The Nonhydrostatic Limited-Area Model

LM (Lokal-Modell) of DWD

Part II :

Implementation Documentation

U. Schättler and G. Doms

Preliminary Version LM_F90 **October 1998**

Deutscher Wetterdienst

Geschäftsbereich Forschung und Entwicklung

Postfach 100465

D-63004 Offenbach

Germany

Contents

Chapter 1

Introduction

1.1 Preface

This report forms - in a first version - Part II of a comprehensive description of the new limited area model, the *Lokal-Modell* (LM), of the *Deutscher Wetterdienst* (DWD). It is intended to provide an implementation documentation including an introduction to special features, the basic design of the model code, the description of the different parts of the code and a subroutine glossary.

The LM is currently under development at the department of research (*Geschäftsbereich Forschung und Entwicklung*) of DWD. The authors of the LM documentation and the developers of the code are aware of the fact that all products, including this one, are far away from being complete. Also, incompatibilities between the documentation and rapidly changing versions of the code may occur. We apologize for any inconvenience caused by providing a first version of the Implementation Documentation of the LM in the stage of model development.

The authors of this Implementation Documentation recognize that typographical and other errors as well as discrepancies in the code and deficiencies regarding the completeness may be present, and your assistance in correcting them is appreciated. All comments and suggestions for improvement or corrections of this documentation, on the model code and related areas are welcome and may be directed to the authors or the Project Manager of LM, Dr. Jürgen Steppeler, at DWD.

For additional information on the *Geschäftsbereich Forschung und Entwicklung*, please consult our home page at `http://www.dwd.de`.

1.2 Organization of the LM-Documentation

For the documentation of the LM we follow closely the *European Standards for Writing and Documenting Exchangeable Fortran 90 Code*. These standards provide a framework for the use of Fortran 90 in European meteorological organizations and weather services and thereby facilitate the exchange of code between these centres. According to these standards, the model documentation is split into two categories: external documentation (outside the code) and internal documentation (inside the code).

In order for the internal documentation to be useful it needs to be both up to date and readable at centres other than that at which the code was originally produced. Thus, (most) of the internal documentation of LM is available in English. LM provides extensive documentation within the codes of the model subroutines. This is in form of procedure headers, section comments and other comments.

Following the *European Standards*, the external documentation of the LM comprises three parts:

- *Part I : Scientific Documentation*
  This part of the documentation provides information on the theoretical and the numerical formulation of the model covering both the basic thermodynamics and the parameterization of physical processes. Additionally, more technical aspects as the nesting technique, the initialization procedure and the data base of external parameters are also considered. Some pre- and postprocessing utilities are also described. The scientific documentation is independent of (i.e. does not refer to) the code itself.
- *Part II : Implementation Documentation*
  This part of the documentation, the report on hand, describes the particular implementation of the numerical methods of solution as presented in Part I. This includes information on the basic design of the model code and on the strategy for parallelization using the MPI library for message passing on distributed memory machines. A program glossary lists all routines (modules, subroutines, functions, etc) of the model together with a brief description of what they do. Information on the calling sequence of subroutines and function is provided by an overall calling tree of LM, and the model code structure is detailed by a number of flow charts. Additionally, the treatment of data input and output using *GRIB 1* I/O files is described.
- *Part III : User Guide*
  The User Guide of LM provides information on code access as well as hard- and software requirements for executing the code. Short tutorials on how to install and run the model are included. In addition, overviews are given on the computer systems at DWD and at the *European Centre for Medium Range Weather Forecasts* (ECMWF). Also, the programming environment

you have to deal with when working on the code or when running experiments is explained shortly. LM provides its users with flexible control over many parameters that can be used to configurate the model for various applications at different scales. Most of these parameters, including the model domain parameters, can be set at runtime without modifying and recompiling the code. These parameters are specified in a model input file, which is in NAMELIST format and grouped in blocks. The User Guide contains a detailed description of these control parameters and provides information on how to choose appropriate values. Recommended values and sensible value ranges are also given.

We plan to supplement the external documentation of the LM by an additional *Part IV : A Short Description.* This part will provide condensed information on the model formulation, the code structure, the handling of the model input/output files and the available products. Also, a short reference guide on how to run the model will be included.

1.3 Outline of the LM-Project

1.3.1 Development of a New Operational System at DWD

By the end of 1995 DWD has started with the development of a new modelling system for operational numerical weather prediction (NWP). The new NWP-system will be comprised of the global model GME and the nonhydrostatic limited area model LM. The global model GME is formulated as a gridpoint model with a semi-Lagrangian and semi-implicit time integration scheme and uses a triangular grid for finite differencing on the sphere. The *Lokal-Modell* LM will be nested within the GME covering a domain of about 2000×2000 km^2.

The operational use of the new system is scheduled for the late 90s. GME and LM will then replace the current operational system, which is made up of three models: the global spectral model GM with about 180 km horizontal resolution, the *Europa-Modell* (EM) with a grid spacing of 55 km for forecasts on the meso-α scale and the high resolution *Deutschland-Modell* (DM) with a grid spacing of 14 km for forecasts on the meso-β scale. The hydrostatic limited area models EM and DM are successively nested within the domain of the corresponding driving model, i.e EM is nested in GM and DM is nested in EM.

As a first step, it is planned to run the new NWP-system pre-operationally by the end of 1998 in a configuration that allows for meso-α scale global forecasts using GME with a horizontal resolution of about 60 km and for meso-β scale forecasts using LM with a grid spacing of about 8 km. With increasing computer power at DWD, it is expected that the final set-up of the new system can be run in the year 2001. This configuration will use a horizontal grid spacing of about 25 km for GME and of about 2.5 km for LM. In both models about 50 layers will be used for vertical discretization.

Boundary data for LM forecasts will be provided by a preprocessor program `GME2LM` that performs the horizontal and vertical interpolation of the coarse grid forecast data from a corresponding GME run. With respect to the initial conditions, a four-dimensional data assimilation cycle based on the nudging analysis method is planned for LM. At present, as long as the model development of GME is not completed and an analysis scheme is not available, LM is nested in DM or EM for test simulations on real data cases. A corresponding preprocessor program `HM2LM` has been written to provide initial and boundary data for LM simulations from hydrostatic EM/DM analyses and forecasts.

1.3.2 Goals, Objectives and Basic Model Design

The development of the *Lokal-Modell* is organized as an internal project of DWD. It was initiated by the head of the department of research and development, Dr. E. Müller, following two national workshops held at DWD in 1994. These workshops assessed the existing nonhydrostatic models and their history as well as the potential to use the available experience at universities and research institutes for a model development. DWD came to the conclusion that it would be appropriate to develop a new nonhydrostatic model taking into account the latest physical and numerical findings, recognizable technical requirements and specific demands resultig from the need of operational applications.

The principal objective of the LM-project is the creation of a meso-to-micro scale prediction and simulation system. This system, with the *Lokal-Modell* LM as its basic model component, is intended to be used as a flexible tool for specific tasks of weather services as well as for various scientific applications on a broad range of spatial scales.

