforms) from
`http://www.unidata.ucar.edu/software/netcdf/index.html`

Please make sure that the F90 package is also installed, since the model reads and writes data through the F90 NetCDF functions.

For this training course, the path to access the NetCDF library on the used computer system is already included in the `Fopts` and / or `Makefile`.

1.3 Installation of the Programs

1.3.1 The source code

The files `int2lm_yymmdd_x.y.tar.gz` and `cosmo_yymmdd_x.y.tar.gz` (or `.bz2`) contain the source codes of the INT2LM and the COSMO-Model, resp. `yymmdd` describes the date in the form "Year-Month-Day" and `x.y` gives the version number as in the DWD Version Control System (VCS). Since between major model updates the code is developed in parallel by the DWD and the CLM community, the code names are slightly different for COSMO-CLM where a number `z` is added: `int2lm_yymmdd_clmz` and `cosmo_yymmdd_x.y_clmz`.

By detaring (`tar xvf int2lm_yymmdd_x.y.tar` or `tar xvf cosmo_yymmdd_x.y.tar`) a directory `int2lm_yymmdd_x.y` (or `cosmo_yymmdd_x.y`) is created which contains a Makefile together with some associated files and (for the NWP Versions) some example run-scripts to set the Namelist Parameters for certain configurations and to start the programs.

`DOCS`	Contains a short documentation of the changes in version `x`.
`Fopts`	Definition of the compiler options and also directories of libraries.
`LOCAL`	Contains several examples of `Fopts`-files for different computers.
`Makefile`	For compiling and linking the programs.
`runxxx2yy`	Scripts to set the Namelist values for a configuration from model xxx to application yy and start the program.
`src`	Subdirectory for the source code.
`obj`	Subdirectory where the object files are written.
`Dependencies`	Definition of the dependencies between the different source files.
`Objfiles`	Definition of the object files.
`work`	Subdirectory for intermediate files.

Generally, only the `Fopts` part needs to be adapted by the user for different computer systems.

1.3.2 Adaptation of `Fopts` and the `Makefile`

In `Fopts`, the variables for compiler-calls and -options and the directories for the used libraries have to be adapted. In the subdirectory `LOCAL`, some example `Fopts`-files are given for different hardware platforms. If there is nothing suitable, the variables have to be defined as needed.

`F90`	Call to the Fortran (90) compiler.
`LDPAR`	Call to the linker to create a parallel binary.
`LDSEQ`	Call to the linker to create a sequential binary without using the Message Passing Interface (MPI).
`PROGRAM`	The binary can be given a special name.
`COMFLG1`	
`...`	Options for compiling the Fortran routines of the COSMO-Model.
`COMFLG4`	
`LDFLG`	Options for linking.
`AR`	Name of the archiving utility. This usually is `ar`, but when using the NEC cross compiler it is `sxar`.
`LIB`	Directories and names of the used external libraries.

1.3.3 Compiling and linking the binary

With the Unix command `make exe` (or just `make`), the programs are compiled and all object files are linked to create the binaries. On most machines you can also compile the routines in parallel by using the GNU-make with the command `gmake -j` *np*, where *np* gives the number of processors to use (typically 8).

1.4 Installation of the reference-data

The file `reference_data.tar(.bz2)` contains a GME reference data set together with the ASCII output, created on the NEC of DWD. Included are GME output data from the 20th of January, 2013, 12 UTC, for up to 12 hours together with a GME bitmap. For working with GME bitmap data see also Section 2.1. By detaring (`tar xvf reference_data.tar`) the following directory structure is created:

`COSMO_2_input`	Binary input data for the COSMO-Model (2.8 km resolution; only selected data sets available for comparison with your results)
`COSMO_2_output`	Binary output data from the COSMO-Model (2.8 km resolution; only selected data sets available for comparison with your results)
`cosmo_2_output_ascii`	ASCII reference output from the COSMO-Model (2.8 km resolution) together with a run-script `run_cosmo_2`.

`COSMO_7_input`	Binary input data for the COSMO-Model (7 km resolution; only selected data sets available for comparison with your results)
`COSMO_7_output`	Binary output data from the COSMO-Model (7 km resolution; only selected data sets available for comparison with your results)
`cosmo_7_output_ascii`	ASCII reference output from the COSMO-Model (7 km resolution) together with a run-script `run_cosmo_7`.
`GME_2013012012_sub`	Binary GME GRIB1 input data for the INT2LM together with a bitmap.
`int2lm_2_output_ascii`	ASCII reference output from the INT2LM run that transforms the COSMO-Model 7 km data to the 2.8 km grid together with a run-script `run_cosmo7_2_cosmo2`.
`int2lm_7_output_ascii`	ASCII reference output from the INT2LM run that transforms the GME data data to the 7 km COSMO-Model grid together with a run-script `run_gme_2_cosmo7`.
`README`	More information on the reference data set.

The characteristics of this data set are: Start at 20. January 2013, 12 UTC + 12 hours.

Application	Resolution km / degrees	grid size	`startlat_tot`	`startlon_tot`
COSMO-7	7 / 0.0625	129×161×40	-5.0	-4.0
COSMO-2	2.8 / 0.025	121×141×50	-3.5	-3.5

1.5 Installation of the starter package for the climate mode

For the climate simulation, all the necessary code and input files to start working with the COSMO-CLM are included in the *starter package* which is available on the CLM community web page. After unpacking the archive (`tar -xzvf cclm-sp.tgz`) you find the source code, initial and boundary data, external parameters and run scripts as described in Appendix B and D. Note that the starter package requires the netCDF4 library linked with HDF5 and zip libraries and extended by the Fortran netCDF package. The netCDF4 package comes with the programs `ncdump` and `nccopy`. If you are using the starter package on another computer than `blizzard` make sure that these programs are installed correctly[1].

The compilation of the libraries and binaries is very similar to that for the NWP mode (Sections 1.2–1.3). In this course, we need to compile the GRIB library and the INT2LM and COSMO-CLM binaries. The NetCDF library is already available on the `blizzard` computer. An additional package of some small auxiliary fortran programs is needed which is called CFU.

In all cases you have to type `'make'` in the respective subdirectory to compile the codes. For INT2LM and CCLM you have to adopt the `Fopts` file and set your working directory (`SPDIR`) first. An initalization script called `init.sh` helps with these adoptions. It is therefore useful to run this script once directly after unpacking the whole package. On computer systems other than the

[1] software available from http://www.unidata.ucar.edu/software/netcdf/index.html

`blizzard` at DKRZ you have to be careful to make the necessary additional changes in the `Fopts` and `Makefile`. Some of these options are already available in the files.

More information on the setup of the model can be found on the CLM-community webpage. If you are logged in as community member you see a page called "Support" where you can find the starter package, the WebPEP tool for generating external data files, a namelist tool with detailed information on the namelist parameters of the model and links to the bug reporting system (RedC) and some other utilities.

1.6 Exercises

NWP-Mode:

EXERCISE:
Install the grib-library, the INT2LM and the COSMO-Model on your computer and make a test run using the reference data set. For that you have to do:

- Copy the source files to your home directory and the reference data set to your work- or scratch-directory (see Appendix C-1).
- Install the libraries and programs according to Sections 1.2.1, 1.3 and 1.4.
- Run the INT2LM and the COSMO-Model for (at least) the 7 km resolution. Copy the corresponding run-scripts to the directories where you installed INT2LM and the COSMO-Model, resp.. You only have to change some directory names and the machine you want to run on (hopefully). See Appendix A-3 for an explanation of the run-scripts.

CLM-Mode:

EXERCISE:
Install the COSMO-CLM starter package on your computer and compile the necessary programs. For that you have to do:

- Copy the starter package to your scratch-directory (see Appendix D).
- Install the library and programs according to Sections 1.2.1, 1.3 and 1.5.

Chapter 2

Preparing External, Initial and Boundary Data

In this lesson you should prepare the initial and boundary data for a COSMO-Model domain of your choice. You will learn how to specify some important Namelist variables for the INT2LM.

2.1 Namelist Input for the INT2LM

The execution of INT2LM can be controlled by 6 NAMELIST groups:

`CONTRL`	parameters for the model run
`GRID_IN`	specifying the domain and the size of the GME grid
`LMGRID`	specifying the domain and the size of the LM grid
`DATABASE`	specification of database job
`DATA`	controlling the grib input and output
`PRICTR`	controlling grid point output

All NAMELIST groups have to appear in the input file `INPUT` in the order given above. Default values are set for all parameters, you only have to specify values, that have to be different from the default. The NAMELIST variables can be specified by the user in the run-script for the INT2LM, which then creates the `INPUT` file. See Appendices A-3 (NWP) and / or B-3 (CLM) for details on the run scripts.

2.2 External Data

Before you can start a test run, you have to specify a domain for the COSMO-Model and get some external parameters for this domain. How the domain is specified is explained in detail in Sect. 2.3. Because of interpolation reasons the external parameter must contain at least one additional row of grid points on all four sides of the model domain.

For the first exercises, you have to take one of the external data sets specified in Appendix C (for NWP) or Appendix D (for CLM). In the future, if you want to work with different domains, you can contact DWD or the CLM-Community to provide you an external parameter set for your domain.

2.3 The Model Domain

This section explains the variables of the Namelist group `LMGRID` and gives some theoretical background.

To specify the domain of your choice, you have to choose the following parameters:

1. The lower left grid point of the domain in rotated coordinates.
2. The horizontal resolution in longitudinal and latitudinal direction in degrees.
3. The size of the domain in grid points.
4. The geographical coordinates of the rotated north pole, to specify the rotation.
5. The vertical resolution by specifying the vertical coordinate parameters.

All these parameters can be determined by Namelist variables of the Namelist group `LMGRID`. In the following the variables and their meaning is explained.

Note: For describing the bigger domain of the external parameters, you have to specify values for 1) - 4) above. A vertical resolution cannot be given for the external parameters, because they are purely horizontal.

2.3.1 The Horizontal Grid

The COSMO-Model uses a rotated spherical coordinate system, where the poles are moved and can be positioned such that the equator runs through the centre of the model domain. Thus, problems resulting from the convergence of the meridians can be minimized for any limited area model domain on the globe.

Within the area of the chosen external parameter data set, you can specify any smaller area for your tests. Please note that you have to choose at least one grid point less on each side of the domain of the external parameters because of interpolation reasons.

The Namelist variables to specify the horizontal grid are:

`startlat_tot = -17.0,`	lower left grid point in rotated coordinates
`startlon_tot = -12.5,`	
`dlon=0.0625, dlat=0.0625,`	horizontal resolution of the model
`ielm_tot=225, jelm_tot=269,`	horizontal size of the model domain in grid points
`pollat = 40.0, pollon = -170.0,`	geographical coordinates of rotated north pole

2.3.2 Choosing the Vertical Grid

To specify the vertical grid, the number of the vertical levels must be given together with their position in the atmosphere. This can done by the Namelist variables `kelm_tot` and by defining the list of vertical coordinate parameters $\sigma_1, \ldots, \sigma_{\texttt{ke}}$.

This is the most difficult part of setting up the model domain, because a certain knowledge about the vertical grid is necessary. The COSMO-Model equations are formulated using a terrain-following

coordinate system with a generalized vertical coordinate. Please refer to the Scientific Documentation (Part I: Dynamics and Numerics) for a full explanation of the vertical coordinate system. For practical applications, three different options are offered by INT2LM to specify the vertical coordinate. The options are chosen by the Namelist variable `ivctype`:

1. A pressure based coordinate:
   The σ values are running from 0 (top of the atmosphere) to 1 (surface),
   e.g. $\sigma_1 = 0.02, \ldots, \sigma_{\texttt{ke}} = 1.0$.

2. A height based coordinate:
   The σ values are given in meters above sea level,
   e.g. $\sigma_1 = 23580.4414, \ldots, \sigma_{\texttt{ke}} = 0.0$.

3. A heigth based SLEVE (Smooth LEvel VErtical) coordinate:
   The σ values are specified as in 2. In addition some more parameters are necessary. Please refer to the documentation for a full specification of the SLEVE coordinate.

All these coordinates are hybrid coordinates, i.e. they are terrain following in the lower part of the atmosphere and change back to a pure z-coordinate (flat horizontal lines with a fixed height above mean sea level) at a certain height. This height is specified by the Namelist variable `vcflat`, which is also specified pressure based or height based.

Most operational applications now use 40 levels together with coarser horizontal resolutions (up to 7 km) and 50 levels for very high horizontal resolution runs (about 2-3 km).

Summary of the Namelist variables regarding the vertical grid:

`ivctype=1,`	to choose the type of the vertical coordinate
`vcoord_d=`$\sigma_1, \ldots, \sigma_{\texttt{ke}}$`,`	list of vertical coordinate parameters to specify the vertical grid structure
`vcflat=0.22,`	height, where levels change back to z-system
`kelm_tot=40,`	vertical size of the model domain in grid points

Because the specification of the vertical coordinate parameters σ_k is not straightforward, INT2LM offers some pre-defined sets of coordinate parameters. For `ivctype=1` you can choose `kelm_tot` to be 20, 32 or 40. For `ivctype=2` you can choose `kelm_tot` to be 50. Then the list of vertical coordinate parameters `vcoord_d`= $\sigma_1, \ldots, \sigma_{\texttt{ke}}$ and `vcflat` are set by the INT2LM.

2.3.3 The Reference Atmosphere

In the COSMO-Model, pressure p is defined as the sum of a base state (p_0) and deviations from this base state (p'). The base state (or reference state) p_0 is prescribed to be horizontally homogeneous, i.e. depending only on the height above the surface, is time invariant and hydrostatically balanced. To define the base state, some other default values like sea surface pressure (p_{SL}) and sea surface temperature (T_{SL}) are needed. With these specifications, the reference atmosphere for the COSMO-Model is computed. For a full description of the reference atmosphere, please refer to the Scientific Documentation (Part I: Dynamics and Numerics)

The COSMO-Model offers 2 different types for the reference atmosphere.

1: For the first reference atmosphere a constant rate of increase in temperature with the logarithm of pressure is assumed. Therefore, the reference atmosphere has a finite height and the top of the model domain has to be positioned below this maximum value in order to avoid unrealistical low reference temperatures in the vicinity of the upper boundary.

2: The second reference atmosphere is based on a temperature profile which allows a higher model top:

$$t_0(z) = (t_{0sl} - \Delta t) + \Delta t \cdot \exp\left(\frac{-z}{h_{scal}}\right),$$

where $z =$ `hhl(k)` is the height of a model grid point. If this option is used, also the values for $\Delta t =$ `delta_t` and $h_{scal} =$ `h_scal` have to be set.

Which type of reference atmosphere is used, is chosen by the Namelist variable `irefatm = 1/2`. Some additional characteristics of the chosen atmosphere can also be set. For further details see the User Guide for the INT2LM (Part V: Preprocessing, Sect. 7.3: *Specifying the Domain and the Model Grid*).

The values of the reference atmosphere are calculated analytically on the model half levels. These are the interfaces of the vertical model layers, for which most atmospheric variables are given. Because the values are also needed on the full levels (i.e. the model layers), they also have to be computed there. For the first reference atmosphere (`irefatm=1`) there are 2 different ways how to compute the values on the full levels. This is controlled by the namelist switch `lanalyt_calc_t0p0`:

- `lanalyt_calc_t0p0=.FALSE.`
  The values on the full levels are computed by averaging the values from the half levels. This has to be set for the Leapfrog dynamical core (`l2tls=.FALSE.` in the COSMO-Model) or for the old Runge-Kutta fast waves solver (`l2tls=.TRUE.` and (`itype_fast_waves=1` in the COSMO-Model).

- `lanalyt_calc_t0p0=.TRUE.`
  The values on the full levels are also calculated analytically on the full levels. This has to be set for the new Runge-Kutta fast waves solver (`l2tls=.TRUE.` and (`itype_fast_waves=2` in the COSMO-Model).

For `irefatm=2` the values on the full levels are always calculated analytically.

2.3.4 Summary of Namelist Group `LMGRID`

Example of the Namelist variables in the group `LMGRID`, that are necessary for these exercises:

```
&LMGRID
 startlat_tot = -17.0, startlon_tot = -12.5,
 dlon=0.0625, dlat=0.0625,
 ielm_tot=225, jelm_tot=269, kelm_tot=40,
 pollat = 40.0, pollon = -170.0,
 ivctype=1,
 irefatm=2,
 delta_t=75.0,
 h_scal=10000.0,
/
```

2.4 Coarse Grid Model Data

This section explains most variables of the Namelist group `GRID_IN`.

2.4.1 Possible Driving Models

With the INT2LM it is possible to interpolate data from several driving models to the COSMO grid. Up to now the following input models are supported in NWP mode:

- GME: Global model from DWD with an icosahedral model grid
- IFS: Global spectral model from ECMWF; data is delivered on a Gaussian grid
- COSMO: Data from a coarser COSMO-Model run can be used to drive higher resolution runs

In the CLM mode, additional driving models are possible:

- NCEP: Data from the National Center for Environmental Prediction (USA)
- ERA40 or ERA-Interim: Data from the ECMWF Reanalysis
- ECHAM5/6: Data from the global model of MPI Hamburg
- HADAM: Data from the Hadley Centre, (UK)
- REMO: Data from the regional climate model REMO, MPI Hamburg.

The data of these models are pre-pre-processed, to have the data in a common format, which can be processed by the INT2LM.