Current operational NWP-models operate on the hydrostatic scales of motion with grid spacings down to about 15 km. Thus, they lack the spatial resolution required to capture explicitly all small-scale, short-duration severe weather events and significant flow systems, which are related to the nonhydrostatic scales of motion. LM is designed for just these spatial scales where nonhydrostatic effects play an essential role. The general goal of the LM-system is to predict as accurately as possible the evolution of atmospheric flow at the following scales of motion (in terms of a typical horizontal grid spacing Δs):

- Meso-β scale phenomena (Δs: 5 km to 50 km),
- Meso-γ scale phenomena (Δs: 500 m to 5 km),

- Micro-α scale phenomena (Δs: 50 m to 500 m).

Of course, the major use of the model will be the application to operational weather forecasting within the new NWP-system as described in the previous section. The key issue for this application, which will be first on the meso-β scale and finally on the meso-γ scale, is the numerical prediction of near-surface weather focussing on clouds, fog, precipitation, orographically and thermally forced local wind systems, and on severe weather events such as supercell thunderstorms, intense mesoscale convective complexes, prefrontal squall-line storms and heavy snowfall from wintertime mesocyclones.

By using grid spacings of about 1 to 3 km for operational forecasts on a large domain, it is expected that deep moist convection and the associated feedback mechanisms with the larger scales of motion can be resolved explicitly. Thus, meso-γ scale NWP-models in principle have the potential to overcome the shortcomings resulting from the application of parameterized convection in nowadays coarse-grid hydrostatic models. Additionally, the impact of topography on the organization of penetrative convection by, e.g., channeling effects, is represented much more realistic in high resolution nonhydrostatic forecast models.

A number of additional operational applications are planned at DWD. These include the use of LM as a meteorological preprocessor to drive connected models as, e.g. for the transport and deposition of chemical and radioactive pollutants, and the application of the modelling system for nowcasting purposes. With regard to the latter, the model should be able to accommodate data from weather radars and wind profilers by a suitable assimilation strategy and to serve as a tool for evaluating the predictability of small-scale severe weather events. It is also planned to use LM for model-based expertises related to the evaluation of regional climate on the meso-γ and the micro-α scale. Rather than to investigate the regionalization of global climate change on local scales, the purpose is to estimate quantitatively the impact of specified changes of the surface parameters, as e.g. land-use or terrain structure, on local climate.

Apart from these operational applications of LM, the project intends to provide a nonhydrostatic modelling framework for various scientific and technical purposes. Examples are applications of the model to large-eddy simulations, cloud resolving simulations, studies on orographic flow systems and storm dynamics, development and validation of large-scale parameterization schemes by fine-scale modelling, and tests of computational strategies and numerical techniques. For these type of studies, the model should accommodate both real data cases and artificial cases using idealized initial data.

As a specific scientific application it is planned to apply a special high-resolution version of the model ($\Delta s = 100$ m), the *Lindenberg LM* (LLM), within the LITFASS project of DWD (Müller et al., 1995). The scientific objective of this project is the determination and parameterization of mean fluxes of momentum, heat, moisture and other substances, representative for a horizontal scale of about 10 km, over heterogeneous land surfaces. A combined approach based on both measurements and detailed fine-scale modelling with LLM has been chosen to determine the representative fluxes in this long-term study.

The wide range of possible applications imposes challenging requirements on the physical, numerical and technical design of the model. The basic design requirements are listed below:

- Use of nonhydrostatic, compressible (i.e. unfilterd) dynamical equations to avoid restrictions on the spatial scales and the domain size;
- Application of an efficient numerical method of solution;
- A comprehensive physics package must be provided to cover adequatly the spatial scales of application;
- Flexible choice of initial and boundary conditions to accommodate both real data cases and idealized initial states;
- Use of a mesh-refinement technique to focus on regions of interest and to handle multi-scale phenomena;
- High resolution data sets for all external parameters required by the model must be provided;
- Use of pure Fortran-90 constructs to render the code portable among a variety of computer systems;
- Use of the standard MPI-software for message passing on distributed memory machines to accommodate broad classes of parallel computers.

1.3.3 Project History and Plans

We have chosen not to develop an entirely new model from the very beginning, but to rely on a more incremental strategy for the development. The first version on LM was constructed from our nested regional model DM by replacing its hydrostatic solver by a nonhydrostatic dynamical core and adapting the physics routines accordingly. For the formulation of the nonhydrostatic dynamics we followed closely the MM5-model as described by Dudhia (1993) and Grell et al. (1994). In this way, a prototype version of LM could be readily realized and we could start to gather some experience in nonhydrostatic modelling in an early stage of the project. The model had then to be upgraded gradually to meet the basic design requirements mentioned above. This work is still in progress.

The LM-project team has prepared a multi-year plan that describes in detail incremental changes to upgrade the modelling system. The principal milestones of the LM-project are summarized shortly:

1996: Selection, implementation and testing of the dynamical algorithms;
parallel processing and use of Fortran 90;

1997: Selection, implementation and testing of the physics package;
Development of a nudging analysis scheme;
set-up of an assimilation cycle;

1998: Interface to the GME as driving model;
Quasi-operational forecasts with the LM;
improvements in the physics and the numerics;

1999: LM with 8 km grid spacing replaces the DM as operational regional model;
further upgrading of physics and numerics;

2001: LM with 2.5 km grid spacing runs operationally;
final documentation.

1.3.4 Project Personnel and Collaborations

In order to broaden the expertise of model development, LM is offered to be a cooperative project. At present, a number of national research institutes and universities as well as the Swiss Meteorological Institute take part in the project. Collaboration ranges from the use of the LM for case studies to taking responsibility for work packages related directly to the model development. Also, a number of visiting scientists have significantly contributed to the upgrading of the model.

Within the framework of the SRNWP (Short-range Numerical Weather Prediction) initiative of the EUMETNET network of the European weather services, DWD has taken responsibility for the topic "Nonhydrostatic Models" as a *Centre for Nonhydrostatic Modelling* (CNM). A first workshop was held in 1996 to define the role of CNM (workshop reports are available from DWD). CNM will facilitate exchange of scientific and technical information by yearly workshops, newsletters, etc., organize common numerical experimentation, define various tests for nonhydrostatic models that are currently under development in Europe and keep contact to other SRNWP centres. The focus of CNM will be on model applications to numerical weather prediction. LM takes part in this European cooperation.

The basic model development is done at DWD where different sections of the research department contribute to the LM-Project. Below are listed some useful contact persons and their responsibilities as of April 1998:

- *Jürgen Steppeler*
  Project Manager. Responsible for overall project coordination and planning, external collaborations and general support.
  e-mail: jsteppeler@dwd.d400.de

- *Günter Doms*
  Model development, dynamical part. Responsible for the basic design of the model.
  e-mail: gdoms@dwd.d400.de

- *Erdmann Heise*
  Physical parameterizations.
  e-mail: eheise@dwd.d400.de

- *Ulrich Schättler*
  Programming and documentation. Responsible for the design of the program code, software standards and documentation.
  e-mail: uschaettler@dwd.d400.de

- *Michael Gertz*
  Source Code Management. Responsible for administrating and maintaining the different versions of the program, the programming environment and auxiliary libraries.
  e-mail: mgertz@dwd.d400.de

- *Hans-Joachim Herzog*
  Development of the high resolution model version LLM.
  e-mail: herzog@mop.dwd.d400.de

1.4 Model Policy, Code Access and Support

This section summarizes some information on the general model policy, how to access the code, management of the code and support for users. A detailed list of the available programs, documents and products is given in Part III of the LM-Documentation, the User's Guide.

(a) *Model Policy*
The LM and also some other numerical models of DWD are available free of charge (or for the cost of media) for scientific and educational purposes, especially for cooperational projects with DWD. However, all users are required to sign an agreement with DWD and observe certain conditions and restrictions on code usage. In the case of a planned commercial use of the models, special regulations will apply. For questions concerning the request and the agreement for LM please contact the Project Manager, Jürgen Steppeler.

(b) *Code Access at DWD and ECMWF*
Registered users for DWD's and ECMWF's computer centres can access the program codes on Cray T3E, C90 or J90 (at DWD) or on the Fujitsu VPP700 (at ECMWF). For that purpose not only the validation for the

computer but also the validation for the model is needed. Please contact the source code administrator Michael Gertz at DWD for your validation for a specific program code.

(c) *Code Access via ftp*
For external users there is the possibility to get the programs, the programming environment and all necessary scripts via the ftp-server of DWD (see (???, 1997)).

(d) *Source Code Management*
At DWD, the different versions of all NWP models including the LM and related software components are maintained using the Source Code Control System (SCCS). SCCS is a standard UNIX tool for the management of a software system. For safety reasons, only a single person - the administrator - is responsible for the routine programs. Only the administrator is authorized to modify the source code permanently and to implement new routines and libraries in SCCS. Of course, all other persons - the users - can develop modifications, extensions and user-owned test programs which, however are not implemented directly into the SCCS library. Only after careful testing, modifications can be submitted to the administrator, who will implement them into the SCCS source code library by defining a new version of the model.
For ease of use command procedures were developed to support programmers and users. External users can get these procedures, the *programming environment*, together with the model, but they have to have the UNIX tool SCCS on their machines. For more details see the User Manual.

(e) *Support*
DWD is not equipped to provide extensive support to users of its NWP models. If problems occur we would kindly ask you to contact any of the staff members of DWD listed in Section 1.3.4. We try to assist you as well as possible.

1.5 LM Characteristics and Features

The *Lokal-Modell* of the DWD is a nonhydrostatic limited area atmospheric prediction model. It is designed for operational numerical weather prediction as well as various scientific applications on the meso-γ and meso-β scale. The 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.

The recent release of the model, Version LM_F90 is significantly different from the previous Fortran77 release and a large number of features have been added.

The basic characteristics of the present release, LM_F90 1.10, are summarized by the following list.

- **Equations**
  Nonhydrostatic, full compressible, no scale approximations.

- **Coordinate System**
  Generalized terrain-following vertical coordinate and rotated geographical coordinates. Equidistant grid spacing in the horizontal directions and user defined grid stretching in the vertical.

- **Model Geometry**
  1-D, 2-D and 3-D model configurations. Metrical terms can be adjusted to represent tangential cartesian geometry with constant or zero Coriolis Parameter.

- **Prognostic Variables**
  Physical components of wind velocity respective the orthogonal z-system, perturbation pressure, temperature, specific humidity, cloud water content and (optionally) cloud ice content.

- **Diagnostic Variables**
  Total density, precipitation fluxes of rain and snow.

- **Spatial Discretization**
  Second-order finite differences for advection and other terms on an Arakawa C-grid. Vertically, the Lorenz-grid is used for the staggering of model variables.

- **Time Discretization**
  Second-order leapfrog for large time steps with Asselin time filter. First-order forward-backward scheme for small time steps.

- **Time Integration**
  Split-explicit with vertically implict option for sound and gravity wave propagation.

- **Initial Conditions**
  Options for real data cases with interpolated initial data from a coarse-grid driving model (DM) and user-specified idealized initial fields.

- **Lateral Boundary Conditions**
  Options for externally forced (Davies-type) and periodic conditions.

- **Top Boundary Conditions**
  Options for rigid lid conditions and Rayleigh damping layer.

- **Nesting**
  One-way nesting in a coarse-grid hydrostatic driving model (DM). One-way self-nesting is also available. Interactive self-nesting to be added.

- **Numerical Smoothing**
  Options for 2nd- and 4th-order linear horizontal diffusion and 2nd order nonlinear diffusion. Option for divergence-type damping to control horizontal acoustic waves.

- **Initialization**
  Initialization of unbalanced initial states using a digital filter. Options for adiabatic and diabatic initialization.

- **Subgrid Scale Turbulence**
  Diagnostic second order closure of hierarchy level 2 for vertical turbulent fluxes. Options for turbulent kinetic energy formulation and isotropic as well as anisotropic 3-D treatments of turbulence to be added.

- **Surface Layer Parameterization**
  Stability-dependent drag-law formulation of momentum, heat and moisture fluxes according to similarity theory.

- **Grid Scale Clouds and Precipitation**
  Options for a warm-rain (Kessler scheme), a 1-category ice (from EM/DM) and a 2-category ice (cloud ice scheme) bulk parameterization of cloud microphysics. Further options to be added.

- **Longwave and Shortwave Radiation**
  The radiation scheme employs eight spectral intervals and is based on the solution of the two-stream version of the radiative transfer equation. Direct radiation interaction with cloud water and cloud ice is allowed for. Subgrid-scale cloudiness is diagnosed from relative humidity.

- **Soil Model**
  Two-layer model for heat budget formulated according to the extended force-restore method. Two-layer or optionally three-layer model for the water budget including interception storage of water and snow.

- **Cumulus Parameterization**
  Tiedke mass-flux scheme for nonresolvable moist convective processes with equilibrium closure. Scheme with non-equilibrium closure to be added.

- **Terrain and Surface Data**
  Orography, soil and land-use data are available at 0.0125, 0.025, and 0.06 degrees model resolution for the domain of DM (covering Germany and surrounding countries).

- **Code Parallelization using MPI**
  The Parallelization is done by domain decomposition. The *Message Passing Interface* software (MPI) is used for message passing on distributed memory machines.

- **Compilation of the Code**
  The model code is written in Fortran 90. The compilation of all programs is performed by a Unix shell script invoking the Unix *make* command. All dependencies of the routines are automatically taken into account by the script.

LM_F90 runs on conventional scalar machines (e.g. IBM and SGI workstations), vector supercomputers (e.g. CRAY-YMP, CRAY-C90 and Fujitsu VPP700) and MPP machines with an implementation of the MPI library for message passing.

Most of the model features listed above are described in detail in corresponding chapters of the scientific documentation.

Chapter 2

Portable Program Development

2.1 Introduction

The main goal for the source code development of the LM is the portability. This means that the same code has to run on every computer platform without having to change the source itself. The term *platform* refers not only to different processors but also to different architectures ranging from workstations to distributed memory parallel supercomputers. Portability influences the following aspects:

- *Use of a standard programming language*
  Fortran 77 is used as programming language in many meteorological applications. But because it lacks some desirable features for large software packages, most vendors introduced extensions that are not portable to other machines. These extensions are now included in the Fortran 90 standard. Therefore it was decided to use Fortran 90 without any vendor dependent extensions for the LM.

  Note that the program will not run with old Fortran 77 compilers, because new constructs of Fortran 90 are used. This is described in more detail in Section 2.2.

- *Parallelization of the model*
  The LM is parallelized for distributed memory parallel computers. Section 2.3. explains the parallelization and the programming paradigm used. The library that was chosen for explicit message passing is the Message Passing Interface (MPI). MPI is adopted as a standard by nearly all computer vendors and efficient implementations are available for their machines. For sequential platforms having no MPI implementation, dummy interfaces for the MPI routines are provided for LM. In this way the same code can be executed also on normal workstations.