2.4.2 Specifying Regular Grids

The Namelist group `GRID_IN` specifies the characteristics of the grid of the driving model chosen. Most input models use a rectangular grid which can be specified by the following Namelist variables:

`startlat_in_tot`	latitude of lower left grid point of total input domain
`startlon_in_tot`	longitude of lower left grid point of total input domain
`dlon_in`	longitudinal resolution of the input grid
`dlat_in`	latitudinal resolution of the input grid

`ie_in_tot`	longitudinal grid point size
`je_in_tot`	latitudinal grid point size
`ke_in_tot`	vertical grid point size
`pollat_in`	geographical latitude of north pole (if applicable)
`pollon_in`	geographical longitude of north pole (if applicable)

2.4.3 Specifying the GME Grid

The GME has a special horizontal grid derived from the icosahedron. The characteristics of the GME grid are specified with the variables:

`ni_gme=` `ni` stands for number of intersections of a great circle between two points of the icosahedron and determines the horizontal resolution.

`i3e_gme=` vertical levels of GME

In the last years the resolution of GME has been increased several times. Depending on the date you are processing, the above Namelist variables have to be specified according to the following table:

Date	`ni_gme`	`i3e_gme`
before 02.02.2010, 12 UTC	192	40
02.02.2010, 12 UTC - 29.02.2012, 00 UTC	256	60
29.02.2012, 12 UTC -	384	60

With the GME data (from all resolutions), you can do an interpolation to a COSMO grid resolution of about 7 km, NOT MORE. If you want to run a higher resolution, you have to choose COSMO-EU data or you first have to run a 7 km domain yourself and then go on to a finer resolution (which would be a nice exercise).

Working with GME bitmap data

DWD offers the possibility, not to provide the full GME data sets (which are very large) but only the subset that is needed to run the COSMO-Model for a special domain. To specify the grid points from GME, for which data are needed, bitmaps are used. If such GME data sets, that were created with a bitmap, are used, the same bitmap has to be provided to INT2LM, in order to put all data to the correct grid point. For providing the bitmap, the two NAMELIST parameters `ybitmap_cat` and `ybitmap_lfn` in the group `DATA` (see Sect. 2.5.3) have to be specified.

2.5 Specifying Data Characteristics

This section explains some variables of the Namelist group `DATA`. In this Namelist group the directories and some additional specifications of the data have to be specified.

For every data set you can specify the format to be Grib (`'grb1'`) or NetCDF (`'ncdf'`). Note that the Climate Community prefers NetCDF, while the NWP Community uses the Grib format.

2.5.1 External Parameters

The following variables have to be specified for the dataset of the external parameters:

`ie_ext=965, je_ext=773`	size of external parameters
`ylmext_lfn='cosmo_d5_07000_965x773.g1_2009121700'`	name of external parameter file
`ylmext_cat='/tmp/data/ext1/'`	directory of this file
`ylmext_form_read='grb1'`	to specify the data format (also: `'ncdf'`)

2.5.2 External Parameters for the Coarse Grid Input Model

The INT2LM also needs the external parameters for the coarse grid driving model. The following variables have to be specified for this dataset:

`yinext_lfn='invar.i384a',`	name of external parameter file for input model
`yinext_cat='/tmp/data/ext2/',`	directory of this file
`yinext_form_read='grb1',`	to specify the data format (`'grb1'` or `'ncdf'`)

2.5.3 Coarse Grid Input Data

The following variables have to be specified for the input data sets of the driving model:

`yin_cat='/tmp/data/input/',`	directory of input data
`ybitmap_cat='/tmp/data/btmp/',`	directory of the bitmap file (for GME only)
`ybitmap_lfn='bitmp384'`	name of the bitmap file (for GME only)
`yin_form_read='grb1',`	to specify the data format (`'grb1'` or `'ncdf'`)
`ytunit_in='f',`	to specify the time unit and hence the file name of the input data
`yinput_type='forecast',`	to specify the time unit and hence the file name of the input data

2.5.4 COSMO Grid Output Data

The following variables have to be specified for the output data sets of the COSMO-Model:

`ylm_cat= '/tmp/data/out/',`	directory of the results
`ylm_form_write='grb1',`	data format for output data
`ytunit_out='f',`	to specify the time unit and hence the file name of the output data

There are different variants to specify the file name for the input and output data sets. Again, the NWP- and the CLM-Communities use different settings here, which are specified by the variables `ytunit_in` and `ytunit_out`, resp. While for NWP applications mainly the default `'f'` is used, the CLM-Community takes the setting `'d'`. For a more detailed explanation of these settings see the User Guide for the INT2LM (Part V: Preprocessing, Sect. 6.5: *Conventions for File Names*).

2.6 Parameters to Control the INT2LM Run

This section explains only few basic variables of the Namelist group `CONTRL`. Most of the parameters in this group are used to specify a certain behaviour of the INT2LM, depending on special demands for the COSMO-Model. Please refer to the User Guide for the INT2LM (Part V: Preprocessing, Sect. 7.1) for a full documentation of all variables.

2.6.1 Forecast Time, Domain Decomposition and Driving Model

The following parameters have to be specified in any case and need no longer explanation.

To specify the date and the length of the forecast

`ydate_ini`	date when the forecast begins
`hstart`	start hour relative to the start of the forecast
`hstop`	stop hour relative to the start of the forecast
`hincbound`	increment for boundary update given in hours

To specify which data should be computed

`linitial`	compute initial data
`lboundaries`	compute boundary data

To specify the driving model

`yinput_model`	the name of the driving model is specified as a character string. The following names are possible: `'GME'`, `'IFS'`, `'COSMO'`, `'CM'`.

This variable has been introduced in INT2LM Version 1.14. In former versions, logical variables had to be specified instead:

`lgme2lm`	for nesting in the GME
`llm2lm`	for nesting in the COSMO-Model
`lec2lm`	for nesting in the IFS
`lcm2lm`	for nesting in a global climate model

Note, that exactly one (and not more) of these logical variables must be `.TRUE.`

To specify the number of processors

`nprocx`	number of processors in X-direction (longitudinal)
`nprocy`	number of processors in Y-direction (latitudinal)
`nprocio`	number of processor for asynchronous IO (usually 0)

2.6.2 Basic Control Parameters

Depending on the COSMO-Model configuration, additional initial and / or boundary data has to be provided. To control this, the following Namelist parameters can be used:

`lforest`	provide additional external parameters for the distribution of evergreen and deciduous forest
`lsso`	provide additional external parameters to run the subgrid scale orography scheme
`lprog_qi`	interpolate cloud ice from the driving model
`lprog_qr_qs`	interpolate rain and snow from the driving model
`lmulti_layer_in`	use multi-layer input soil levels
`lmulti_layer_lm`	compute multi-layer output soil levels
`llake`	provide additional external parameters for the FLake model
`lprog_rho_snow`	interpolate prognostic snow density from the driving model
`lfilter_oro`	to apply a filter for the orography
`lfilter_pp`	to apply a filter for the pressure deviation (after vertical interpolation)
`lbalance_pp`	to compute a hydrostatic balanced pressure deviation (after vertical interpolation)

2.6.3 Main Switch to choose NWP- or CLM-Mode

The main switch to choose between a NWP or a CLM application is (in both models, INT2LM and the COSMO-Model), the Namelist variable `lbdclim`. For the NWP-mode, this variable is `.FALSE.` If it is set to `.TRUE.`, additional boundary data will be provided for the COSMO-Model for every output set. These are the slowly varying variables like sea surface temperature, plant cover, leaf area index, etc. In a typical NWP application, which is running only for few days, these variables are held constant. This, of course, cannot be done in a climate application.

`lbdclim`	to run in climate mode

2.6.4 Additional Switches Interesting for the CLM-Mode

`itype_w_so_rel`	to select the type of relative soil moisture input
`itype_t_cl`	to select the source for the climatological temperature
`itype_rootdp`	to select the treatment of the external parameter for the root depth
`itype_ndvi`	to select the treatment of plant cover and leaf area index
`itype_calendar`	to specify a gregorian calendar or a calendar where every year has 360 days
`luse_t_skin`	to use skin temperature for surface
`l_cressman`	to use a Cressman scheme during 'M'atch interpolation

Again, for a full documentation of these parameters please refer to the User Guide for the INT2LM (Part V: Preprocessing, Sect. 7.1).

2.7 Examples for INT2LM Namelist Input Settings

Here are some examples of NAMELIST variables for the INT2LM:

For the first namelist group, we give example values, that are used at DWD for the application *COSMO_EU* (running with 7 km over nearly the whole of Europe) and *COSMO_DE* (running with 2.8 km over Germany).

&CONTRL

Variable	for gme2eu	for eu2de	for CLM-Mode	
lreorder=	.FALSE.	.FALSE.	.FALSE.	
lgme2lm=	.TRUE.	.FALSE.	.FALSE.	old: $\leq$ 1.13
llm2lm=	.FALSE.	.TRUE.	.FALSE.	old: $\leq$ 1.13
lec2lm=	.FALSE.	.FALSE.	.FALSE.	old: $\leq$ 1.13
lcm2lm=	.FALSE.	.FALSE.	.TRUE.	old: $\leq$ 1.13
yinput_model=	'GME'	'COSMO'	'CM'	new: $\geq$ 1.14
llake=	.FALSE.	.FALSE.	.FALSE.	
lforest=	.TRUE.	.TRUE.	.TRUE.	
lsso=	.TRUE.	.FALSE.	.TRUE.	
lbdclim=	.FALSE.	.FALSE.	.TRUE.	
lprog_qi=	.TRUE.	.TRUE.	.TRUE.	
lprog_qrqs=	.TRUE.	.TRUE.	.FALSE.	
lmulti_layer_in=	.TRUE.	.TRUE.	.TRUE.	
lmulti_layer_lm=	.TRUE.	.TRUE.	.TRUE.	
lprog_rho_snow=	.TRUE.	.TRUE.	.FALSE.	
itype_w_so_rel	1	1	0	
itype_t_cl	0	0	1	
itype_rootdp	0	0	3	
itype_ndvi	0	0	0	
itype_calendar	0	0	0	
lfilter_oro=	.TRUE.	.TRUE.	.TRUE.	
lfilter_pp=	.FALSE.	.TRUE.	.FALSE.	
lbalance_pp=	.FALSE.	.TRUE.	.FALSE.	
eps_filter=	0.1	0.1	N.A.	
norder_filter=	1	5	N.A.	
ilow_pass_oro=	1	4	N.A.	
ilow_pass_xso=	0	5	N.A.	
rxso_mask=	0.0	625.0	N.A.	
luse_t_skin=	.FALSE.	.FALSE.	.TRUE.	

2.8 Exercises

To do the following exercises, you have to adapt some namelist input for the INT2LM. If necessary, take a look to the last sections, where the most important namelist parameters are given for special NWP- and CLM-applications.

2.8.1 NWP-Mode

EXERCISE:
Choose a domain and prepare the initial and boundary conditions for up to 48 hours for a 7 km resolution COSMO-Model grid. For reasons of memory and computer time, the domain size should not be bigger than about $400 \times 400 \times 50$ grid points.

- Choose an appropriate runscript from the INT2LM and adapt it to your needs.
- Choose an appropriate external parameter data set.
- Choose data from a driving model.

For available data sets see Appendix C-2 for external parameters and C-3 for driving model data.

2.8.2 CLM-Mode

EXERCISE:
Prepare the initial and boundary conditions for one month for a 0.44° resolution COSMO-CLM grid in the step-by-step mode using the NCEP input.

- Go to the directory `$SPDIR/step_by_step/gcm_to_cclm` and look at the run script for INT2LM (`run_int2lm`).
- Adapt the variable `SPDIR` (if not done yet with the `init.sh` script).
- Submit the job and look at the results (see Appendix B for how to submit jobs on `blizzard`).

For available data sets see Appendix D.2 for external parameters and D.3 for driving model data.

Chapter 3

Running the COSMO-Model

In this lesson you will learn how to run the COSMO-Model, using the initial and boundary data you produced in the last lesson.

3.1 Namelist Input for the COSMO-Model

The execution of the COSMO-Model can be controlled by several `NAMELIST` groups:

`LMGRID`	specifying the domain and the size of the grid
`RUNCTL`	parameters for the model run
`TUNING`	parameters for tuning physics and dynamics
`DYNCTL`	parameters for the adiabatic model
`PHYCTL`	parameters for the diabatic model
`DIACTL`	parameters for the diagnostic calculations
`NUDGING`	controlling the data assimilation
`INICTL`	parameters for the initialization of model variables
`IOCTL`	controlling the environment
`GRIBIN`	controlling the grib input
`GRIBOUT`	controlling the grib output
`SATCTL`	controlling computation of synthetic satellite images
`EPSCTL`	controlling ensemble runs

All NAMELIST groups have to appear in corresponding input files `INPUT_**` in the order given above. Default values are set for all parameters, you only have to specify values, that have to be different from the default. The NAMELIST variables can be specified by the user in the run-scripts for the COSMO-Model, which then create the `INPUT_**` files. See Appendices A-3 (NWP) and / or B-3 (CLM) for details on the run scripts.

We will only describe some basic variables here. For a detailed explanation see the User Guide for the COSMO-Model (Part VII: User Guide, Sect. 7).

3.1.1 The Model Domain

The model domain is specified by the variables in the Namelist group `LMGRID`. This group is similar to the corresponding group in the INT2LM, but has fewer variables. For example, in the COSMO-Model you cannot specify vertical coordinate parameters. These are given to the COSMO-Model by the initial data (via Grib or NetCDF headers).

These are the variables in `LMGRID` with some example settings:

`&LMGRID`

Variable	COSMO_EU	COSMO_DE	for CLM-Mode
ie_tot	665	421	101
je_tot	657	461	111
ke_tot	40	50	32
startlat_tot	-17.0	-5.0	-24.09
startlon_tot	-12.5	-5.0	-25.13
pollat	40.0	40.0	39.25
pollon	-170.0	-170.0	-162.00
dlon	0.0625	0.0625	0.44
dlat	0.0625	0.0625	0.44

3.1.2 Basic Control Variables

In this group, the basic control variables are set. Among them are the variables for specifying the date and the range of the simulation, and the number of processors for the domain decomposition. These variables have the same name and meaning as the corresponding variables in INT2LM, group `CONTRL` and are not explained again here.

Other variables are for specifying the time step, the number of boundary lines for the domain decomposition and which components should be used during the run of the COSMO-Model

These are some example settings:

`&RUNCTL`

Variable	COSMO_EU	COSMO_DE	for CLM-Mode	
dt	66.0	25.0	240.0	time step in seconds
nboundlines	3	3	3	number of boundlines
lphys	.TRUE.	.TRUE.	.TRUE.	to run physical parameterizations
luseobs	.TRUE.	.TRUE.	.FALSE.	to run with observation processing
ldiagnos	.TRUE.	.TRUE.	.TRUE.	to produce diagnostics
luse_rttov	.TRUE.	.TRUE.	.FALSE.	to compute syn. satellite images
lartif_data	.FALSE.	.FALSE.	.FALSE.	to run idealized test cases

3.1.3 Settings for the Dynamics

In the group `DYNCTL`, either the Leapfrog dynamical core (a 3 time level scheme), or the Runge-Kutta dynamical core (a 2 time level scheme: `l2tls` must be chosen. For both schemes, additional parameters can be specified.

In Version 4.24 a new fast-waves solver has been introduced, which has the following features:

- Improvement of the accuracy of vertical derivatives and averages.
- Use of the divergence operator in strong conservation form.
- Isotropic treatment of the artificial divergence damping.

These characteristics now allow for more stable integrations also in mountainous terrain with steeper slopes. The different fast-waves solver are chosen by the Namelist switch `itype_fast_waves = 1` (old solver) or `= 2` (new solver).

Usage of the Leapfrog dynamical core is no more recommended. Most COSMO applications have been ported to the Runge-Kutta dynamical core now. There are some differences between the COSMO_EU (with about 7 km resolution) and the COSMO_DE (with about 2.8 km resolution). The following tables give some values for the different applications.

`&DYNCTL`

Variable	COSMO_EU	COSMO_DE	for CLM-Mode	
l2tls	.TRUE.	.TRUE.	.TRUE.	to choose 2- or 3-tl scheme
lcond	.TRUE.	.TRUE.	.TRUE.	cloud water condensation
ldyn_bbc	.TRUE.	.TRUE.	.FALSE.	dynamical bottom boundary
lexpl_lbc	.TRUE.	.TRUE.	.TRUE.	explicit lateral bound. relaxation
rlwidth	85000.0	50000.0	500000.0	width of relaxation layer (*)
y_scalar_advect		'BOTT2_STRANG'		Bott advection
irunge_kutta	1	1	1	to select Runge-Kutta scheme
irk_order	3	3	3	order of Runge-Kutta scheme
iadv_order	5	5	5	order of advection scheme
lspubc	.TRUE.	.TRUE.	.TRUE.	sponge layer with damping
nrdtau	5	5	10	damping at model top

(*) The width of the relaxation layer should extend about 15 grid points. Thus it should be *resolution in meters * 15*.

Some variable names for specifying the horizontal diffusion have been changed in Version 4.12. Before, the horizontal diffusion was specified with the following variables (still true for the CLM-Mode).

itype_hdiff	2	2	2	type of horizontal diffusion
lhdiff_mask	.TRUE.	.TRUE.	.FALSE.	mask for horizontal diffusion
hd_corr_q	0.5	0.0	0.5	diffusion factor for q
hd_corr_t	0.75	0.75	0.75	diffusion factor for t
hd_corr_u	1.0	0.75	1.0	diffusion factor for u

From Version 4.12 onward, there are different diffusion factors for the boundary and the interior zone. This avoids the usage of a diffusion mask. Now there is also a diffusion factor for pressure. And in Version 4.26 the name of the diffusion factors for humidity changed from `hd_corr_q_[in,bd]` to `hd_corr_trcr_[in,bd]`, because these variables are now treated as tracers.

itype_hdiff	2	2	2	type of horizontal diffusion
hd_corr_u_in	0.25	0.0	-	diffusion factor (interior domain) for wind
hd_corr_u_bd	0.25	0.75	-	diffusion factor (boundary zone) for wind
hd_corr_t_in	0.0	0.0	-	diffusion factor (interior domain) for temperature
hd_corr_t_bd	0.0	0.75	-	diffusion factor (boundary zone) for temperature
hd_corr_p_in	0.0	0.0	-	diffusion factor (interior domain) for pressure
hd_corr_p_bd	0.0	0.75	-	diffusion factor (boundary zone) for pressure
hd_corr_trcr_in	0.0	0.0	-	diffusion factor (interior domain) for tracers
hd_corr_trcr_bd	0.0	0.0	-	diffusion factor (boundary zone) for tracers

3.1.4 Settings for the Physics

In the group PHYCTL, the different parametrizations can be switched on or off. Most parameterization packages have additional parameters for special settings. In &PHYCTL, the most important difference between the COSMO_EU and the COSMO_DE are in the settings of the convection scheme. While COSMO_EU uses the Tiedtke convection (itype_conv=0), the COSMO_DE uses a shallow convection scheme (itype_conv=3).

&PHYCTL

Variable	COSMO_EU	COSMO_DE	for CLM-Mode	
lgsp	.TRUE.	.TRUE.	.TRUE.	to run with microphysics
itype_gscp	3	4	3	to select special scheme
lrad	.TRUE.	.TRUE.	.TRUE.	to run with radiation
ltur	.TRUE.	.TRUE.	.TRUE.	to run with turbulence
lconv	.TRUE.	.TRUE.	.TRUE.	to run with convecton
itype_conv	0	3	0	to select special scheme
lsoil	.TRUE.	.TRUE.	.TRUE.	to run with soil model
ke_soil	7	7	9	to select special scheme
lsso	.TRUE.	.FALSE.	.TRUE.	to run with SSO scheme

3.1.5 Settings for the I/O

There are 4 Namelist groups to specify various settings for Input and Output, like to choose Grib or NetCDF as data format. In IOCTL, the basic settings, which are valid for all I/O, are specified.

Note also the special Namelist variable lbdclim, with which the climate mode of the COSMO-Model is activated.

&IOCTL

Variable	COSMO_EU	COSMO_DE	for CLM-Mode	
lbdclim	.FALSE.	.FALSE.	.TRUE.	to run the climate mode
yform_read	'grb1'	'grb1'	'ncdf'	to choose Grib or NetCDF
yform_write	'grb1'	'grb1'	'ncdf'	to choose Grib or NetCDF
ngribout	1	3	3/7	set number of different outputs

&DATABASE only needed if the database system of DWD is used

The settings of the following groups depend on the special job and there is no need to give different specifications here.

```
&GRIBIN
 ydirini='/gtmp/uschaett/gme2lm/',   directory of the initial fields
 ydirbd='/gtmp/uschaett/gme2lm/',    directory of the boundary fields
 hincbound=1.0,                      increment for lateral boundaries
 lchkini=.TRUE., lchkbd =.TRUE.,     additional control output
 lana_qi=.TRUE., llb_qi=.TRUE.,      cloud ice is provided for analyses and lateral
                                     boundaries

&GRIBOUT
 hcomb=0.0,48.0,1.0,                          start, stop and incr. of output
 lcheck=.TRUE.,
 yvarml='default',
 yvarpl='default',
 yvarzl='default',
 lwrite_const=.TRUE.,
 ydir='/gtmp/uschaett/lm_test/with_out/',     directory of the result data
```

3.1.6 Settings for the Diagnostics

The COSMO-Model can produce some diagnostic output for a quick monitoring of the forecast. Which output is generated can be specified in the group `DIACTL`.

```
&DIACTL
 n0meanval=0, nincmeanval=1,
 lgplong=.TRUE., lgpshort=.FALSE., lgpspec=.FALSE., n0gp=0, hincgp=1.0,
 igp_tot = 3, 13, 25, 46,
 jgp_tot = 31, 29, 96, 12,
 stationlist_tot= 0, 0, 46.700, 6.200, 'RODGAU',
                  0, 0, 50.050, 8.600, 'Frankfurt-Flughafen',
                  53, 68, 0.0 , 0.0 , 'My-Own-Station',
```

3.2 Exercises

3.2.1 NWP-Mode

EXERCISE:
Run the COSMO-Model using the initial and boundary conditions you produced in the last session.

- Choose an appropriate runscript for the COSMO-Model and adapt it to your needs. For coarser resolutions (up to 7 km), the run-scripts `run_cosmo_eu` or `run_cosmo_7` can be used. For higher resolutions (to 2-3 km), the run-scripts `run_cosmo_de` or `run_cosmo_2` can be used.
- Use the initial and boundary data, which you produced in the last exercise
- Run the COSMO-Model

3.2.2 CLM-Mode

EXERCISE:
Part I
Run the COSMO-CLM using the initial and boundary conditions you produced in the last session.

- Still in the step-by-step mode, adapt, if applicable, the corresponding run script for COSMO-CLM (`run_cclm`) to your needs (e.g. specifying the working directory `SPDIR`), using the initial and boundary data which you produced in the last exercise.
- Submit the job and look at the results.
- When successful, do a second simulation (both steps INT2LM and CCLM) for a smaller domain nested inside the European domain with a higher resolution of 0.0625°. For this, use the scripts `run_int2lm` and `run_cclm` in the directory `$SPDIR/step_by_step/cclm_to_cclm`.

Part II
Run the INT2LM and COSMO-CLM in the chain environment (see Appendix B.3).

- Go to the directory `$SPDIR/chain/gcm_to_cclm/sp001`. Change the paths in `subchain` and `templates/int2lm.job.tmpl`.
- Start the chain by typing `./subchain prep`.
- If the cclm simulation is successful you can find the output data in the directories `$SPDIR/chain/arch/sp001`
  and as post-processed time series in `$SPDIR/chain/work/sp001/post`.
- Similarly, you can do also the nesting simulation in the directory `$SPDIR/chain/cclm_to_cclm/sp002`.

Chapter 4

Visualizing COSMO-Model Output using GrADS

4.1 Introduction

The Grid Analysis and Display System (GrADS) is an interactive desktop tool that can be used to visualize data from meteorological models. GrADS implements a 4-dimensional data model, where the dimensions are latitude, longitude, level and time. Each data set is located within this 4-dimensional space by use of a data descriptor file. This describes the dimension environment by the user as a desired subset of the 4-dimensional space. Data is accessed, manipulated and displayed within this subset.

After specifying the data descriptor file, the `gribmap` program has to be run to build up a correspondance between the data set (usually a grib-file) and the description file. `gribmap` scans the given data set for the desired data and produces an additional file, called an `index`-file.

Once the data set has been scanned, data may be displayed using a variety of graphical output techniques, including line, bar and scatter plots, as well as contour, shaded contour, streamline and wind vector plots.

More information can be found on `http://grads.iges.org/grads/head.html`. This is the GrADS home page from where you can download all files, programs and documentations necessary to install GrADS on your local platform.

4.2 The data descriptor file

The data descriptor file has the suffix `.ctl` and specifies the subset of the 4-dimensional space. The user has to specify ranges for all four dimensions:

Dimension	Command	Example
Time	`tdef`	`tdef 13 linear 12Z20jan2013 1hr`
Longitudes	`xdef`	`xdef 325 linear 1 1`
Latitudes	`ydef`	`ydef 325 linear 1 1`
Vertical levels	`zdef`	`zdef 40 levels`

The full syntax for the `zdef` command is

`zdef` *number mapping* <*start increment*> <*value-list*>.

with

number the number of grid values in the z-direction, specified as an integer number.

mapping the mapping type, specified as a keyword. Valid keywords are

`linear` linear mapping.

`levels` arbitrary levels (e.g. pressure or σ-levels).

start when *mapping* is `linear`, this is the starting value.

increment when *mapping* is `linear`, this is the increment in the z direction.

value-list when *mapping* is `levels`, the specific levels are simply listed in ascending order.

The specification of the *value-list* for the COSMO-Model σ-levels is not straightforward. You will find data descriptor files for 40 and 50 vertical levels in

`/k/uhome/fegast3/COSMO_CLM_TRAINING_2013/grads`

(called `test_m40.ctl` and `test_m50.ctl`, resp.). There are also example data descriptor files for data on p- and on z-levels (called `test_p.ctl` and `test_z.ctl`, resp.).

Other necessary specifications for the data descriptor file are:

- The directory and the name of the data set, e.g.
  `dset /gtmp/uschaett/test_vampir/laf2002012300`
- The name of the index file, e.g.
  `index laf.idx`
- The number of the grid that is used. The COSMO-Model grid in rotated latitude-longitude coordinates is not defined in the official tables of GrADS, so this specification has to be
  `dtype grib 255` (which means a user-defined grid).
- A value that should be placed instead of undefined values in a record (if any).
  `undef 9.999E+20`

The third section is the specification of the variables that should be displayed. This section begins with `VARS` *nn*, where nn is the number of variables that are specified and ends with `ENDVARS`. The variables are determined by means of the grib code headers, e.g.:

Name	NLEV	EE	LEVTYP	LEVBOT	Description
`HSURF`	0	8	1	0	** Geometric height [m]
`T`	40	11	110	0	** Temperature [K]
`U10M`	0	33	105	10	** u wind [m/s] in 10 m

with

NLEV Number of vertical levels (0 stands for a two-dimensional field).

EE Element number in the Grib tables.

LEVTYP Leveltyp in the Grib tables.

LEVBOT Bottom level for Leveltyp 105.

A weakness of GrADS is, that it cannot distinguish between different Grib tables. Therefore, variables from different Grib tables, but with the same element number cannot be displayed properly. This holds for example for the wind speed `U` (in Grib Table 2, `ee=33`) and cloud ice `QI` (in Grib Table 201, `ee=33`).

Figure 4.1 shows an example for a data descriptor file.

4.3 The gribmap program

`gribmap` produces the connection between the data set specified in the data descriptor file and the variables contained in this data set and also specified by the data descriptor file. The usage of `gribmap` is:

```
gribmap -v -i test_m40.ctl
```

where:

`-v` Option for verbose output.

`-i` Option for specifying a data descriptor file for input.

If the `-v` option is specified, `gribmap` gives detailed information about every record processed from the Grib file. This version is helpful to detect possible errors in the data descriptor file.

4.4 A Sample GrADS Session

Before you start

The following is based on a tutorial taken from the GrADS web page. The Grib output from the reference data set (7 km version) is needed to go through this sample session. To avoid the problem with same element numbers in different Grib tables and to have a higher output frequency, rerun the `run_cosmo_7`-job with the following changes in the Namelist group `GRIBOUT`:

- eliminate `QI` from the output list in `yvarml`.
- For a higher output frequency set `hcomb=0.0,78.0,1.0`.

Then you need the following files:

```
dset /gtmp/for2sch/test_program/laf2002012300
index laf_test.idx
dtype grib 255
undef 9.999E+20
*
tdef 1 linear 00Z23jan2002 1hr
xdef 325 linear 1 1
ydef 325 linear 1 1
zdef 35 levels
8996 8739 8482 8225 7968 7711 7454 7197 6940 6683 6426 6169 5912 5655 5398 5141
4884 4627 4370 4113 3856 3599 3342 3085 2828 2571 2314 2057 1800 1543 1286 1029
772 515 258
*
VARS            32
HSURF           0   8,  1,0     ** Geometric height [m]
SOILTYP         0  57,  1,0     **  type of soil [idx]
ROOTDP          0  62,  1,0     **  depth of roots [m]
FR_LAND         0  81,  1,0     ** Land-sea mask (land=1;sea=0) [fraction]
PLCOV           0  87,  1,0     ** Vegetation [%]
PS              0   1,  1,0     ** Surface pressure [Pa]
PMSL            0   2,102,0     ** Pressure reduced to MSL [Pa]
QV_S            0  51,  1,0     ** Specific humidity [kg/kg]
TOT_PREC        0  61,  1,0     ** Total precipitation [kg/m^2]
RAIN_GSP        0 102,  1,0     ** grid scale rain [kg/m^2]
RAIN_CON        0 113,  1,0     ** convective rain [kg/m^2]
SNOW_GSP        0  79,  1,0     ** grid scale snow [kg/m^2]
SNOW_CON        0  78,  1,0     ** Convective snow [kg/m^2]
VMAX_10M        0 187,105,10    ** conv. cloud cover [%]
U10m            0  33,105,10    ** u wind [m/s]
V10m            0  34,105,10    ** v wind [m/s]
TMAX2m          0  15,105,2     ** Max. temp. [K]
TMIN2m          0  16,105,2     ** Min. temp. [K]
TD2m            0  17,105,2     ** Dew point temp. [K]
T2m             0  11,105,2     ** Temperature [K]}
U              35  33,110,0     ** u wind [m/s]
V              35  34,110,0     ** v wind [m/s]
T              35  11,110,0     ** Temperature [K]}
P              35   1,110,0     ** Pressure deviation
QC             35  31,110,0     ** Specific cloud water content [kg/kg]
QV             35  51,110,0     ** Specific humidity [kg/kg]
T_S             0  85,111,0     ** Soil temp. [K]
T_M             0  85,111,9     ** Soil temp. [K]
T_CL            0  85,111,41    ** Soil temp. [K]
W_G1            0  86,112,10    ** Soil moisture content [kg/m^2]
W_G2            0  86,112,2660  ** Soil moisture content [kg/m^2]
W_CL            0  86,112,25790 ** Soil moisture content [kg/m^2]
ENDVARS
```

Figure 4.1: Example for a GrADS data descriptor file

`all_lfff_refdata`	This grib files contains all output files from the COSMO-Model reference data set.
	`lfff00000000c, lfff00000000, ..., lfff00120000`
	If you made the reference run, you can create it by typing
	`cat lfff000000c lfff*00 > all_lfff_refdata`

`ref_model.ctl`	The GrADS data descriptor file for the reference data set.

The grib file `all_lfff_refdata` and the data descriptor file `ref_model.ctl` can be accessed from `/k/uhome/fegast3/COSMO_CLM_TRAINING_2013/data`.

Note that there is a tool `grib2ctl`, shortly described in section (4.5.2), which can automatically generate .ctl-files from a given grib-file.

You may want to look at the descriptor file before continuing. Especially the description of the subset of the 4-dimensional space needs to be explained in detail:

`tdef 13 linear 12Z20jan2013 1hr`	This data set contains COSMO-Model output data for 13 hours (12-24 UTC) from 20. January 2013 in hourly intervals.
`xdef 129 linear 1 1`	129 grid points in longitude direction.
`ydef 161 linear 1 1`	161 grid points in latitude direction.
`zdef 35 levels`	40 vertical levels.
`10281 10024 9767 9510...`	specification list of the vertical coordinates.

Please copy the data descriptor file to a local directory before proceeding.

GrADS Environment Variables

GrADS is working with scripts, that can be executed during a GrADS session. Also you have to access special data e.g. to display a map background. The directories, where data and scripts are stored, have to be specified by special GrADS environment variables. `GADDDIR` specifies the directory with special GrADS data and `GASCRP` defines the directory holding GrADS scripts. The settings for this course are:

```
export GADDIR=/usr/local/pkg/grads/data
export GASCRP=/k/uhome/fegast3/COSMO_CLM_TRAINING_2013/grads/grads_scripts
```

Starting GrADS

GrADS can be started interactively by using the command `grads`. The user is prompted for specifying a landscape or a portrait window. Normally, a landscape window is chosen, so just press `ENTER`. GrADS then opens a graphic output window on your console, in which the data will be displayed. This window can be moved or resized. You have to enter GrADS commands from the window where you started GrADS – this window will need to be made the *active* window and you must not entirely cover that window with the graphics output window.

In the text window (from where you started GrADS), you should now see the GrADS prompt `ga->`. You have to enter GrADS commands at this prompt and you will see the results displayed in the graphics output window.

First, the data descriptor file has to be opened. For that you have to enter:

```
ga-> open ref_model.ctl
```

You can take a look on the contents of this file by entering

```
ga-> query file
```

For displaying COSMO-Model output, a number of initial settings have to be done. These are just for specifying the plotting area and defining some COSMO-Model specific features. These initial settings have been placed into a file called `pin.gs` (.gs stands for *grads script*), which can be executed by typing

```
ga-> run pin
```

The file `pin.gs` can be accessed from the GrADS scripts-directory, which is set by an environment variable (see above). You can take a look to see how this script looks like.