- *Grib code interface*
  The LM reads and writes data decoded in Grib Code (grided binary), a standard coding that was adopted by the World Meteorological Organization (WMO) to ensure that meteorological data can be exchanged between weather services all over the world. The version used for the LM is Grib 1. Unfortunately, the routines necessary for decoding and encoding Grib files make use of special hardware features of the machine, therefore they cannot be written portable. All these routines are grouped together in a supplementary library which has to be implemented on specific hardware platforms. For a description of the interface see Section 2.4.

2.2 Fortran 90 Features

For the documentation and the programming of the LM we closely follow the *European Standards for Writing and Documenting Exchangeable Fortran 90 Code*. In the following some of the new Fortran 90 features used in the LM are explained, namely the dynamic memory allocation, the use of `MODULE`s and the `KIND`-type parameter.

Dynamic memory allocation
In Standard Fortran 77 arrays could only be allocated at runtime if their dimensions were declared with fixed sizes, i.e. with parameters that were known at compile time. As a consequence the program had to be recompiled, if the dimensions of the arrays were changed. Especially for a parallel program, where the different subdomains can have different dimensions, code development was not very comfortable.

Fortran 90 offers dynamic memory allocation. Arrays can be declared and dimensioned using variables that are only known at runtime. There are three different ways of obtaining dynamic memory in Fortran 90:

1. *Automatic arrays*
   These are arrays initially declared within a subprogram whose extents depend upon variables known at runtime. An automatic array is undefined outside the scope of the subroutine in which it is declared and memory is freed automatically.

2. *Allocatable arrays*
   Array variables declared with the `ALLOCATABLE` attribute may be allocated space at runtime by using the `ALLOCATE` command. However, unlike `POINTER`s, allocatable arrays are not allowed inside derived data types.

3. *Pointer arrays*
   Array variables declared with the `POINTER` attribute may be allocated space at runtime by using the `ALLOCATE` command.

Memory allocated using 2. or 3. above must be explicitly freed using the Fortran 90 statement `DEALLOCATE`. In the LM 2. is used for the meteorological fields used throughout the program and for variables used in several routines. Additional space necessary for computation in single routines is defined using automatic arrays.

Use of `MODULE`*s*
Fortran 90 supports a modular development approach by grouping together variable declarations and subprograms into `MODULE`s. A `MODULE` is a program unit (also: compilation unit) containing a specification part (declaration of data, namelist groups, procedure interface blocks) and a subprogram part (procedure definitions). Procedures contained in modules are called module procedures.

In the specification part, variables can either be declared or `USE`d from other modules. All module procedures defined in the subprogram part of a module have access to the variables declared or used in the specification part. With modules, several subroutines that solve a special problem and the necessary data can be grouped together in one package.

In the LM modules are used in different ways:

1. *Source modules* group together subroutines as module procedures for special parts of the forecast to a package. The data necessary for the source modules are defined via `USE` lists in the specification part. Not every module procedure is known to the program using a module. Only special driver or organization routines can be called from the main program. In this way the interface of a package to the main program is clearly defined.

2. *Utility modules* contain routines for special tasks that are independent of the source modules, like time measurement or determination of the actual date and time. All routines dealing with the message passing library are also placed in utility modules.

3. *Data modules* contain only a specification part, i.e. the declaration of variables. They replace the `COMMON` blocks used in Fortran 77.

The LM consists of several source modules, e.g. for integrating the model equations (dynamics) or for the different parametrization packages. The basic design and the structure of the LM using modules is described in Chapter 3.

`KIND`*-type parameter*
Fortran 90 parametrizes the intrinsic data types, i.e. the programmer can choose a variant of a data type (e.g. `REAL` or `DOUBLE PRECISION` for real variables) by specifying a type parameter, called the `KIND`-type parameter. If no `KIND`-type parameter is specified, a default value is assumed by the system. Note that the default

need not be the same on different systems. There are two different intrinsic procedures to determine the `KIND`-type parameter of a given value or variable. `KIND(`x`)` can be used to determine the `KIND`-type parameter of x. `SELECTED_REAL_KIND (`p, q`)` returns a `KIND`-type parameter which guarantees a precision of at least p digits and an exponent range up to q. In the same way `SELECTED_INT_KIND (`p`)` provides a `KIND`-type parameter to represent integer variables with at least p digits. The LM specifies the real and integer data types with the parameters

- `iintegers = KIND (1)`
  this gives the default integer of the system used.
- `ireals = SELECTED_REAL_KIND (12,200)`
  this gives a real data type with a precision of at least 12 digits. Therefore, variables of this `KIND`-type parameter are stored using 8 bytes, no matter on which machine.

To choose a different variant of these data types, only the parameters `iintegers` or `ireals` have to be changed.

2.3 Parallelization of the Model

The LM is parallelized using the data parallel programming paradigm. While task parallelism means that a program is divided into several independent tasks that can run in parallel, data parallelization distributes the data over the processors. MPI uses the Single Program Multiple Data (SPMD) model where every processor executes the same program but on different data.

For distributing the data to the processors, the two-dimensional domain decomposition is used, i.e. the total domain is decomposed along the latitudes and longitudes. The vertical dimension is not decomposed because the data dependencies would require too much communication otherwise.

The processors available for a model run are organized in the same way as the decomposition, i.e. in a two-dimensional cartesian grid. When a forecast starts, one processor reads in the specification variables via the namelists and distributes them to all other processors. Then every processor computes the decomposition and the location of its subdomain in the total domain. This also includes information on the neighbours and processor groups for special tasks.

During the forecast, data have to be exchanged between the processors, especially between neighbouring ones. While computing the forecast, every processor can be considered of running an own regional model for its special subdomain.

The implementation of the decomposition and the organization of the parallel program are described in Section 5.3.

2.4 Grib Code Interface

Reading and writing Grib code data requires a set of routines for handling the following tasks:

- file management
    - `copen`
    - `cclose`
    - `crewind`
    - `cinquire`
    - `cback`
- reading and writing
    - `cuegin`
    - `cuegex`
- packing and unpacking the data
    - `grbin1`
    - `grbex1`
    - `refstf`
    - `irefts`

These routines (and further subroutines called by them) are included in a special grib library. On DWD's systems this library is called `dwdlib`. The implementation of the `dwdlib` depends on the size of the standard real- and integer-variables of the computer used and therefore is not portable, but has to be adapted to the special machine. The routines for file management, reading and writing the data are written in C. Because the Fortran-C interface differs from platform to platform, this is also a non-portable part of the grib library.

To adapt the size of the variables used in the LM to the size of the variables used in the grib library, additional `KIND`-type parameters have been introduced in the LM. These are

- `intgribf`
  `KIND`-type for Fortran `INTEGER` variables of the grib library.
- `intgribc`
  `KIND`-type for default C `INTEGER` variables of the grib library.

- `irealgrib`
  `KIND`-type for default `REAL` variables of the grib library.

Most important for the implementation of the grib library is the size of the Fortran `INTEGER` variables used. Another parameter used in the LM therefore indicates the size of these:

- `iwlength`
  Length of the Fortran `INTEGER` variables of the grib library in bytes.