Displaying Data

Data can be displayed by using the command `display` (or abbreviated: `d`):

```
ga-> d var_name
```

where *var_name* is the name of the variable as it was defined in the data descriptor file, e.g.:

```
ga-> d u
ga-> d hsurf
```

Note that GrADS displays an x-y-plot at the first time and at the lowest level in the data set. Per default, contour lines are plotted. This can be changed by entering

```
ga-> set gxout style
```

where *style* is one of the following:

`contour`	the default
`shaded`	coloured plots
`grfill`	the single grid cells can be seen in the coloured plot
`grid`	the values are written in the grid cell (not useful for bigger domains)
`vector`	for displaying special variables (see below)
`stream`	for displaying special variables (see below)

Display some variables in different styles, e.g.

ga-> `d u` (the default are contour lines)

ga-> `c` (for `clear`: clears the display)

```
ga-> set gxout shaded
ga-> d u
ga-> c
ga-> set gxout grfill
ga-> d u
```

For the styles `shaded` and `grfill`, you can draw a color-bar to see the range of the values of the displayed variable by entering

```
ga-> run cbar
```

Note that `cbar` also is a user defined grads script.

Using the styles `vector` or `stream`, you can visualize e.g. the wind:

```
ga-> set gxout vector
ga-> d u;v
ga-> d skip(u,5,5);skip(v,5,5)
```

(Note the difference!)

```
ga-> c
ga-> set gxout stream
ga-> d u;v
ga-> c
```

By specifying a 3rd field, the vectors can also be colorized, e.g.

```
ga-> set gxout vector
ga-> d u;v;t
ga-> c
```

You will not see very much with the style `grid`, unless you change the size of the displayed domain.

Altering the dimensions

You now will enter commands to alter the *dimension environment*. The `display` command (and, implicitly, the access, operation, and output of the data) will do things with respect to the current dimension environment. You control the dimension environment with the `set` command.

```
ga-> c
ga-> set x 35
ga-> set y 30
ga-> d t
```

In the above sequence, the x- and y-dimensions are set to a single value. Because the z- and t-dimensions are also only a single value, all dimensions are now *fixed* and the result is just a single value, in this case the value for grid point (35,30), at the lowest level and the 1st time in the data set.

If you now enter

```
ga-> c
ga-> set x 1 129
ga-> d t
```

Now x (the longitude) is a varying dimension and displaying a variable will give a line graph.

Now enter

```
ga-> c
ga-> set y 1 161
ga-> d t
```

and you will get a two-dimensional plot again. You can also choose a subdomain for plotting:

```
ga-> c
ga-> set x 31 43
ga-> set y 25 45
ga-> d t
```

And on such a subdomain you can take a look on the values of the grid cells:

```
ga-> c
ga-> set gxout grid
ga-> d t
```

If also a third dimension is varied, you will get an animation sequence, which can be done e.g. through time:

```
ga-> c
```

```
ga-> set gxout shaded
ga-> set x 1 129
ga-> set y 1 161
ga-> set t 1 13
ga-> d t
```

The command to change the vertical level is the same

```
ga-> set z num
```

where *num* is a number from 1 (surface) up to the number of vertical levels (here: 40: top of atmosphere).

By setting the y- (latitude) and z- (level) dimension to vary, you can plot vertical cross sections (Exercise!)

Operations on data

GrADS offers the possibility to perform operations or several built in funtions on the data. To see the temperature in Fahrenheit instead of Kelvin, you can do the following conversion:

```
ga-> d (t-273.15)*9/5+32
```

(How can you display the temperature in Celsius?). If you use the display style `shaded`, you should also plot the color bar to see the differences.

Any expression may be entered that involves the standard operators of +, -, * and /, and which involves operands which may be constants, variables, or functions. Here is an example involving functions to calculate the magnitude of the wind. The sequence

```
ga-> c
ga-> d sqrt(u*u+v*v)
```

can be replaced by the function

```
ga-> d mag(u,v)
```

Another built in function is the averaging function:

```
ga-> d ave(ps,t=1,t=7)
```

In this case the mean surface pressure over 7 values (from 00 UTC to 06 UTC!) is calculated.

You can also perform time differencing

```
ga-> c
ga-> d ps(t=2) - ps(t=1)
```

to see the change of a variable between two output times.

In this example you see a more common form to refer to a variable by explicitly setting the time in brackets. You can also do such a calculation by using an offset from the current time:

```
ga-> c
ga-> d ps(t+1) - ps
```

The complete specification of a variable name is:

ga-> *name.file (dim +| − | = value, ...)*

If there are two files open, perhaps one with model output, the other with analyses, you can also take the difference between two fields by entering

```
ga-> d ps.2 - ps.1
```

.

Printing data

Newer version of GrADS have a comfortable way of printing data to a gif-file:

ga-> `printim` *file_name*

where *file_name* is the name that is given to the file. You can take a look on the contents of *file_name* by using `display`.

Ending GrADS

The command to end the GrADS session is

```
ga-> quit
```

4.4.1 Drawing a Map Background

Changing the `ctl`*-file*

To draw a map background is an advanced feature when visualizing COSMO-Model output with GrADS, because of the rotated latitude-longitude grid the predefined GrADS maps cannot be used. But we have some special maps for rotated grids that are used for the reference run. The maps are provided in a low, medium and high resolution over Europe for the rotation with `pollat=32.5`:

`gmt_l_rot_eu` low resolution map

`gmt_m_rot_eu` medium resolution map

`gmt_h_rot_eu` high resolution map

and for the rotation with `pollat=40.0`:

`gmt_l_rot_40.0` low resolution map

`gmt_m_rot_40.0` medium resolution map

These maps can be accessed from the directory `/usr/local/pkg/grads/data`. For other rotations, other maps have to be used. To use these maps, the rotated coordinates of the latitudes and the longitudes have to be specified in the data descriptor file:

```
xdef 129 linear -4.0 0.0625
ydef 161 linear -5.0 0.0625
```

Note: the setting before counted the grid points, now the rotated coordinates are specified with the start value (-3.5 and -12.0, resp) and the increment (the resolution) 0.125 (degree).

Then `gribmap` has to be run again.

Control the map drawing

Start GrADS, open the data descriptor file and run the initialization script.

```
ga-> open ref_model.ctl
ga-> run pin
```

To control the map drawing, you have to set

`ga-> set mproj latlon` (sets the map projection to a lat-lon grid where the aspect ration is maintained)

`ga-> set mpdset < gmt_l_rot_40.0 | gmt_m_rot_40.0 >`

If you now display a variable, the political borders and the coastlines are drawn by default in addition to the variable.

```
ga-> d hsurf
```

To switch off the borders and / or the coastlines you can use the `set mpt` command. The syntax is

`set mpt` *type* `< off |` <*color*>,<*style*>,<*thickness*>>

where *type*, *color*, *style* and *thickness* can be chosen from the following table:

type	*: asterisk for application to all types 1: coastlines 2: political borders 3: main rivers 4: additional rivers 5: canals
color	values for GrADS default colors range from 1 to 16 (1: black; 2: red; 3: green; 4: blue, etc.)
style	1: solid 2: long dash 3: short dash 4: long dash, short dash 5: dotted 6: dot dash 7: dot dot dash
thickness	values range from 1 to 6

To also switch on the main rivers in a black dotted line, you can enter

```
ga-> set mpt 3 1 5
```

and then display a variable. To switch all maps off, you can type

```
ga-> set mpt * off
```

Note, that if you now want to set a dimension to a special grid point, you cannot use the numbers $1, \ldots, 161$ any more, but have to specify the values for the latitudes and longitudes, resp.

4.4.2 Use of GrADS in Batch mode

You have already used the scripts `pin.gs` or `cbar.gs` above. You can easily generate your own scripts by embracing the most commands by an apostrophe (exceptions are variable definitions, ...). A short example is shown here

```
# comments must start with a # in the FIRST COLUMN!

'open test_m40.ctl'

'run pin'
'set gxout grfill'

'd t'

'printim  nice_plot.gif'
```

Let us name this script 'myscript.gs'. You can of course call this script during your grads session; but you can call it by a shell command, too:

```
> grads -b -c 'myscript.gs'
```

To automatically quit grads you should finish the above script by a line `'quit'`. Using scripts in this way allows to use GrADS as a graphic production tool.

4.5 Helpful Tools: wgrib and grib2ctl

As already mentioned above, GrADS has problems when two variables should be visualized that have the same element number but are from different Grib tables. Another problem is the visualization of threedimensional variables, that have different vertical locations. In the COSMO-Model, most atmospheric variables are defined on main levels, while the vertical velocity `W` (and some related variables) are defined on half levels, i.e. the interfaces between the main levels. This is the reason, why 41 different twodimensional levels are defined for `W` in the ctl-file `test_m40.ctl`. But therefore it is not possible to draw cross sections for `W`.

There are tools whcih offer now convenient solutions for these problems. With `wgrib` it is possible to extract certain variables out of a Grib file into a new Grib file. And if only one type of atmospheric variable is present, it is no problem for GrADS. And with the tool `grib2ctl`, a ctl-file can be automatically created, so there is no need to keep several of these files.

4.5.1 wgrib

A list of contents of a Grib file:

To see a list of all variables in a Grib file <*name*> type:

`wgrib` <*name*>.

The listing gives you (among others) information about

`d=09122112:`	the date
`U:`	the name used in the COSMO-Model
`kpds5=33:`	the element number (here defined by `kpds5`)
`kpds6=110:`	the leveltyp (110 are variables on main levels, 109 variables on half levels)

Extracting variables out of a Grib file:

The output of the listing above can be piped into `wgrib` again to extract certain variables, which you have to specify by the element number (or `kpds5`):

`wgrib` <*name*> `| egrep -i <string> | wgrib -grib -i` <*name*> `-o` <*name*>`_suffix`,

where `<string>` is e.g.

":U:kpds5=33:kpds6=110:" to extract U

":V:kpds5=34:kpds6=110:" to extract V

":W:kpds5=40:kpds6=109:" to extract W

":T:kpds5=11:kpds6=110:" to extract T

":P:kpds5=1:kpds6=110:" to extract P

":PP:kpds5=139:kpds6=110:" to extract PP

":QI:kpds5=33:kpds6=110:" to extract QI

You can also add several entries with the | in between:
":U:kpds5=33:kpds6=110:|:V:kpds5=34:kpds6=110:"

4.5.2 grib2ctl

Since early versions of GrADS, a tool grib2ctl has been offered, that automatically creates ctl-files for a special grib file:

grib2ctl <*gribfile*>.

But early versions could not handle the rotated lat-lon grid of the COSMO-Model and the different grib tables, that are used in the COSMO-Model. Since some time, a version is available, that works properly on Grib data from the COSMO-Model. You can access this file in
/k/uhome/fegast3/COSMO_CLM_TRAINING_2013/grads

4.5.3 Exercise

EXERCISE:
Extract the vertical velocity W out of one grib file and try to plot a x-z cross section.

Chapter 5

Postprocessing and visualization of NetCDF files

5.1 Introduction

Many visualization programs such as GrADS, R or Matlab now include packages with which NetCDF data files can be handled. In this course we will use a very simple program, `ncview`, which does not have a large functionality but is a very easy-to-use program for a quick view of NetCDF output files and therefore very useful for a first impression.

For a quick overview of dimensions and variables, the tool `ncdump` can be used, which is part of the basic NetCDF package. This will shortly be described first. More sophisticated tools exist for e.g. cutting out subsets of data and producing averages or time series. Two of these tools which are currently frequently used in regional climate modelling are the `cdo` and `nco` utilities.

In the CLM community, a tool for combining temperature and precipitation fields to monthly values, cut out the PRUDENCE regions and compare to E-OBS observation data exists. This tool is called `ETOOL` and is available on the CLM community web page.

Note that for this years exercises, the netcdf package of version 4 or higher is needed and some netcdf files (e.g. external parameter file or tar-archived CCLM output) are internally compressed using `nccopy`. Also, the netcdf tools `ncview`, `CDO` and `NCO` have to be compiled and linked with the netcdf4 package. This may cause problems if you want to handle the data files on other computer systems.

5.2 ncdump

Ncdump comes with the NetCDF library as provided by Unidata and generates a text representation of a NetCDF-file on standard output. The text representation is in a form called CDL (network Common Data form Language). Ncdump may be used as a slimple 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 look at the structure of a NetCDF file use

`ncdump -c` *data-file.nc*

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

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

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

`ncdump -v` <Variable> *data-file.nc*

To send data to a text file use

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

to produce an annotated CDL version of the structure and the data in the NetCDF-file *data-file.nc*. You can also save data for specified variables for example in *.txt-files just using:

`ncdump -v` <Variable> *data-file.nc* > *data-file.txt*

For further information on working with `ncdump` see

`http://www.unidata.ucar.edu/software/netcdf/docs/ncdump-man-1.html`

5.3 CDO and NCO

CDO (Climate Data Operators) is a collection of command line operators to manipulate and analyse NetCDF data, which have been developed at MPI for Meteorology in Hamburg. Source code and documentation are available from:

`https://code.zmaw.de/projects/cdo`

The tool includes more than 400 operators to: print information about datasets, copy, split and merge datasets, select parts of a dataset, compare datasets, modify datasets, arithmetically process datasets, 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.

NCO is short for "NetCDF Operator". It has been developed in the US and is now an Open Source Project. As CDO, the NCOs are a collection of command line operators. Source code and documentation are available from:

`http://nco.sourceforge.net`

The main operators are: arithmetic processor, attribute editor, binary operator, ensemble averager, ensemble concatenator, file interpolator, kitchen sink, permute dimensions quickly, pack data quietly, record averager, record concatenator, renamer, weighted averager.

Depending on the specific task of your postprocessing steps, it is useful to have a closer look at whether a CDO or NCO operator would be more suitable.

5.4 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 your NetCDF files. It is possible to view simple movies of data, view along different dimensions, to have a look at actual data values at specific coordinates, change colormaps, invert data, etc.

To install ncview on your local platform, see the ncview-Homepage:
`http://meteora.ucsd.edu/~pierce/ncview_home_page.html`

You can run the program by typing:

`ncview` *data-file.nc*

which will open a new window with the display options.

If *data-file.nc* contains wildcards such as '*' or '?' then all files matching the description are scanned, if all of the files contain the same variables on the same grid. Choose the variable you want to view. Variables which are functions of longitude and latitude will be displayed in 2-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 view of the image can be changed by varying the colours of the displayed range of the data set values or by adding/removing coastlines. Each 1- or 2-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 *.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 `x`-icon on the top right of the window will close all ncview windows and terminate the entire program.

5.5 Further tools for visualization and manipulation of NetCDF data

There are many more programs for viewing and manipulating NetCDF data. A list of freely available software as well as commercial or licensed packages can be found at
`http://www.unidata.ucar.edu/software/netcdf/software.html`.

Furthermore, there are now interfaces and routines to handle NetCDF data for almost every programming and script language available.

5.6 ETOOL

The ETOOL is a shell script which is designed to perform some basic analysis of COSMO-CLM output data and to compare it to the E-OBS gridded observational data. Monthly statistics for temperature and precipitation are calculated, and the European 'PRUDENCE' regions are extracted. It is also possible to define custom regions. The program works for all horizontal resolutions, as the validation is calculated only for area means of the selected sub-regions. Note that only land points (FR_LAND >= 0.5) are included in the analysis, since the model output is remapped onto the E-OBS grid. To run the program you need the E-OBS data set and the CDO package, as the calculations are performed using the CDO commands.

The script has been provided by Peter Berg from KIT and is available from the CLM community web page[1].

5.7 Exercises

At DKRZ, any postprocessing should be performed on the `lizard` computer, not the `blizzard` (cf. appendix B.1).

[1] http://www.clm-community.eu/index.php?menuid=85&reporeid=79

For this exercise, we use the output of the 2-month sp001 chain experiment. In the directory `/work/bb0721/PP_VISU/2013/` you will find some observation data (E-OBS data set) and some scripts to start with, including the ETOOL script which is available from the CLM community webpage.

EXERCISES:

- Have a look at your data (model output and observations) using `ncview`:
  `ncview <file>.nc`
- Practice the use of the CDO tools. For this, use the template `ex_CDO.sh` and fill in the missing commands. You can use the documentation `cdo.pdf` if you need detailed information about a command.
- Practice the use of the NCO tools. For this, use the template `ex_NCO.sh` and fill in the missing commands. You can use the documentation `nco.pdf` if you need detailed information about a command.
- Now use the ETOOL to manipulate the 2 m temperature and precipitation data of your model run and compare them to the E-OBS data. Use the template `ex_ETOOL.sh` to call the ETOOL script and fill it according to the comments.
  The output of a one-year simulation (an extension of the starter package sp001 simulation) is available in the sub-directory `mod_data`. If you want you can use this to produce a whole annual cycle with the ETOOL.

Chapter 6

Running Idealized Test Cases

6.1 General remarks

A so-called idealized case is a model run without coupling/nesting to any sort of coarser "driving" model. There is no need for preprocessing with INT2LM. Instead, all necessary initial and boundary data are computed within the COSMO-Model itself. To accomplish this, some simplifications can be introduced, e.g., horizontally homogeneous profiles of p, T and $\vec{v}$, time-invariant lateral boundary data, running without soil model and radiation, idealized terrain-shapes etc, but more complicated things are imaginable as well.

Many analytic results from theoretical meteorology are based on the solution of the atmospheric flow equations under simplified conditions (e.g., wave flow over hills), and it is often possible to reproduce these conditions numerically in the idealized framework. Thefore, the ability to run idealized model setups serves, on the one hand, as simple a possibility to test the correctness of particular aspects of the COSMO model (e.g., the dynamical core only, some physical parameterization) by comparison to such analytical reference solutions from theory, and, on the other hand, it enables the researcher to inspect atmospheric processes in their pure form.