All variables that are passed from the LM to the grib library via the parameterlists have to be declared with the `KIND`-type parameters given above. If necessary, the variables have to be converted to the size used in the LM using the intrinsic functions `INT` and `REAL`, respectively.

Chapter 3

Basic Design of the Program

3.1 Introduction

In this chapter the structure of the program is explained. Section 3.2 to Section 3.5 document the main program and the different modules and give a brief description of what they do and what they contain. The detailed implementation documentations of the modules can be found in Chapter 4 (for the source modules) and in Chapter 5 (for the utility modules).

All subroutines of the program are grouped together in modules. A module is an implementation of a *package*, a term defined by (Kalnay E., 1989) regarding the physical packages, i.e. the parameterization of the atmospheric subgrid-scale physical processes. This term is also used in the *European Standards for Writing and Documenting Exchangeable Fortran 90 Code* ((Andrews Ph., 1995)). These Standards state:

> *A package shall provide separate set up and running procedures, each with a single entry point. All initialization of static data must be done in the set up procedure.*

In the LM there are different levels of modules:

1. The top level is the main program `lmorg` (file `lmorg.f90`. Note, that this is a `PROGRAM`, not really a `MODULE`). `Lmorg` uses the source modules to perform the forecast.

2. The source modules are the packages that solve the different tasks of the forecast. If appropriate, there are routines for initializing variables of the package. In any case there are driving routines that organize the tasks of the package. Only these routines are useable by the main program (they are `PUBLIC` while the others are `PRIVATE`). The source modules do not use each other, so there is no interference between them.

3. The utility modules contain routines like time measurement or determination of the actual date. The routines necessary for parallel execution of the program are also included in utility modules. They act as a layer between the program and the message passing library which is used.

4. The data modules form the data pool from which the source modules can fetch the variables for computation. They replace the `COMMON`-blocks of Fortran 77.

3.2 The Main Program

Description
`lmorg` is the main program of the model that organizes the forecast. To handle the different tasks, it uses the source modules of the second level. The special tasks are

- Setup of the model.
- Input of Grib coded fields
- Initializations of data needed in the source modules and digital filtering
- Time stepping (physics, dynamics, diagnostics, output of data)
- Cleanup of the model

`lmorg` contains several internal procedures:

- `initialize_loop` prepares variables for each new iteration of the time loop.
- `finalize_loop` performs actions necessary at the end of an iteration of the time loop (reinitializing variables, determination of actual UTC time and date).
- `organize_diagnosis` is the driving routine for the diagnostic computations. It determines which diagnostic computations have to be performed.
- `organize_print_diag` is the driving routine for the ASCII output of the diagnostic computations.
- `near_surface` calculates values near the surface.

3.3 The Source Modules of the LM

The source modules are the packages that solve the different tasks of the forecast. The source modules of the LM are:

- `src_setup.f90`
- `src_artifdata.f90`
- `src_input.f90`
- `src_output.f90`
- `src_gscp.f90`
- `src_radiation.f90`
- `src_turbulence.f90`
- `src_soil.f90`
- `src_convection.f90`
- `src_dynamics.f90`
- `src_relaxation.f90`
- `src_meanvalues.f90`
- `src_diagnosis.f90`
- `src_differences.f90`
- `src_gridpoints.f90`

Some of these modules have several module procedures. In order to keep the Unix files at a reasonable size, the procedures of some modules are held in extra files (with the name `name_of_procedure.incf`) and are `INCLUDE`d in the module. In the lists of the module procedures below, these routines are marked with the extension `.incf`.

Description of the source modules:

- `src_setup.f90`
  Source module for the setup of the LM

  Description
  This module performs the setup of the model. Special tasks are:

  - The initialization of the environment (sequential or parallel); (if running in parallel mode: MPI-Initialization).
  - Initialization of the timing.
  - Reading (and distributing) the Namelist groups.
  - If running in parallel mode: domain decomposition (in sequential mode only further variables are set).
  - Allocation of space for the meteorological fields.
  - Calculating mathematical, physical and other constants.
  - Writing a coverpage for the ASCII output.

  Module procedures included

`organize_setup.incf`	`input_lmgrid.incf`
`input_dynctl.incf`	`input_runctl.incf`
`input_inictl.incf`	`input_diactl.incf`
`input_phyctl.incf`	`input_gribin.incf`
`input_environment.incf`	`input_nudging.incf`
`input_gribout.incf`	`calc_ngrib.incf`
`alloc_meteofields.incf`	`constants.incf`
`coverpage.incf`	`dealloc_meteofields.incf`

  Public procedures are
  `organize_setup, dealloc_meteofields`

- `src_artifdata.f90`
  Source module for generating artificial data for the LM.

  Description
  Artificial data can be generated for the LM for special purposes, e.g. theoretical tests, tests for runtime performance, etc. The user has to reprogram the two routines included in this module according to his/her needs to generate appropriate initial and boundary conditions.

  Module procedures included

  `gen_ini_data` `gen_bound_data`

  Public procedures are
  `gen_ini_data, gen_bound_data`

- src_input.f90
  Source module for reading Grib files.

  Description
  This module contains the routine necessary for reading GRIB files for the initial and boundary data needed for an LM forecast.

  The records of a GRIB file are read in by one processor (PE 0), which distributes them to the processors of the cartesian grid, such that each PE gets one GRIB record. The GRIB data can then be unpacked in parallel. After unpacking, every PE has a two-dimensional field for the total domain. It decomposes this field and sends the subparts to the appropriate PEs.

  The bottleneck in this implementation is the sequential reading of the GRIB file. In the future, implementations based on parallel I/O or asynchronous I/O will be investigated.

  The procedures of this module that use message passing routines are used from the utility module io_utilities.f90.

  Module procedures included

organize_input.incf	fill_procarray.incf
get_vertcoord.incf	bd_from_id.incf
id_mix_bd.incf	

  Public procedures are
  organize_input

- `src_output.f90`
  Source module for writing Grib files.

  Description
  This module contains the routine necessary for writing GRIB files output of the LM.

  It is possible to write two-dimensional fields on model levels, on pressure levels and on height- (or z-)levels. In case of pressure- or z-levels, a vertical interpolation on the appropriate levels is performed with tension splines.

  The subdomains of a total two-dimensional field are gathered to one PE, such that each PE gets a total field that should be written. The data are then packed to GRIB records in parallel. Every PE sends its GRIB record to PE 0, which writes it to disk. The bottleneck in this implementation is the sequential writing of the GRIB file. In the future, implementations based on parallel I/O or asynchronous I/O will be investigated.

  The procedures of this module that use message passing routines are used from the utility module `io_utilities.f90`.

  Module procedures included

`init_output.incf`	`organize_output.incf`
`output_grib_data.incf`	`makepds.incf`
`combine_subarrays.incf`	`p_int.incf`
`z_int.incf`	`tautsp.incf`
`spline.incf`	

  Public procedures are
  `init_output, organize_input`

- `src_gscp.f90`
  Source module for computing grid scale precipitation.

  Description
  The module **`src_gscp`** calculates the rates of change of temperature, cloud water and water vapor due to cloud microphysical processes related to the formation of grid scale clouds and precipitation. The tendencies are added to the global tendency fields.

  The precipitation fluxes of rain and snow are calculated diagnostically by integrating the corresponding budget equations from the top of the atmosphere down to the surface. The resulting precipitation fluxes at the surface are stored on the corresponding global fields.

  The module contains as an organizational procedure the subroutine **`organize_gscp`** which is called from the main routine of the model. Depending on the value of the namelist-input parameter **`itype_gscp`**, various types of parameterizations may be called from "organize_gscp". At present, only two schemes are implemented:

  1. a warm rain Kessler parameterization (`itype_gscp=1`) and
  2. a scheme including snow (`itype_gscp=2`).

  Module procedures included

  `hydor  kessler`

  Public procedures are
  `hydor, kessler`

- `src_turbulence.f90`
  Source module for computing diffusion and transfer coefficients.

  Description
  The module `src_turbulence` calculates the diffusion coefficients for turbulent vertical transport of momentum (`tkvm`) and heat (`tkvh`). At the surface, the dimensionless transfer coefficients for momentum and heat are calculated.

  The module contains the subroutines partura (for the diffusion coefficients of the atmosphere) and parturs (for the transfer coefficients atmosphere-surface). In case of the two time level version of the model, the time tendencies due to turbulent diffusion are calculated in the subroutine `vertical_diffusion`.

  All routines are called from the main program of the model.

  Module procedures included

  `vertical_diffusion` `partura` `parturs`

  Public procedures are
  `vertical_diffusion, partura, parturs`

- `src_radiation.f90`
  Source module for computing the radiative transfer.

  Description
  The module `src_radiation` performs calculations related to the parameterization of radiative transfer. Driving routine is the module procedure `organize_radiation`, which calls the main routine `fesft` of the radiation package. Additionally, some diagnostics are done.

  All global variables of the model that are used by the interface routine `organize_radiation` are imported by `USE` statements. All parametric data that are required by the module procedures are defined in the data module `data_radiation`.

  Additionally, the routine `init_radiation` has to be called before the start of the forecast to initialize static data needed for the calculations. The routine `aerdis` is performed every time the radiation package is called, to determine some parameters for the vertical distribution of background aerosols which are not in the long term storage.

  The radiation package has been provided by B. Ritter in a plug-compatible Fortran 77 version. Some technical modifications have been done for the F90-Version:
  Internal communication by common-blocks is replaced by module parameters, scalars and arrays defined in module `data_radiation`.

  Module procedures included

`init_radiation.incf`	`organize_radiation.incf`
`fesft.incf`	`coe_so.incf`
`coe_th.incf`	`inv_so.incf`
`inv_th.incf`	`opt_so.incf`
`opt_th.incf`	`aerdis.incf`

  Public procedures are
  `init_radiation, organize_radiation`

- `src_soil.f90`
  Source module for computing the parametrizations of the soil processes.

  Description
  The module `src_soil` performs calculations related to the parameterization of soil processes. It contains the two soubroutines `terra1` and `terra2` which comprise the soil model of LM. All parametric scalar and array data for these soil model routines are defined in the data module `data_soil.f90`. All global variables of the model that are used by the soil model routines are imported by `USE` statements.

  The parameterization package has been provided by B. Ritter in a plug-compatible Fortran 77 version, which is based on the EM/DM soil model by E. Heise. Some technical modifications have been done for the F90 and the parallel Version:
  Internal communication by common-blocks is replaced by module parameters, scalars and arrays defined in module `data_soil.f90`. The plug compatible I/O lists of the subroutines have been replaced by the Module interface defined by `USE` lists.

  Module procedures included
  `terra1.incf` `terra2.incf`

  Public procedures are
  `terra1, terra2`

- `src_convection.f90`
  Source module for computing the parametrizations of the convection.

  Description
  The module `src_convection` performs calculations related to the para- meterization of subgrid-scale moist convection. The present parameteri- zation sheme is a mass-flux scheme according to (Tiedtke, 1989).

  The parameterization package has been provided by B. Ritter in a Plug-compatible Fortran77-Version, which is based on an earlier version by M. Tiedtke (ECMWF). Some modifications have been done for the F90 and the parallel Version.

  Module procedures included

`organize_convection`	`cu_tied`	`cu_asc`
`cu_ddraf`	`cu_dtdq`	`cu_dudv`

  Public procedures are
  `organize_convection`

- `src_dynamics.f90`
  Source module for computing the dynamical time stepping.

  Description
  The module `src_dynamics` performs one time step of the integration of the spatially disretized thermodynamical equations. Driving routine is the model procedure `organize_dynamics`, which calls the routines required. At the moment, the user can choose between two different time stepping schemes. The one used operationaly is the 3 time-level leapfrog scheme. The other one is a 2 time-level scheme. The 2 time-level scheme (by Wicker and Doms) still is in the process of testing and validation, so it should be used with care.

  Module procedures included

`init_dynamics.incf`	`organize_dynamics.incf`
`horizontal_advection.incf`	`horizontal_diffusion.incf`
`slow_tendencies.incf`	`fast_waves.incf`
`advection.incf`	`complete_tendencies.incf`
`add_tendencies.incf`	`update_moisture.incf`
`gauss.incf`	

  Public procedures are
  `init_dynamics, organize_dynamics`

- `src_relaxation.f90`
  Source module for boundary relaxation, Asselin filtering and saturation adjustment.

  Description
  The module `src_relaxation` performs the lateral boundary relaxation, the Asselin filtering for the 3 time-level leapfrog scheme and the saturation adjustment at the end of a large time step.

  Module procedures included

  `init_relaxation  sardass`

  Public procedures are
  `init_relaxation, sardass`

- `src_diagnosis.f90`
  Source module for diagnostic computations in LM subdomains.

  Description
  This module provides routines for diagnostic computations in subdomains of the LM. The subdomains are splitted according to the two dimensional domain decomposition. A subdomain that has a part of the diagnostic domain allocates space for storing diagnostic data for this domain. At the end of the program these data are gathered by PE 0, which performs additional calculations (meanvalues, global sums, etc.) and writes the data to a file YUPRDIAG.

  Module procedures included

`init_diagnosis`	`diagnosis`	`print_diagnosis`
`alloc_diagnosis`	`dealloc_diagnosis`	

  Public procedures are
  `init_diagnosis, diagnosis, print_diagnosis, dealloc_diagnosis`

- `src_differences.f90`
  Source module for computing differences to the driving model.

  Description
  This module provides routines to compute the differences of the prognostic variables of LM from the driving model (i.e. the boundary fields). Every subdomain allocates space for storing data (meanvalues, root mean square and maxima of the differences) during the forecast. At the end of the program these data are gathered by PE 0, which performs additional calculations (global meanvalues, maxima, etc.) and writes the data to a file YUPRDIFF.

  Module procedures included

`init_differences`	`alloc_differences`	`differences`
`evaluate_differences`	`print_differences`	`dealloc_differences`

  Public procedures are
  `init_differences`, `differences`, `print_differences`,
  `dealloc_differences`

- `src_meanvalues.f90`
  Source module for computations of meanvalues for printout.

  Description
  This module provides routines for computing and printing meanvalues and global maxima of special variables. These values act as a control to monitor the forecast. The data computed by all PEs are gathered and printed to a file YUPRMEAN by PE 0.

  Module procedures included

  `init_meanvalues   meanvalues`

  Public procedures are
  `init_meanvalues, meanvalues`

- `src_gridpoints.f90`
  Source module for gridpoint output.