To run the model in idealized mode, the following "ingredients" have to be defined/computed within the model, which would normally, at least partly, come from other sources (INT2LM, EXTPAR):

- height distribution of vertical model levels,
- orography,
- external parameters, depending on the configuration (e.g., run w/o soil model, radiation),
- initial conditions for p, T and $\vec{v}$ (can be simplified, so that any prognostic variable $\phi = \phi(z)$ only),
- boundary conditions (e.g., same as initial condition and held constant in time, or periodic, or "open"),
- For studies of convection: artificial convection triggers, random noise.

The code for these ingredients is implemented in the source code file `src_artifdata.f90`. This file contains a quite large number of subroutines, offering different possibilities for different things, but it is certainly not exhaustive and may be extended by the user on its own. **However, a lot of idealized cases can already be configured by namelist parameters of the Namelist**

group ARTIFCTL in the file INPUT_IDEAL, and for most beginning users there will be no need to change the source code.

This namelist-driven one-soure-code approach not only facilitates parameter changes, but more importantly allows to freely combine the various ingredients to create "new" cases. This flexibility offers, e.g., the possibility for model developers to quickly set up an idealized case tailored to test their current development. And it opens up new ways for meteorological researchers to conduct idealized studies on atmospheric processes.

However, as has been said, the user is free to add own specialized ingredients to `src_artifdata.f90`, if he is not satisfied with the possibilities already given. Most of such ingredients are likely to fit into the present code structure of this file at some more or less well-defined entry points, and the top-level routines of `src_artifdata.f90` are:

- `SUBROUTINE input_artifctl`: Input of the various namelist parameters. You can add own ones here by orienting yourself on the already present parameters.
- `SUBROUTINE gen_ini_data`: Is called once at the beginning of a run and generates vertical coordinates, orography, initial atmospheric profiles ($\vec{v}$, T, moisture, p) and, if needed, soil- and other external parameters. Here most of the namelist parameters take effect, and a number of subroutines for certain tasks are called. This is the routine where own code for the state should be put in.
- `SUBROUTINE gen_bound_data`: Is called in every timestep and sets the boundary fields.

These subroutines are extensively commented, so do not hesitate to dive into the code! You will see that there are a lot of (well-commented) child-subroutines, which are called from the above top-level ones. However, a detailed description is beyond the scope of this introduction.

Necessary material for the exercises in this chapter can be found in:

`/k/uhome/fegast3/COSMO_CLM_TRAINING_2013/ideal` :

- The runscripts `run_ideal`, `run_ideal_hill`, `run_ideal_hill_2` , `run_ideal_wk82`
- In the subdirectory `Doc`:
  - GrADS reference card, wgrib readme,
  - `artif_docu.pdf`, Documentation of the namelist `ARTIFCTL`,
  - articles of Weisman, Klemp (1982) for the warm bubble test, Schär et al. (2002) for flow over a mountain.

To use `src_artifdata.f90` you have to change the Namelist parameter `lartif_data=.TRUE.` in the Namelist group `RUNCTL`.

6.1.1 Visualization

The test cases need vertical cross sections, too. GrADS is not well suited for this purpose, mainly because it cannot use the height information of the terrain following coordinate. As a workaround you can use the following methods (see also Section 4.5 for more information):

1. output on z-levels:
   define several z-levels (up to 500) in the GRIBOUT-Namelist section

2. output on model levels:
   use `wgrib` to extract only `T, P, U, V` fields (on main levels) or `W` field (on half levels) out of a file `<name>` (e.g. `<name> = lfff01000000`):
   `> wgrib <name> | egrep -i <string> | wgrib -grib -i <name> -o <name>_uv`
   and for `<string>` e.g.
   `":U:kpds5=33:kpds6=110:|:V:kpds5=34:kpds6=110:"`
   if you want to extract U and V.

To generate a `.ctl` file use e.g.:
`> grib2ctl lfff01000000_uv > l24.ctl`
and then generate an appropriate index-file by
`> gribmap -0 -i l24.ctl`
(try also `gribmap -help`)

6.1.2 gnuplot

The COSMO model does not only generate Grib or NetCDF output but also several ascii files. In particular `YUPRMASS` and `YUPRHUMI` contain time series of some variables. It is sometimes useful to have a graphical impression of those time series. There exist a lot of graphic tools to do that.

`gnuplot` is a very simple plotting tool mainly to produce 2D line plots from ascii data files. It is public domain and available for nearly all possible platforms.

Call it by typing
`> gnuplot`

Now you can e.g. type
`plot "YUPRMASS" u 1:4`
This takes the 1st column of the file `YUPRMASS` as the x values and the 4th column as the y-values.

Type `? plot` to get help and some examples about the plot-command.

Type `? set` to get help about a lot of possible settings for labels, ...

6.2 Exercises

`run_ideal` is a job-file containing a documented namelist `ARTIFCTL`, which contains all possible namelist parameters for idealized cases in a cook-book-like fashion.

However, to get you started, there are job-files for certain cases with reduced namelists, which contain only those parameters relevant for the respective case (e.g. `run_ideal_hill`).

If you later feel familiarized with the concept, you can start from the full `run_ideal` and explore the various possibilities to define the idealized elements (orography, initial conditions, boundary conditions (lower, upper, lateral), convection triggers, ...). After that, a more advanced topic is to introduce own modifications into the source file `src_artifdata.f90` according to your special needs.

Mountain flow 'gaussian hill'

- If not already done, compile the model.
- Adapt the runscript `run_ideal_hill`:
  - Choose computing platform.
  - Name of the COSMO-model directory, under which the `src/`-dir is found.
  - Name of the `executable`.
  - Name of the output directory.
  - Mountain height `hill_height`.
  - Mountain width `hill_width_x`.
  - Inflow velocity `u_infty` (e.g. also $v = 0$: atmosphere at rest).

  and run tests by typing: `> run_ideal_hill`
- Have a look to `YUSPECIF`. Are the NAMELIST-Parameter-Settings really what you wanted to simulate?
- Have a look to the temporal behaviour of some means in YUPRMASS and YUPRHUMI (you can visualize the temporal behaviour e.g. with `gnuplot`).
- Have a look into one of the grib files by
  `> wgrib lfff01000000 | less` ;
  extract the field `U` or `W` as described above.
- Plot vertical cross sections of `U, W, T, P` and θ with GrADS. (`hill.gs` is a very simple script for a plot of `U`).

Warm bubble, non-linear convection with phase change

Literature: Weisman, Klemp (1982) MWR

- use `run_ideal_wk82` and run test with different inflow velocity profiles
  (modify NAMELIST-Parameter `u_infty`)
- Plot vertical cross sections of `W, T, P,` θ and the humidity variables `QV, QC, QI`.
- Plot horizontal cross section in $z = 2000$m of `W` with vector plots of `(U,V)`.

Advanced cases

Flow over own 'hill-valley-hill'-combination

- Use job-file `run_ideal_hill_2`, which is an extension of `run_ideal_hill` to allow your own creative modifications. Play around with the number of mountains/valleys, the initial T- and u-profiles, etc:
  - E.g., switch on 3 mountains/hills by setting `lhill=.true.,.true.,.true.,`.

- Expand the other hill-related namelist variables to corresponding lists of 3 items, e.g. mountain height `hill_height=1000.0,-300.0,500.0,` (negative values denote a valley) mountain position in gridpoints `hill_i=80.5,113.5,145.0,`
- To change the initial wind profile $u(z)$, set `itype_anaprof_uv` accordingly (see comments in job-file). Possibilities are constant u `itype_anaprof_uv=3, u_infty=...` (e.g. also $\vec{v} = 0$: atmosphere at rest), or different layers with linear windspeed gradient `itype_anaprof_uv=2` and related other parameters, or Weisman-Klemp-type tanh-shaped profile `itype_anaprof_uv=1, u_infty, href_wk` .
- To change the initial profiles $T(z)$ and $Q_v(z)$, set `itype_anaprof_tqv` and the related namelist parameters according to your wishes (see comments in job-file), e.g., choose a no-slip or free-slip lower boundary condition by setting `lnosurffluxes_m=.false.` or `.true.`
- The resulting orography is added to the base height `href_oro`. By choosing `href_oro` appropriately, make sure that the resulting orography is not lower than the base height of the thermodynamic profiles!
- Think what you are doing, it is your responsibility that the chosen combination makes any sense!

Convective cell moving over a mountain

- Copy `run_ideal_hill` to e.g. `run_ideal_hill_bubble`, modify it accordingly:
- Combine the flow over a 2D-hill (a flow-perpendicular dam over the whole model domain) with the triggering of convection upstream of the mountain. Explore `hcond_on` (time in h of switching on the condensation and microphysics) to allow the mountain flow to relax against a steady state before triggering convection by a bubble of type, e.g., "`cos-hrd`" (heating rate disturbance in a Weisman-Klemp-shaped bubble during a specified heating period).
- Choose fixed relaxation boundary conditions in x-direction and periodic boundary conditions in y-direction by `lperi_x=.false., lperi_y=.true.`
- Plot vertical cross sections of `W, T, P,` θ and the humidity variables `QV, QC, QI`.
- Plot horizontal cross section in $z = 2000$m of `W` with vector plots of `(U,V)`

Mountain flow 'Rippelberg'

Literature: Schär et al. (2002), Mon. Wea. Rev., section 5b

- Change the program code in `src_artifdata.f90`:
  - Program an orography as given in Schär et al. (2002);
    hint: look for subroutine `add_orography_ana`, implement your own case `'schaer'` by orienting yourself on the case `'gauss-2d-simple'` (hidden in the subroutine) and make best use of existing namelist parameters like `hill_height`, `hill_width_x`.
  - Add the possibility to use 65 equidistant levels ($z = 300$m); choose `ivctype=2` , `zspacing_type='linear'`, and choose combination of `zz_top` and `ke` accordingly.
  - Choose an isothermal atmosphere.
- Compile the model

- Copy `run_ideal_hill` to e.g. `run_ideal_hill_schaer`, modify it accordingly and run the test.

- Plot vertical cross sections of `U, W, T, P` and θ as above.

Chapter 7

Troubleshooting for the COSMO-Model

When you work with the software package for the COSMO-Model System, you can have a lot of trouble. Some of them are due to imperfect (or even missing) documentation (but sometimes also "not reading carefully"), others are caused by program bugs. We apologize right now, for any inconvenience this may cause. But there are also troubles resulting from compiler bugs or hardware problems.

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

7.1 Compiling and Linking

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

1. First you have to find some proper compiler and linker options, if there are no expample `Fopts`-files given in the delivered version. This is a job, that we cannot support. You have to take a look into the manual of your compiler. But we can give one hint: Try a moderate optimization first! Many compiler problems come from very high optimizations.
2. If you found proper options and can start the compilation, an error message like `Unable to access module symbol file for module netcdf` may occur. When you have a library for NetCDF available and add that to the LIB-definition in the Makefile, you also have to give a compiler option: `-I xxx/include`, indicating the path, where the compiled module files for NetCDF are lying.

3. In the linking process, you can get error messages like:

   ```
   Unsatisfied external references: .__netcdf_NMOD_nf90_def_dim
   ```

   or

   ```
   Undefined symbol .rttov7_rttvi
   ```

   . Then you forgot to link a certain library, to activate a certain `dummy_xxx` module in the file `ObjFiles` or you unintentionally specified a preprocessor directive.

4. You can get these messages, even if you put all libraries to the `LIB` Variable, e.g.

   ```
   #LIB =      /rhome/routarz/libdwd/lib/libgrib1.a  \
    LIB =      /uhome/trngxyz/lib/libgrib1.a  \
               /uhome/trngxyz/lib/libnetcdf.a
   ```

   Note: Some systems treat all these lines as a comment, because the first line has the comment-sign and a continuation-sign at the end.

7.2 Troubleshooting in NWP-Mode

For the following examples you need the reference data set. Special run-scripts with Namelist Input are provided in `/k/uhome/fegast3/COSMO_CLM_TRAINING_2013/troubleshooting`.

7.2.1 Running INT2LM

1. The most common mistakes, when running INT2LM, are the treatment of bitmaps (if necessary) and the specification of the COSMO-Model domain. Unfortunately, the error messages are sometimes missleading.

   > EXERCISE:
   > Take the runscripts 1-3, modify the directories (if necessary, e.g. `IN_DIR`, `LM_DIR` path and name of the working directory and the binary `mpirun ...`) Try to run the programs using these scripts. Take a look to the error messages in the batch output and correct the run-scripts to get proper runs

2. Another problem is demonstrated in the next example. It is tried to nest a domain with a finer resolution into the output of the reference run. For that exercise you need the output of the 7 km COSMO-run.

   > EXERCISE:
   > Run the INT2LM using the script `runeu2de` from the directory above. Modify only the output directory `LM_DIR` and the path and name of the binary and the actual working directory. What is the error? And what could be the reason?
   > Try to use the output file `lfff00000000c` from the reference data set instead of `cosmo_d5_07000_965x773.g1_2009121700`. Run the corrected script for the 12 hour forecast (is needed in the next exercise).

7.2.2 Running the COSMO-Model

1. Some of the problems of INT2LM you will only notice, when running the COSMO-Model. This can happen, if the COSMO-Model wants to read certain fields, that are not provided by the INT2LM.

   EXERCISE:
   Run the COSMO-Model with the runscript `run_cosmo_test_1` from the directory above. Modify the input directory to use the results from the INT2LM run from the last exercise. Modify the output directory `OUTPUT` and the path and the name of the binary. Take a look to the error message in batch output. What is the reason and what are possible solutions?

2. Sometime problems may occur, even when you think you set everything perfect.

   EXERCISE:
   Run the COSMO-Model with the runscript `run_cosmo_test_2` from the directory above. Modify only the output directory `OUTPUT` and the path and the name of the binary. Take a look to the error message in the batch output. What is the reason?

There are surely much more reasons for problems and errors, which we cannot discuss all during these lessons. It really gets troublesome, when a program aborts and writes core-directories. Then you will need some computer tools (like a debugger) to investigate the problem. In some cases it will help to increase the debug output by setting `idbg_level` to a higher value and activating debug output for special components (with `ldebug_`*xxx*). If you cannot figure out, what the reason for your problem is, we can try to give some support.

7.3 Troubleshooting in CLM-Mode

In the following, some common errors when running INT2LM or COSMO-CLM in climate mode are listed together with their solution:

- The COSMO-CLM simulation does not start; instead you get an error message saying `*** Error while opening file YUSPECIF ***`.
  → This occurs when you have already `YU*` files in the current directory from a previous simulation. You need to delete these files before starting a new run.

- Your INT2LM and CCLM run scripts are arranged to use cloud ice as initial and boundary conditions, but this field is not available in the GCM data. Both parts will crash.
  → In INT2LM, set `lprog_qi = .FALSE.`, in CCLM, set `lana_qi = llb_qi = .FALSE.`.

- You use an external parameter file including 12-monthly vegetation data. INT2LM does not start an complains that not all necessary input fields can be found
  → You need to set `itype_ndvi = 2` in INT2LM.

- You use an external parameter file including maximum and minimum vegetation data. INT2LM does not start an complains that not all necessary input fields can be found
  → You need to set `itype_ndvi = 0` in INT2LM.

- You change your simulation settings from Leapfrog to Runge-Kutta time integration scheme by setting `l2tls = .TRUE.`, but your run crashes and reports a problem with `nboundlines`.
  → `nboundlines` gives the number of lines around each sub-domain which are communicated among the different processors. For Leapfrog a minimum of 2 boundary lines is necessary; for Runge-Kutta 3 lines are needed.

- You have prepared an external parameter file with PEP using exactly the domain information which you intend to use for your CCLM run (namelist `&LMGRID` in INT2LM), but INT2LM does not run.
  → For the setup of the data at the boundary (due to interpolation), 2 more lines of data are required outside the boundary limits. Prepare your PEP file with (at least) 2 more grid points in each direction.

- Sometimes, the error message of a crashed simulation does not point directly at the problem. For example, you use the same script and same GCM data with a new INT2LM code version and get an error in reading the namelist of the input file.
  → It happens sometimes that from one model version to the next a namelist parameter is skipped or its name has been changed. When you use then an old run script where this parameter is included, the model gets problems reading the namelists correctly. Solution: check carefully the changes in the code (files `misc.global` for general COSMO changes and `README_CHANGES` for additional CLM changes) before using a new model version!!

EXERCISE:
You will get a list of 'live' examples on how to apply the COSMO-CLM for a few different situations, where you can try running the model yourself and hopefully avoid or manage the mentionned problems. At the end of the lesson, we will discuss the solutions.

Chapter 8

Running COSMO-ART

Heike Vogel (KIT), Daniel Rieger (KIT), Andrew Ferrone (KIT)

In this lesson you will learn how to run the package for Aerosols and Reactive Trace Gases, COSMO-ART.

8.1 General remarks

COSMO-ART (ART = Aerosols and Reactive Trace Gases, Vogel et al., 2009) is an extension of the COSMO model that was developed at the Institute for Meteorology and Climate Research at the Karlsruhe Institute of Technology (KIT). It allows the online calculation of reactive trace substances and their interaction with the atmosphere. Since model version 4.11 the ART interfaces are part of the COSMO model. Users who want to apply COSMO-ART should contact Bernhard Vogel (bernhard.vogel@kit.edu) in order to get the necessary extensions of the COSMO model.