  Description
  This module provides routines for the calculation of variables for gridpoint output. Every PE allocates space for the selected grid points of its subdomain. At the end of the program all data are gathered by PE 0 and written to a file YUPRGRPT.

  Module procedures included

`init_gridpoints`	`alloc_gridpoints`	`gridpoints`
`print_gplong`	`print_gpshort`	`dealloc_gridpoints`

  Public procedures are
  `init_gridpoints`, `gridpoints`, `print_gpshort`, `print_gplong`, `dealloc_gridpoints`

3.4 The Utility Modules of the LM

The utility modules contain service routines that can be used by the main program and the source modules. These routines are not connected to a model specific task. In the LM there are three different groups of utility modules:

- Common service routines:
  - `utilities.f90`
- Meteorological service routines:
  - `meteo_utilities.f90`
- Service routines for parallel programming:
  - `environment.f90`
  - `parallel_utilities.f90`
  - `io_utilities.f90`

Except the procedures for parallel programming, all routines are written in a plug-compatible manner. That means:

- No routine uses other modules, except the declarations for the `KIND`-type parameter; the data access is by parameter list only.
- No routine allocates dynamic memory; work space needed is provided via the parameter list.
- No derived data types are used.

- `utilities.f90`
  Source module for common utility routines.

  Description
  This module provides service utilities for the model. All routines are public.

  Module procedures included

  - `dfilt4`
    An arbitrary field is smoothed by applying a digital filter of length 4.
  - `dfilt8`
    An arbitrary field is smoothed by applying a digital filter of length 8.
  - `dolph`
    Calculates the Dolph-Chebyshev window for the initialization.
  - `elapsed_time`
    Returns the elapsed wall-clock time in seconds since the last call. On the first call the variables are only initialized. If no system clock is present, an error-value will be returned
  - `get_utc_date`
    Calculates the actual date using the date of the forecast-start and the number of timesteps performed.
  - `istringlen`
    Determines the length of a textstring without tailing blanks.
  - `phirot2phi`
    Converts phi from the rotated system to phi in the real geographical system.
  - `phi2phirot`
    Converts phi from the real geographical system to phi in the rotated system.
  - `rlarot2rla`
    Converts lambda from the rotated system to lambda in the real geographical system.
  - `rla2rlarot`
    Converts lambda from the real geographical system to lambda in the rotated system.
  - `smoother`
    Smoothes a 2-D field by applying digital filters.
  - `uvrot2uv`
    Converts the wind components u and v from the rotated system to the real geographical system.

- `uv2uvrot`
  Converts the wind components u and v from the real geographical system to the rotated system.
- `uv2df`
  Converts the wind components u and v to wind direction and speed.

• `meteo_utilities.f90`
Source module for meteorological utility routines.

Description
This module provides service utilities for meteorological calculations. All routines are public.

Module procedures included

- `calrho`
  Computes the air density for time-level `nx`.
- `calps`
  Computes the surface pressure p_s from the lowest full model level.
- `calpmsl`
  Computes the mean sea-level pressure.
- `calrelhum`
  Computes the relative humidity over water.
- `calomega`
  Computes the p-system vertical velocity (p-dot).
- `calprsum`
  Computes the total amount of precipitation.
- `caltopdc`
  Computes the top index of dry convection.
- `low_winds_temp`
  Computes winds and temperatures near the surface.
- `reference_atmosphere`
  Computes the reference atmosphere and vertical coordinate parameters.
- `satad`
  Corrects the temperature, the specific humidity and the cloud water content for condensation/evaporation.
- `tgcom`
  Computes the ground temperature.

- `environment.f90`
  Source module for the routines dealing with the parallel environment.

  Description
  This module provides routines that have to deal with defining the environment of the program. In case of running on parallel computers with distributed memory, the MPI library is used as message passing library. If running on a sequential machine without MPI, the environment is set accordingly and dummies are provided for linking the program.

  Module procedures included

`initialize_environment.incf`	`finalize_environment.incf`
`model_abort.incf`	`domain_decomposition.incf`
`check_decomposition.incf`	`exchange_boundaries.incf`
`putbuf.incf`	`getbuf.incf`
`comm_barrier.incf`	

- `parallel_utilities.f90`
  Utilitiy module for parallel programming (without I/O).

  Description
  This module provides routines that should enable programmers, not experienced in parallelization, to work with the LM. It contains routines that handle tasks as the distribution or collection of values.

  Module procedures included

`distribute_values.incf`	`remark.incf`
`collect_values.incf`	`collect_gpshort.incf`
`collect_gplong.incf`	`collect_timings.incf`
`grid_indices.incf`	`print_point.incf`
`print_column.incf`	`distribute_field.incf`
`gather_field.incf`	`gather_all.incf`

- `io_utilities.f90`
  Source Module providing utility routines for grib I/O.

Description
This module provides all routines that have to deal with input and output of grib files. Some of these routines use the message-passing library MPI. If possible the routines are written plug-compatible (with the exception of `setup_vartab.incf`).

Module procedures included

`setup_vartab.incf`	`create_filename.incf`
`open_grib.incf`	`close_grib.incf`
`read_grib.incf`	`write_grib.incf`
`check_grid.incf`	`check_data.incf`
`scatter_data.incf`	`collect_subarrays.incf`

3.5 The Data Modules of the LM

Data modules of the LM:

- `data_constants.f90`
  Data module for mathematical, physical and parametrizational constants.
- `data_diagnostics.f90`
  Data module for all data, that handle with the diagnostic routines.
- `data_fields.f90`
  Data module for all global meteorological fields.
- `data_filter.f90`
  Data module for variables used in the digital filtering.
- `data_io.f90`
  Data module for variables concerned with I/O.
- `data_modelconfig.f90`
  Data module for the configuration of spatial and time discretization.
- `data_nudge_all.f90`
  Data module for the variables used to compute the local information.
- `data_nudge_gather.f90`
  Data module for the variables used generally for the nudging.
- `data_nudge_local.f90`
  Data module for the variables only used to compute the local information.
- `data_nudge_spread.f90`
  Data module for variables only used for the spreading and nudging equations.
- `data_obs_process.f90`
  Data module for all data, that are used by the data assimilation.
- `data_obs_record.f90`
  Data module for the variables only used to compute the local information.
- `data_parallel.f90`
  Data module for the variables that organize the parallel program.
- `data_parameters.f90`
  Data module for global parameters.
- `data_radiation.f90`
  Data module for all data arrays, that are used by the radiation scheme.

- data_runcontrol.f90
  Data module for variables controlling the run of the model.

- data_soil.f90
  Data module for all parametric data in the soil model `soil`.

Description of the data modules:

- data_constants.f90
  This module contains mathematical and physical constants as well as constants for the parametrizations and related variables.

  - Mathematical constants.
  - Physical constants and related variables.
  - Constants for parametrizations.

- data_diagnostics.f90
  This module declares all diagnostic arrays, which are in the long term storage.

  - Arrays for grid point output
    - Arrays for the meteograms (short gridpoint output)
    - Arrays for long gridpoint output.
  - Arrays for the differences (LM-forecast – boundary fields)
    - Arrays for the total fields
    - For the prognostic variables, the same variables exist for an extra land- and sea-point diagnosis.
  - Arrays for the results of subroutine `diagnos`
  - Mean vertical movements in 3 levels
  - Variables for computing mean values that have to be saved
  - Variables for organizing external files for ASCII output of diagnostic computations

- data_fields.f90
  This module declares all meteorological fields that have to stay in the long term storage, i.e. that are used in more than one module. Fields included are

  - Constant fields for the reference atmosphere.
  - External parameter fields.
  - Prognostic variables.