8.2 Source Code and Installation

In the following we will describe the necessary subroutines, the namelist variables, the input data, and show some exercises.
Now listed are the ART-specific files additional to the COSMO-model:

Gases and Aerosols

Declaration of fields, constants etc.:

- `art_gas_const.f90`
- `art_gas_const_radm2.f90`
- `art_aerosol_const.f90`
- `data_cosmo_art.f90`
- `art_utilities.f90`
- `art_io_utilities.f90`
- `art_species_data.f90`
- `art_src_setup_vartab.f90`
- `art_restart.f90`

Emission modules:

```
art_dust.f90
art_seasalt.f90
art_emiss_prescribed.f90
art_emiss_param.f90
art_bvoc.f90
art_dms.f90
```

Boundary data modules:

```
art_icbc.f90
art_extpar.f90
```

Organizing gas phase chemistry and aerosol dynamics:

```
organize_cosmo_art.f90
```

Gas phase chemistry, aerosol dynamics:

```
art_radm2.f90
art_chemkpp.f90
art_mademod.f90
art_washout.f90
art_papa.f90
art_deposition.f90
art_isorropia.f
```

Optical properties of different aerosols:

```
art_rad_aero.f90
art_rad_dust.f90
art_rad_seas.f90
```

Aerosol cloud interaction:

```
art_aci.f90
art_nenes_activation.f
art_nenes_icenuc.f90
parametr.inc
```

Pollen:

```
organize_pollen.f90
pol_emissions.f90
data_pollen.f90
```

Installation of COSMO-ART (NWP group)

The source code of COSMO-ART is provided in a tar-file together with a special `Makefile`, and the files `Fopts`, `ObjDependencies_ART` and `ObjDependencies_COSMO_with_ART`. For the COSMO-CLM Training 2013, the version 2.0 of `ART` is provided in

`/k/uhome/fegast3/COSMO_CLM_TRAINING_2013/source/COSMO_ART_2.0.tar.gz`

for people using the NEC at DWD, based on COSMO 4.26.2.

To install COSMO-ART, copy the compressed tar-file to the directory, where you installed the COSMO-Model: `cosmo_130207_4.26.2`. Untar the `COSMO_ART_2.0.tar.gz` to create the subdirectory `COSMO_ART_2.0`, which contains additional subdirectories and files.[1]

`Makefile_for_ART`	This Makefile must be used for the compilation of COSMO-ART and the COSMO-Model.
`ObjDependencies_ART`	This file contains the list of dependencies for the COSMO-ART source code.
`ObjDependencies_COSMO_with_ART`	This file contains the list of dependencies necessary for the COSMO-Model, extended by the additional dependencies for the COSMO-ART interfaces. This file must be used when compiling the COSMO-Model together with COSMO-ART.
`ObjFiles_ART`	This is the list of objects files for COSMO-ART.
`src_art`	Subdirectory for the source code of COSMO-ART.

Copy the file `Makefile_for_ART` to your directory `cosmo_130207_4.26.2` to start the compilation. Clean up the obj-directory, because also the COSMO-Code has to be re-compiled now.

- Necessary changes in `Fopts`:
  To activate compilation of COSMO-ART, insert the option `-DCOSMOART`, and to activate compilation of Pollen, insert the option `-DPOLLEN` to the compiler call. (Check, whether the option `-sx8r` is set in the compiler and linker calls).
- COSMO-ART also contains Fortran 77 subroutines. To handle these routines properly, there is a special compiler call `F77` in `Makefile_for_ART`, which has also to be extended by the option `-sx8r`.
- Compile the COSMO-Code and the COSMO-ART code with

  ```
  gmake -f Makefile_for_ART -j 8 artexe.
  ```

  This creates the binary `lmparbin_art`

Installation of COSMO-ART (CLM group)

The source code of COSMO-ART is included in the starter package. You have already extracted the starter package and compiled the standard version of COSMO in a previous exercise. The code version for COSMO-ART is found in cclm-sp/src/cclm-art. Compile the code by calling

```
make
```

in this directory. This creates the binary `cclm-art.exe`.

The `ART` part of COSMO-ART is activated through preprocessor directives in the Fopts file (`-DCOSMOART`, `-DPOLLEN`). These are already set in the starter package.

[1] For COSMO-ART I/O NETCDF-library is necessary

NAMELIST-variables for COSMO-ART

The execution of COSMO-ART can be controlled by the following NAMELIST groups:

`RUNCTL`	parameters for the model run
`ART`	parameters for gases and aerosols
`POL`	parameters for pollen

With the following namelist parameters you can control different settings of COSMO-ART. LOGICAL switches begin with an `'l'` and you can turn components on (.TRUE.) and off (.FALSE.). Parameters beginning with another letter are either INTEGER, REAL or CHARACTER:

`&RUNCTL`

`l_cosmo_art`	gases and aerosols
`l_pollen`	pollen

`&COSMO_ART`

`lgas`	gas phase
`lkpp`	gas phase generted with KPP (Kinetic PreProcessor, test phase)
`lgasini`	initial values for gases available
`lgasbd`	boundary values for gases available
`lgas_emiss_in`	anthropogenic emissions for gases available
`lbvoc`	biogenic emissions for gases available
`lpoint`	emissions of point sources available
`(ydirin_art)`	directory of emission data
`laero`	aerosols (for secondary aerosols lgas must be .TRUE.)
`laeroini`	initial values for aerosols available
`laerobd`	boundary values for aerosols available
`ldust`	emissions of mineral dust (laero must be .TRUE.)
`lseas`	emissions of sea salt (laero must be .TRUE.)
`lwash`	wet deposition of aerosols
`laero_emiss_in`	anthropogenic emissions for aerosols available
`lrad_aero`	submicron aerosol radiation interaction
`lrad_dust`	mineral dust radiation interaction
`lrad_seas`	sea salt radiation interaction
`laci_warm`	aerosol-cloud interaction warm phase
`laci_cold`	aerosol-cloud interaction cold phase
`aerostart`	start hour for aerosols
`artstart`	start hour for ART
`hinc_emissions`	time increment for emissions (h)
`hinc_artbounds`	time increment for boundary data (h)
`iart_halo_block`	length of blocks for ART HALO exchange

`&POLLEN`

`isp_pollen`	number of pollen
`lpollenini`	initial values for pollen available
`lpollenbd`	boundary values for pollen available

Per default all switches are set to `.FALSE.`

You can run COSMO-ART with different settings for particles and gases. The following matrix shows you which switches has to be set for different settings.

	`l_cosmo_art`	`laero`	`lgas`	`lseas`	`ldust`	`lpollen`
Prim. Aerosol	x	x	-	-	-	-
Prim. & Sec. Aerosol	x	x	x	-	-	-
Gases	x	-	x	-	-	-
Only Sea Salt	x	x	-	x	-	-
Only Dust	x	x	-	-	x	-
Pollen	-	-	-	-	-	x

To run COSMO-ART the following input data are necessary:

Gas phase:

Anthropogenic emissions for different species (area sources, point sources etc.) (if available)
Land use data for biogenic emissions and deposition

Aerosol particles:

Anthropogenic emissions (if available)

Mineral dust:

Soil specific land use data

Pollen:

Land use data of individual plants
Threshold values for start of pollen season (if available)

These files are created from raw input data by the INT2LM-ART extension of the normal COSMO preprocessor INT2LM (see section 8.5 for more details), like it is done for meteorological input. Beside the input data you need the file `photo.dat` for the photolysis. The input files for the following exercises are provided ready to use.

8.3 Exercises at DWD

To run the test cases, take the binary `lmparbin_art` that you created in your cosmo-directory. You will find the files, necessary to run the test cases for COSMO-ART in the directory:
`/e/uhome/fegast3/COSMO_CLM_TRAINING_2013/cosmo_art`

8.3.1 Dispersion of idealized area emissions

For the first exercise we use the setup for idealized simulations and simulate the dispersion of several areas of gaseous area emissions.

- Copy the meteorological input data, the file for input data for the photolysis `photo.dat`, the input data for deposition and for biogenic emissions `laf2008100100x` and the runfile. You will find the data in the COSMO-ART subdirectory `gasaero`.
- Change the directory name in the runfile and the **namelist** parameter for a simulation **only for gases**.
- Plot horizontal cross sections of the different gases. You can use `gas.ctl` in the COSMO-ART directory.

Additionally you can modify the location of the sources in the code:

- Edit the file `art_emiss_prescribed.f90` ,
- Search for the Subroutine `artif_area_emissions` and modify the location of the source.
- Compile and run the job again.

8.3.2 Emission and dispersion of sea salt

Sea salt is an important source of natural aerosols. In the model emissions of sea salt are based on a wind speed dependend parameterization as shown in this example.

- Copy the input data from the COSMO-ART subdirectory `nataero`
- Change **namelist** parameters for `ART` for **only sea salt** simulation.
- Plot horizontal cross sections of the mass and number concentration of the sea salt particles. You can use the file `aero.ctl` in the COSMO-ART directory.

8.3.3 Dispersion of mineral dust particles

During May 2008 (period simulated) mineral dust was transported from the Saharan desert over Europe. Its effects on the radiative fluxes introduced a discernible bias on ground temperature in the NWP predictions for this period.

- Use the same input data as for the sea salt simulation
- Set the **namelist** parameters for `ART` for a simulation for **only dust** particles.
- Plot horizontal cross sections of the mass and number concentration. You can use the file `aero.ctl.`

8.3.4 Dispersion of gases and ozone formation or depletion

This exercise uses the same setup as the first exercise (8.3.1) but the scenario is based on a dataset of actual anthropogenic emissions.

- Copy the emission dataset `lb*e`, the file for input data for the photolysis `photo.dat`, the input data for deposition and for biogenic emissions `laf2008100100x` and the runfile `run_cosmo_art`. You will find the files in the COSMO-ART subdirectory `gasaero`
- Change the directory names in the runfile and set the **namelist** parameters for `ART` for a simulation **only for gases**.
- Plot horizontal and vertical cross sections of the different gases. You can use `gas.ctl` in the COSMO-ART directory.

8.3.5 Dispersion of aerosols and formation of secondary aerosol particles

The final exercise shows the full potential of COSMO-ART, as secondary aerosols are calculated prognostically based on the concentrations of gaseous components.

- Use the same input data as for the simulatione for gases and ozone formation
- Set the **namelist** parameters for `ART` for a simulation for **gases and aerosols**.
- Plot horizontal and vertical cross sections of the different aerosol particles. You can use `gas.ctl` and `aero.ctl`.

8.4 Exercises in Hamburg

You have sucessfully compiled your cclm-art.exe in the src directory. All exercises are provided in the step_by_step directory. All necessary input files are already prepared and can be found in the data subdirectory. For each exercise a runscript is provided which you will have to adapt for the simulation. You will find the output of the simulation in a corresponding data subdirectory created automatically once you submit the script.

8.4.1 Dispersion of idealized area emissions

For the first exercise we use the setup for idealized emissions and and simulate the dispersion of several areas of gaseous and primary aerosol emissions.

- Use run_cclm-art_plume. Change the SPDIR directory name in the runfile and the **namelist** parameter for a simulation **only for gases**.
- Plot the horizontal and vertical cross sections for the different gases.

Additionally you can modify the location of the sources in the code (src/cclm-art):

- Edit the file `art_emiss_prescribed.f90` ,
- Search for the Subroutine `artif_area_emissions` and modify the location of the source.
- Recompile and run the job again.

8.4.2 Emission and dispersion of sea salt

Sea salt is an important source of natural aerosols. In the model emissions of sea salt are based on a wind speed dependend parameterization as shown in this example.

- Use run_cclm-art_seasalt. Change the SPDIR directory name in the runfile and the **namelist** parameters for `ART` for a **only sea salt** simulation.
- Plot horizontal cross sections of the mass and number concentration of the sea salt particles.

8.4.3 Dispersion of mineral dust particles

During May 2008 (period simulated) mineral dust was transported from the Saharan desert over Europe. Its effects on the radiative fluxes introduced a discernible bias on ground temperature in the NWP predictions for this period.

- Use run_cclm-art_dust. Change the SPDIR directory name in the runfile and set the **namelist** parameters for `ART` for a simulation for **only dust** particles.
- Plot horizontal cross sections of the mass and number concentration.

8.4.4 Dispersion of gases and ozone formation or depletion

This exercise uses the same setup as the first exercise (8.4.1) but the scenario is based on a dataset of actual anthropogenic emissions.

- Use run_cclm-art_gas. Change the SPDIR directory name in the runfile and set the **namelist** parameters for `ART` for a simulation **only for gases**.
- Plot horizontal and vertical cross sections of the different gases.

8.4.5 Dispersion of aerosols and formation of secondary aerosol particles

The final exercise shows the full potential of COSMO-ART, as secondary aerosols are calculated prognostically based on the concentrations of gaseous components.

- Use run_cclm-art_aero. Change the SPDIR directory name in the runfile and set the **namelist** parameters for `ART` for a simulation for **gases and aerosols**.
- Plot horizontal and vertical cross sections of the different aerosol particles.

8.5 INT2LM-TRCR

The meteorological preprocessor for COSMO - INT2LM (Documentation of COSMO-model, Part V: Preprocessing) - was designed to interpolate necessary initial and boundary conditions for meteorological quantities from a coarse grid onto the model grid in the atmosphere and at the surface as well as for the soil model. For more general datasets INT2LM was extended by Knote (2012) to interpolate e.g. aerosol or gaseous species initial and boundary conditions. It is also possible to

merge datasets by adding them or by replacing the coarser dataset on a limited domain by a finer grid dataset.

This extended preprocessor INT2LM-TRCR is controled by two logical namelist parameters within the CONTROL namelist: l_trcr is the main switch for the INT2LM-TRCR extension, l_trcr_nested indicates, whether initial/boundary conditions from a coarser COSMO-ART run are provided.

Further, a TRCRCONTROL and TRCRDATASET namelists are necessary. An example for these can be seen in 1 and 2. A further description of the usage of INT2LM-TRCR can be found in the documentation by Knote (2012).

```
&TRCRCONTROL
  ntrcr_ds      = 1,
/
```

Namelist example 1: TRCRCONTROL

```
&TRCRDATASET
  ie_tot=202, je_tot=192, ke_tot=7,
  startlon_tot=-20.17, startlat_tot=-15.07,
  dlon=0.17, dlat=0.17,
  pollon = -170.0, pollat = 43.0, polgam = 0.0,
  hinc=1.0, cvertdist=1.0,
  lconst_in_time=.FALSE., lconserve_mass=.TRUE.,
  ydirin="/mnt/workspc/kno134/int2cosmo/input/emissions/macc_test/",
  ylfn_prefix = "macc_", yvarlist= "SO2e","NO2e",
  yvertical_axis_type="geometric",
  yvertical_method="D", yinterp_type="N", ycombine_action = "O"
/
```

Namelist example 2: TRCRDATASET

Appendix A

The Computer System at DWD

A.1 Available Plattforms at DWD

The NWP exercises this week will be mainly carried out on the supercomputer system at DWD. This system consists of several machines, among them are

- `lxk00`:
  This is a Linux Cluster with 11 X4600 nodes and 8 AMD Opteron QuadCore processors (2.3 GHz). It runs SUSE Linux SLES 10 SP2. `lxk00` is used for compiling and linking, preparation of data, basic editing work and visualization of meteorological fields. It is not used for parallel programs, but jobs can be submitted to the NEC supercomputers.
- `NEC SX-8R`:
  The NEC SX-8R is a vector supercomputer consisting of 7 nodes with 8 vector processors each. Each processor has a peak performance of about 35 GFlop per second. There are 64 GB of memory per node available. For normal test jobs it is enough to use 4 or 8 processors. There are 2 SX-8R machines available called `sx8esiox` and `sx8wsiox`, resp. Tests will be carried out on the `sx8esiox`.
- `NEC SX-9 (not used during this training)`:
  The NEC SX-9 is the follow-up model of the SX series and consists of 14 nodes with 16 processors each. Every SX-9 processor has a peak performance of about 100 GFlop per second. There are 512 GB of memory per node available. The SX-9 is used at DWD for operations and research and is not available for testing this week.

There is a common filesystem across lxk00 and the SX-8R and every user has 3 different main directories:

- `/k/uhome/`<*username*> (`$HOME`)
  Directory for storing source code and scripts to run the model.
- `/k/uwork/`<*username*> (`$WORK`)
  Directory for storing larger amounts of data. There are no file size limitations, but data will be deleted without warning, if there is not enough free space on the file system.
- `/k/uscratch/`<*username*> (`$SCRATCH`)
  Directory for storing very large amounts of data that has also a very fast access. Data on this directory could be deleted without notice.

A.2 The Batch System for the NEC

Jobs for the NEC SX systems have to be submitted on `lxk00` with the batch system NQSII. Because the same batch system is running also on the Linux-cluster, the command names for the NEC SX have been prefixed with an `n`. Together with the source code of the programs we provide some run scripts in which all necessary batch-commands are set. Only some of them have to be adapted. See Section A.3 for more information on the run scripts.

Here are the most important commands for working with the `NQSII`:

`nqsub` *job_name*	to submit a batch job to `NQSII`. This is done in the run-scripts.
`nqstat @`*machine*	to query the status of all your batch jobs on a specific machine. You can see whether your jobs are `QUE` (qeued) or `RUN` (running). You have to submit jobs to machine `sx8esiox`.
`nqstat -Ps @`*machine*	to query the status of all batch jobs on a machine.
`nqdel` *job_nr*`@`*machine*	to cancel your job(s) from the batch queue of a machine. The *job_nr* is given by `nqstat`.

Note, that different binaries have to be created for the SX-8R and the SX-9. An SX-8R binary will run on the SX-9, but with less performance. An SX-9 binary will not run on the SX-8R.

A.3 NEC Run-Scripts for the Programs

The directories for the INT2LM and the COSMO-Model contain run scripts (e.g. `rungme2eu` and `run_cosmo_eu`, resp.) which cat together the NAMELIST-Input of the programs and start the binaries. These scripts are written for the NEC, but can be easily adapted to other platforms. All scripts work in a way that they create a batch description job (e.g. `int2lm_job` or `cosmo_eu_job`, resp.) which is submitted to the batch system (the `NQSII`). These scripts are more or less only examples. Before running the programs, you have to adapt the scripts / NAMELIST-Input to your needs. There are five parts in each script:

1. Number of processors
   Here you have to change the number of processors, but also some directory names and the date. Your settings will be transformed to the namelist input below.

`NPX, NPY`	for a parallel program these values can be chosen as needed, for a sequential program, both values have to be 1. On a vector computer like the NEC, `NPX` should always be chosen as 1, to make the innermost loop (the vector) as long as possible.
`NPIO`	for a parallel program additional processors can be chosen for doing the I/O (see also the User Guides); for a sequential program this value has to be 0. Unless working with very big domains ($\geq 800 \times 800$ grid points), using `NPIO` > 0 is not recommended!
`DATE`	To specify the initial time of the forecast.

   For the INT2LM

`LM_EXT`	To specify the directory with the external parameters for the COSMO-Model
`IN_EXT`	To specify the directory with the external parameters for the driving (input) model
`IN_DIR`	To specify the directory with the input data from the driving (input) model
`LM_DIR`	To specify the directory with the output data (which will be the input data for the COSMO-Model).

For the COSMO-Model

`INIDIR`	To specify the directory with the initial data.
`BD_DIR`	To specify the directory with the boundary data.
`OUTPUT`	To specify the directory where the result files are written.
`RESTAR`	To specify the directory where the restart files are written (if any).

2. Batch commands
On the NEC SX systems of DWD you have to start the programs using the `NQSII` (the batch system of NEC). This section of the run script sets most of the appropriate values. Eventually you have to modify the variables to specify the number of nodes and processors.

`#PBS -q normal@sx8esiox`	To put a job to the SX-8R (east side)
`#PBS -l cpunum_job=`	To specify the number of processors per node.
`#PBS -b`	To specify the number of nodes that should be used.

In most run-scripts examples, the settings are to use 1 node and `$NPY` processors.

In addition, you have to specify the working directory, where the job should be executed. If this directory is not set, the job runs in a temporary directory, but all ASCII output will be lost then. If a wrong directory is specified, the job terminates with an error message. This command tries to run the job in the home-directory of user `fegast3`.

```
cd /k/uhome/fegast3/
```

In some run-scripts also the settings for the IBM loadleveler are contained and can be safely ignored for these tests.

3. Cat together the INPUT*-files
In this part the INPUT-files are created that are needed by the INT2LM and the COSMO-Model, resp. The user can choose the NAMELIST-variables as needed. For a complete description of the NAMELIST-variables see *Part V* (Preprocessing: Initial and Boundary Data for the COSMO-Model) and *Part VII* (Users Guide) of the COSMO-Model Documentation.

For the purpose of these exercises, the most important Namelist-values for both programs are specified in the Chapter 2 and 3.

4. Run the programs
This part contains the call to invoke the program. On the NEC SX systems this is

`mpirun -nn` <*number_of_nodes*> `-nnp` <*number_of_processors_per_node*> *binary*.

If you are working only on one node (the INT2LM is running even on one processor!) it could also be

`mpirun -np` <*number_of_processors*> *binary*.

Also, some environment variables are set for the run on the NEC. *name_of_binary* can be the name of the program, if the working directory is set correctly, or you can also specify an absolute path name.

5. Cleanup
   This is to clean up the directory at the end of the run.

The last part in these scripts submits the batch jobs to the machine. For starting a program run, you just have to type the name of the run-script as a command, e.g.: `run_cosmo_eu`.

Appendix B

The Computer System at DKRZ

B.1 Available Plattforms at DKRZ

The climate simulation exercises will be carried out on the supercomputer at the DKRZ in Hamburg. This is an IBM p575 "Power6" cluster with a total size of 264 nodes with 16 dual core CPUs each. However, the system is installed in such a way that 64 tasks can be assigned per node. The peak performance of the total system is 158 TeraFlops/s.

The computer system is accessible at: `blizzard.dkrz.de`

The file system structure is as follows:

- `/pf/[a,b,k,m,u,g]/<username> ($HOME)`
  Directory for storing source code and scripts to run the model (8 GB quota per user, backup provided).
- `/work/<project>/<username>`
  Directory for storing larger amounts of data, which are accessed frequently (quota per project).
- `/scratch/[a,b,k,m,u,g]/<username> ($SCRATCH)`
  Directory for storing large amounts of data which are only needed for a short time. Files will be deleted frequently (generally after 14 days and after 7 days when space is limited).

The project ID for this years' training course is `bb0721`, thus the disk space in `/work/bb0721/<username>/` can be used for the exercises.

Note that the `blizzard` system should only be used for compilation and job submission. Any postprocessing should be performed on the `lizard.dkrz.de` computer which has access to the same file system.

B.2 The Batch System for the IBM

On the IBM, the `LoadLeveler` system is used for the submission of batch jobs. Generally, only few options have to be changed in the `LoadLeveler` commands for a new job.

Here are the most important commands for working with the `LoadLeveler`:

`llsubmit` *job_name*	to submit a batch job to the `LoadLeveler`. In the step-by-step exercises, this command has to be used to submit the job scripts; for the chain environment, this is included in the run-scripts.
`llq -u` *username*	to query the status of all your batch jobs. You can see in the column `ST` whether your jobs are waiting (`I`) or running (`R`).
`llcancel` *job_nr*	to cancel your job(s) from the batch queue of a machine. The job number *job_nr* is given by `llq`.

B.3 Script environment for running the COSMO-CLM in a chain

B.3.1 The scripts

To use the computer facilities effectively, the whole job chain is not carried out in just one script, but divided into five scripts to perform the main sub-tasks of the cclm chain:

prep.job.tmpl
All necessary preprocessing is performed by this script, e.g., copying boundary data from the archive and transforming them into int2lm compatible netCDF format. In the standard template copying is performed from hard disk. However, the script can be changed to perform copying from tape or via `ftp` or `scp`.

int2lm.job.tmpl
The job to run INT2LM. For detailed information about the INT2LM namlists see Chapter 2.

cclm.job.tmpl
The job to run CCLM. For detailed information about the CCLM namlists see Chapter 3.

post.job.tmpl
Any post processing needed is performed here. At first archiving of the results is performed in this script. In the standard template archiving is performed on hard disk. However, the script can be changed to perform archiving on tape or via `ftp` or `scp`. After archiving all other post-processing is performed, e.g. building time series.

and one main script to rule them all:

subchain
This main script prepares the environment variables for the other scripts, builds `*.job` files for the actual month in the chain from `*.tmpl` files and submit the actual `*.job` files.

Before you start your simulation, change the scripts to your needs. Fig. B.1 shows the job flow of these scripts.

Please keep in mind that you may have only a limited number of serial jobs you can run at the same time (on the DKRZ blizzard the limit is 10 serial jobs). To prevent your chain to be slowed down, save two serial jobs for the chain since `post` can run at the same time as `cclm` and the one month ahead `prep` and `int2lm`.

Changes to be made in the script `subchain`:

`EXPID` an experiment identification, alphanumeric characters

`PFDIR` the basis directory for the different experiments (has to exist)

`WORKDIR` a working directory on the `work` filesystem for log output (has to exist)

`SCRATCHDIR` a working directory on the `scratch` filesystem for intermediate output (has to exist)

`ARCHIVE_OUTDIR` a working directory on the `work` filesystem for storing the final model output (has to exist)

`YDATE_START/YDATE_STOP` start and end date of the simulation

`EMAIL_ADDRESS` is needed if the `mailx` commands are used and if `notification` is set to something else than `never` in the LoadLeveler commands.

`UTILSBINDIR` the path where the auxiliary program `cfu` can be found

`INT2LM_EXE` This variable can be found a little further down in the script and gives the name and full path of the INT2LM executable.

`CCLM_EXE` This variable can be found a little further down in the script and gives the name and full path of the COSMO-CLM executable.

The files in the `templates` subdirectory need only to be changed when other namelist parameters than those in the global settings of the `subchain` script have to be adapted. There is one parameter, however, which probably has to be changed in the `int2lm.job.tmpl`: this is the path and name of the external data, `ylmext_cat` and `ylmext_lfn` (or further variables if a totally different parameter file is used). Also, be careful that in `yinext_lfn` (name of coarse model input data files) you may use `cas` instead of `caf` depending on the coarse data.

The structure of the templates for the `int2lm` and `cclm` batch jobs contain three parts, similar to the parts 2-4 described for the NWP model system (section A.3):

1. Batch commands:
   Here, the LoadLeveler commands have to be set, such as the number of nodes and tasks per node the job should use:

   ```
   # @ node = 1
   # @ tasks_per_node = 64
   ```

   the jobname

   ```
   # @ job_name = int2lm_${EXPID}
   ```

   and the logfiles which will be written:

   ```
   # @ output = ${JOBLOGFILE}$(jobid)
   # @ error = ${JOBLOGFILE}$(jobid)
   ```

   The last command always has to be:
   `# @ queue`

2. Cat together the INPUT*-Files:
   This part contains all the namelist parameters which are set in the INT2LM and COSMO-CLM model. Those parameters which are not set here will be assigned to their default values. For a complete description of the NAMELIST-variables see *Part V* (Preprocessing: Initial and Boundary Data for the COSMO-Model) and *Part VII* (Users Guide) of the COSMO-Model Documentation.

3. Run the programs:
   The call for running the model in parallel mode on the IBM is:

   `poe` <*binary*> `-procs` <*number_of_processors*>.

B.3.2 The environment variables

The cclm chain distinguishes between four types of environment variables defined in the scripts subchain:

constant global
: these are used in at least two of the scripts and keep the same value throughout the whole simulation. They are defined at the beginning in the script subchain. Examples: basis directories, experiment id, grid description ($PFDIR, $EXPID, ...)

time dependent global
: these are used in at least two of the scripts and change their values during the simulation. They are defined at the beginning of the script subchain. Examples: current month ($CURRENT_DATE)

constant local
: these are used just by one of the scripts and keep the same value throughout the whole simulation. Actually they do not have to be defined at all, the values can be put directly into the script. However, it may become nicer and you can minimize errors if the same value has not to be changed at different places. Examples: number of processors ($NODES, $PPN)

time dependent local
: these are either used just by one of the scripts or may have different values in different scripts. They change their values during the simulation. Examples: the logfiles ($JOBLOGFILE)

In the template scripts these environment variables are written with a @ instead of an $ as first character (e.g. @EXPID instead of ${EXPID}) to make clear that these will be replaced via the sed command by the subchain script.

B.3.3 Running the CCLM chain

The steps for running the COSMO-CLM chain are:

1. Create a directory `${PFDIR}/${EXPID}` by hand (all others will be created automatically by the scripts). Copy the contents of `gcm_to_cclm` or `cclm_to_cclm` into that directory

   ```
   cp -R whateverdir/chain/<input>_to_cclm/*  ${PFDIR}/${EXPID}
   ```

2. Change the scripts to your needs

3. start the chain by typing

   ```
   ./subchain prep
   ```

If the job chain crashes you can start the chain again with any of the scripts by typing

```
./subchain <scriptname>
```

For `prep` (except at the very beginning of the chain) and `int2lm` you have to provide a second argument with the date for which the script should be started. The date cannot be taken automatically from the file `date.log` since `prep` and `int2lm` can also be run for one month ahead of the other scripts.

B.3.4 The directory structure

Fig. B.2 shows the directory structure of the CCLM chain simulation. By default the `SCRATCH` directory will be totally removed after job completion. The basic archive input and output directories have to be defined by the user at the beginning of `subchain` (see above), and further special treatments in the files `prep.job.tmpl` and `post.job.tmpl`, respectively.

B.3.5 Remarks

- `INC_DATE=MM:DD:HH` specifies the time increment for the CCLM job (the other parts of the chain are always monthly !!). Valid values are 01:00:00, 00:01:00, 00:02:00, ... , 00:27:00.
- On blizzard, the command 'mailx' only works on the login node, therefore you have to use `ssh plogin1` for sending an email. You need to set authorized keys to make it work in batch mode.
- Do not change any files in the directory `jobs`. This will have no effect, since these files will be overwritten by subchain with the files from the directory `templates`. If you want to make changes always apply these to the files in `templates`.
- It is assumend that not just the login node can submit jobs but also the compute nodes. If this is not the case on your computer system you need to find a work around (e.g. by using the `nqwait` command on the NEC-SX8).

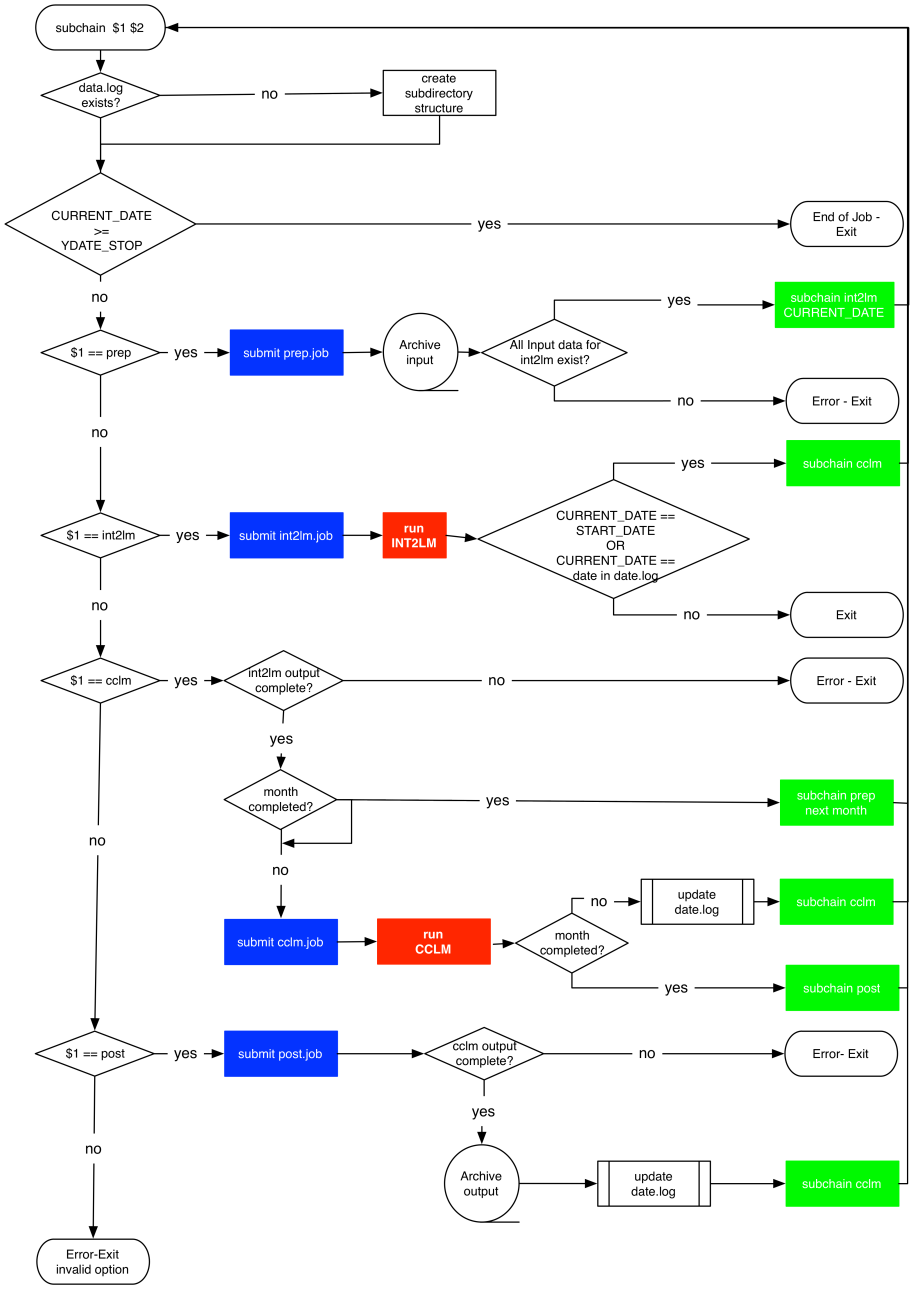

Figure B.1: Job Flow of the CCLM subchain scripts

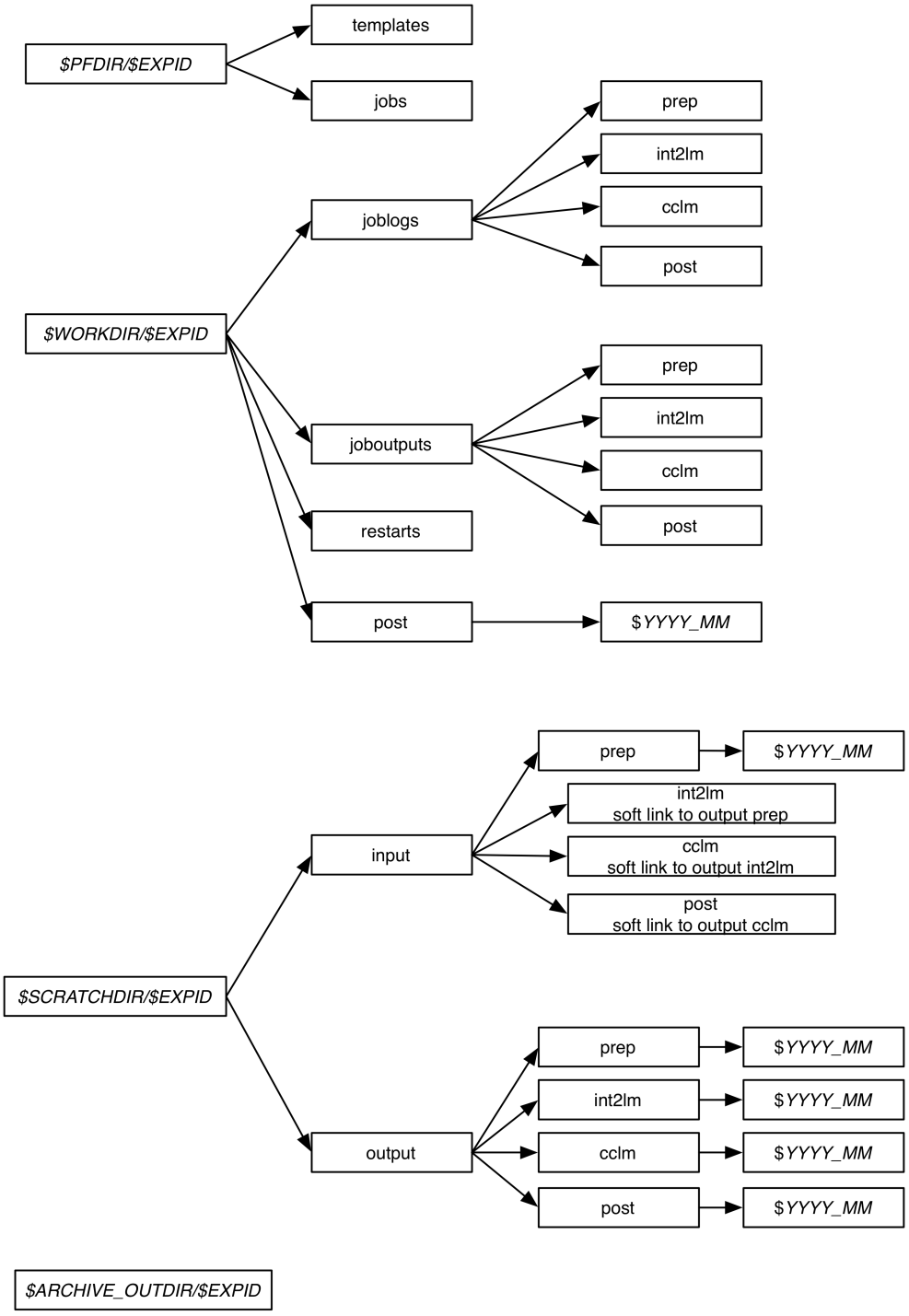

Figure B.2: Directory structure of the CCLM subchain scripts

Appendix C

Necessary Files and Data for the NWP Mode

C.1 The Source Codes

To do the installation and perform some test runs, the following files and data are needed. As a registered user you can download the data from DWD's ftp-server:

Address: `ftp-incoming.dwd.de`, User: `feu3`.

For this course you will find all necessary data on the `lxk00` in the directory

`/k/uhome/fegast3/COSMO_CLM_TRAINING_2013/`*subdirectories*.

Name	Description	Subdirectory
`DWD-libgrib1_110128.tar.gz`	GRIB1-library	`source`
`int2lm_yymmdd_x.y.tar.gz`	INT2LM source code	`source`
`cosmo_yymmdd_x.y.tar.gz`	COSMO-Model source code	`source`
`reference_data.tar.gz`	reference data set	`data`
`cosmo_dn_rrrrr_iiixjjj.g1_date`	external parameters for the COSMO-Model domain	`topo`
`invar.irrra.date`	external parameters for GME	`topo`

Here `yymmdd` means year, month, day, `x.y` the version number, `rrrrr` the resolution in meters and `iiixjjj` the size in grid points. The tar-files you should copy to your home directory (`$HOME`), the data to the work (`$WORK`) or the scratch (`$SCRATCH`) directory. The data you create with your tests should also be written to your work (`$WORK`) or the scratch (`$SCRATCH`) directory.

Available Model Versions (February 2013)

`int2lm_120831_1.20`

`cosmo_130207_4.26.2`

C.2 External Parameters

For fixed domain sizes and resolutions some external parameter files for the COSMO-Model are available. The files listed in the following table have a rotated pole with `pollat=32.5` and `pollon=-170.0`.

Data Set	Grid Size	Degrees	Meters
`cosmo_d1_07000_961x769.g1_2009121700`	961×769	0.0625	7000
`cosmo_d1_14000_481x385.g1_2009121700`	481×385	0.125	14000
`cosmo_d1_21000_321x257.g1_2009121700`	321×257	0.1875	21000
`cosmo_d1_28000_241x193.g1_2009121700`	241×193	0.25	28000
`cosmo_d1_56000_121x113.g1_2009121700`	121×113	0.5	56000

There are also 3 data sets with a rotated pole `pollat=40.0` and `pollon=-170.0`.

Data Set	Grid Size	Degrees	Meters
`cosmo_d0_02800_1605x1605.g1_2007082400`	1605×1605	0.025	2800
`cosmo_d5_07000_965x773.g1_2009121700`	965×773	0.0625	7000
`cosmo_d5_14000_483x387.g1_2009121700`	965×773	0.125	14000

Figures C.1 and C.2 show the areas for these domains. Note that the domain for `d0` is smaller than the one for `d5`.

External parameter data sets can be generated for any domain on the earth. Depending on the raw data sets available up to now, the highest resolution possible is about 2 km (0.02 degrees). At DWD there is a tool `EXTPAR` for creating the external parameters. The CLM Community has a slightly different tool `PEP` (Preparation of External Parameters), which can use additional raw data sets. More information on this issue can be found in the COSMO-Model documentation.

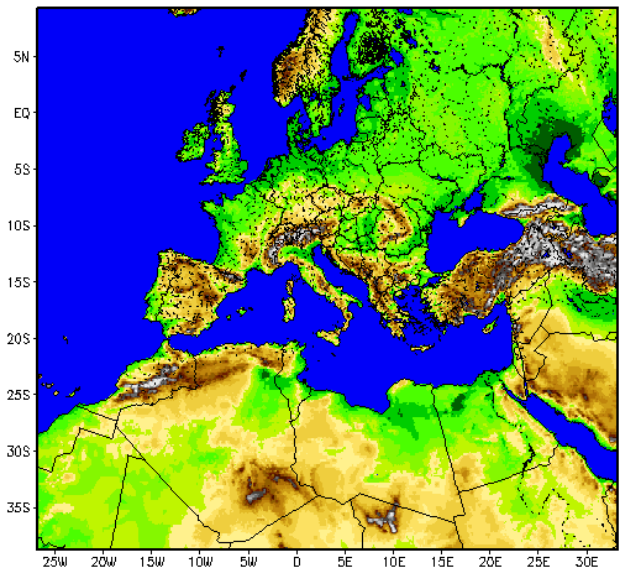

Figure C.1: Area for the domain `d1`

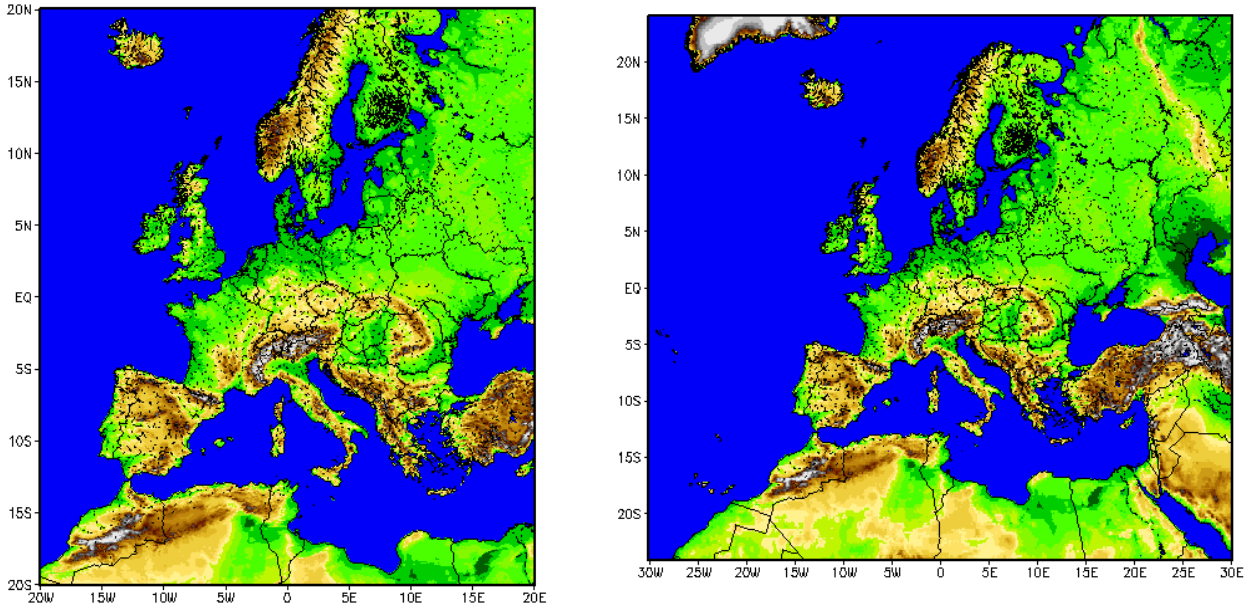

Figure C.2: Areas for the domains `d0` and `d5`

C.3 Driving Data

Hourly data are provided from the GME and the COSMO-EU for the following dates (for up to +24 or +48 hours):

- 15.08.2012 12 UTC: Heavy precipitation events
- 20.08.2012 00 UTC: Heavy precipitation events
- 20.01.2013 12 UTC: Freezing rain over Germany

Because of the amount of data, these data are in the work-directory of `fegast3`. They are provided together with a bitmap for domains over Europe. The data directories are:

- `/k/uwork/fegast3/GME_2012081512_sub/`
- `/k/uwork/fegast3/GME_2012082012_sub/`
- `/k/uwork/fegast3/GME_2013012012_sub/`

Appendix D

Necessary Files and Data for the Climate Mode

For the climate simulation, all the necessary code and input files to start working with the COSMO-CLM are included in the 'starter package' which is available on the CLM community web page. Copy this file to your work directory (`/work/bb0721/<userid>/` on blizzard). After unpacking the archive (`tar -xzvf cclm-sp.tgz`) you find 5 directories:

`src`	Source code for INT2LM, COSMO-CLM, the GRIB-Library and the auxiliary program package CFU
`data`	Data fields for initial, boundary and external parameters; Initial and boundary data are provided for two months (January and February 2000) from NCEP/NCAR reanalysis data.
`step_by_step`	seperate run scripts for a sample INT2LM and COSMO-CLM simulation, with two versions provided: one (subdirectory `gcm_to_cclm`) for an interpolation from coarse model input to COSMO-CLM and one (subdirectory `cclm_to_cclm`) for a nesting from COSMO-CLM to COSMO-CLM
`chain`	run scripts for the same sample INT2LM and COSMO-CLM simulations but with the complete chain script environment for long-term simulations.
`other_systems`	adoptions of the starter package for other computer systems. If you want to use these, please replace the files of the IBM Power6 version accordingly. Currently, templates are available for the NEC SX8 at the DWD.

The starter package is written for the Blizzard at DKRZ. Additional changes need to be applied on other machines. This is noted in the text for those cases we already know. The examples are built in a way that everything will be stored under the directory tree starting with `SPDIR` (create the help environment variable with `export SPDIR=$PWD/cclm-sp` in the directory where you have unpacked the starter package). Additionally, you will find a script `init.sh` in the main directory which initalizes the package by replacing the current `SPDIR` in the scripts (it has to be called once after unpacking the starter package). Running all examples and keeping all files will need about 50 GB disk space.

D.1 The Source Codes

For this training course we use a model version which is very similar to the one which is used for the next IPCC simulations for the 5th Assessment Report. The available model versions (February 2013) are:

`int2lm_091216_1.10_clm14`

`cosmo_090213_4.8_clm17`

D.2 External Parameters

For the 0.44° grid spacing we are using in this course, we provide the external parameter file (in the subdirectory `data/ext/`). The grid size of this file is `rlon = 105` and `rlat = 115` with a rotated pole at `pollat = 39.25` and `pollon = -162.0`.

Figure D.1 shows the area of this domain.

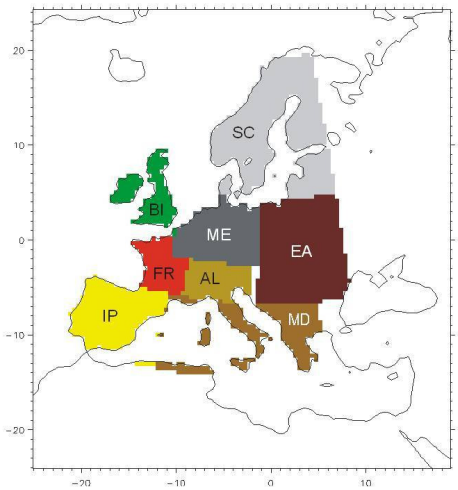

Figure D.1: Area for the domain of the provided external parameters.

External parameter data sets for any other domain and resolution can be generated by a tool called `PEP` (Preparation of External Parameters). This can be found on the CLM community home page. Generally, also the external parameters data sets which are provided by the DWD for NWP can be used (see Appendix D.2).

D.3 Initial and Boundary Data

For this course we will use the NCEP/NCAR reanalysis data as global model input. These data can be downloaded from the NCEP/NCAR homepage and are freely available for scientific applications.

On the `blizzard` computer in Hamburg, a preprocessed version (ready to use for INT2LM) of the NCEP data is stored in the directory `/work/coast/prj1/reanalysis/NCEP1/`.

The data are available at 6-hourly interval for the years 1948 – 2012. For this training course, we have selected two months of data: January and February 2000. These can be found in the subdirectory `data/gcm/ncep/2000_mm` with `mm=` 01 and 02.

D.4 Preprocessor and model system overview

The type of global model input which can be used for COSMO-CLM and the structure of the total workflow, as well as the name convention of intermediate files is shown in Figure D.2.

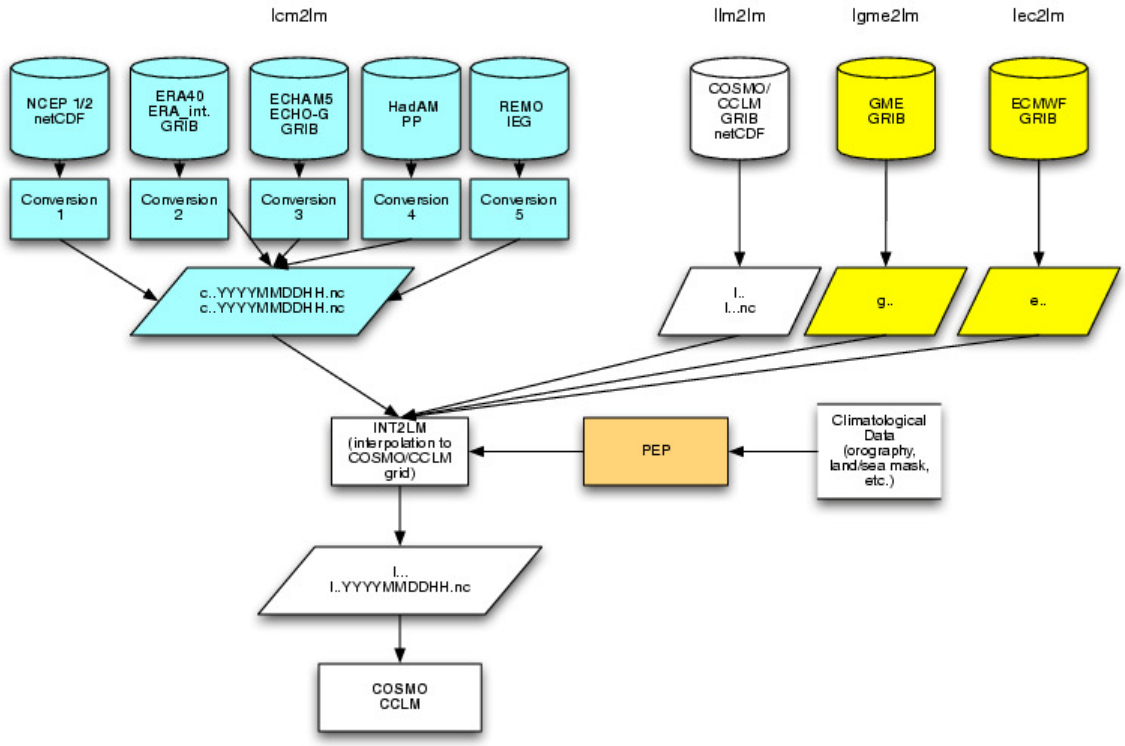

Figure D.2: Model chain structure of the COSMO-CLM.

GOOD LUCK

and

HAVE FUN