  - Tendency fields for the prognostic variables.
  - Fields for surface values and soil model variables.
  - Fields that are computed in the parametrization and dynamics.
  - Fields for model output and diagnostics.
  - Fields for the boundary values.

- `data_filter.f90`
  This module declares data necessary for the digital filtering in the initialization step. These are scalar variables that are initialized by or derived from Namelist parameters and a three-dimensional field for every prognostic atmospheric variable to store the filtered fields.

- `data_io.f90`
  This module contains all data necessary for input and output of grib files.

  - Parameters and variables for the Grib code interface.
  - Derived data types for the LM variable table and the Namelist group `gribout`.
  - Additional control variables.

- `data_modelconfig.f90`
  This module contains all variables (scalars and arrays) concerned with the configuration of the horizontal and vertical grid, the time discretization and the reference atmosphere. These are

  - Vertical coordinate parameters and related variables.
  - Horizontal and vertical sizes of the fields and related variables.
  - Start- and end-indices for the computations in the horizontal layers.
  - Constants for the horizontal rotated grid and related variables.
  - Variables for the time discretization and related variables.

  The arrays are declared as allocatable arrays and are allocated in the setup and deallocated in the cleanup of the model. The scalar variables are either part of a NAMELIST (read in the setup) or are initialized in the routine `constants`.

- `data_nudge_all.f90`
  This module contains those variables (scalars and arrays) which must be generally available in the data assimilation (nudging) part of the code (i.e. which do not meet the restrictions imposed on the other data modules used for the data assimilation). These are

  - All NAMELIST variables controlling the data assimilation.

- Some parameters (constants) and associated variables (station ID).
- Tables for pressure dependent scales / geometry of horizontal correlations.
- I/O device numbers and file names for nudging.

All scalar variables and arrays including their size are either part of a `NAMELIST` (read in the setup) or are initialized here.

- `data_nudge_gather.f90`
  This module contains the variables (scalars and arrays) which are both concerned with the computation and broadcasting to other processors of all the required 'local' information (i.e. observation increments, and further parameters on the observations and their location) and the spreading of the observation increments. These are

  - Some commonly used parameters (constants) and general variables.
  - The arrays containing the 'local' information to be broadcast / gathered.
  - The format of the observation increment array for multi-level reports.
  - Arrays on local informations for printing for control.
  - Variables defining the size of the arrays.

  The arrays are declared as allocatable arrays and are allocated in procedure `local_info_aux` and deallocated in `gather_spread_aux`.

- `data_nudge_local.f90`
  This module contains those variables (scalars and arrays) which are merely concerned with the computation of all the required 'local' information (i.e. observation increments, and further parameters on the observations and their location) which will later be broadcasted to other processors for the spreading of the observational information. These are

  - Some parameters (constants).
  - The tables for pressure dependent quality control thresholds.
  - Some arrays related to the observation data records (ODR).
  - Indices and some other variables.

  The arrays are declared as allocatable arrays and are allocated in the setup and deallocated in the cleanup of the model. The scalar variables are either part of a NAMELIST (read in the setup) or are initialized in the routine `calculate_consts`.

- `data_nudge_spread.f90`
  This module contains those variables (scalars and arrays) which are merely concerned with the spreading of observation increments, computation of the analysis increments, and updating of the prognostic variables by nudging, i.e. after the processor has received all the required 'local' information from all observations. These are

  - The analysis increment fields.
  - The output of spreading procedures (sum of weights and weighted increments).
  - Further fields used to compute analysis increments.
  - Geometrical fields used for horizontal distances and wind correlations.
  - The horizontal fields used for the spreading of the observation increments.
  - Other fields, variables, and indices used for the spreading.

  The arrays are declared as allocatable arrays and are allocated after the gathering of the 'local information' and deallocated after updating the prognostic fields due to the nudging, except for some analysis increment fields (section 1), which are allocated in the setup and deallocated in the cleanup of the model.

  The scalar variables are initialized after the gathering of the 'local information' or during the spreading.

- `data_obs_process.f90`
  This module declares and initializes all parametric data and data arrays that are used by the data assimilation scheme (nudging and surface parameter analysis). The variables are divided into several groups:

  - Analysis observation file formats.
  - Numerical and scaling constants.
  - Variables used for the production of aircraft multi-level reports.
  - Data event counter arrays and diagnostics arrays and formats.
  - Observation errors.
  - Observation and variable code type numbers.
  - Limit values.
  - Further parameters controlling the observation processing.
  - Output buffer and formats.

- `data_obs_record.f90`
  This module contains those variables (scalars and arrays) which are computed in the observation processing and used for the computation of the

required 'local' information (i.e. observation increments, and further parameters on the observations) needed for the spreading of the observational information. These are

- The format of the observation data records (ODR) and related variables.
- The ODR themselves, and related arrays
- The masking constants for packed information

- `data_parallel.f90`
  This module contains all variables that are necessary for the organization of the parallel program (number of processors, virtual topology, neighbors, etc.). In a sequential environment some of these variables are also necessary and are set accordingly (e.g. nproc = 1 and my_world_id = 0). There are several groups:

  - Information about the processor grid.
  - Further information for MPI.
  - Static send buffers.

- `data_parameters.f90`
  This module contains global parameters for defining the KIND types of real- and integer-variables of the program.

- `data_radiation.f90`
  This module declares and initializes all parametric data and data arrays that are used by the radiation scheme (i.e. the `fesft`-routine). This data module replaces the COMMON blocks and local data arrays that are used in the original Fortran77 version of the `fesft`-code provided by Bodo Ritter.

  - Absorption properties of atmospheric gases (1=H_2O; 2=CO_2; 3=O_3).
  - Limits of spectral intervals in the radiation code (for information only).
  - Rayleigh scattering coefficients in solar spectral intervals.
  - Fraction of solar energy at TOA contained in individual solar spectral intervals.
  - Coefficients for black body radiation and E-type coefficients.
  - Aerosol optical properties for 8 spectral intervals.
  - Optical properties of liquid water for all 8 spectral intervals (solar and thermal).
  - Optical properties of ice clouds for all 8 spectral intervals (solar and thermal).

- `data_runcontrol.f90`
  This module contains variables for running and controlling the forecast. Concerned are the organization of the forecast, the grib I/O, grid point calculations and diagnostic calculations. The variables are divided into the following groups:

  - Start and end of the forecast.
  - Boundary definition and update.
  - Controlling the physics.
  - Controlling the dynamics.
  - Controlling the nudging.
  - Controlling the upper boundary condition.
  - Additional control variables.
  - Controlling the grib-input.
  - Grid point calculations and output.
  - Diagnostic calculations and output.
  - Calculations and output for the difference fields (LM - boundary).
  - Calculations and output of mean values and maxima for the total domain.
  - Time measuring.

- `data_soil.f90`
  This module declares and initializes all parametric scalar and array data which are used in the soil model (terra1 and terra2).

  - Data arrays for properties of different soil types (array index).
  - Additional parameters for the soil model.

The relations between the different modules are shown in Figure 3.1.

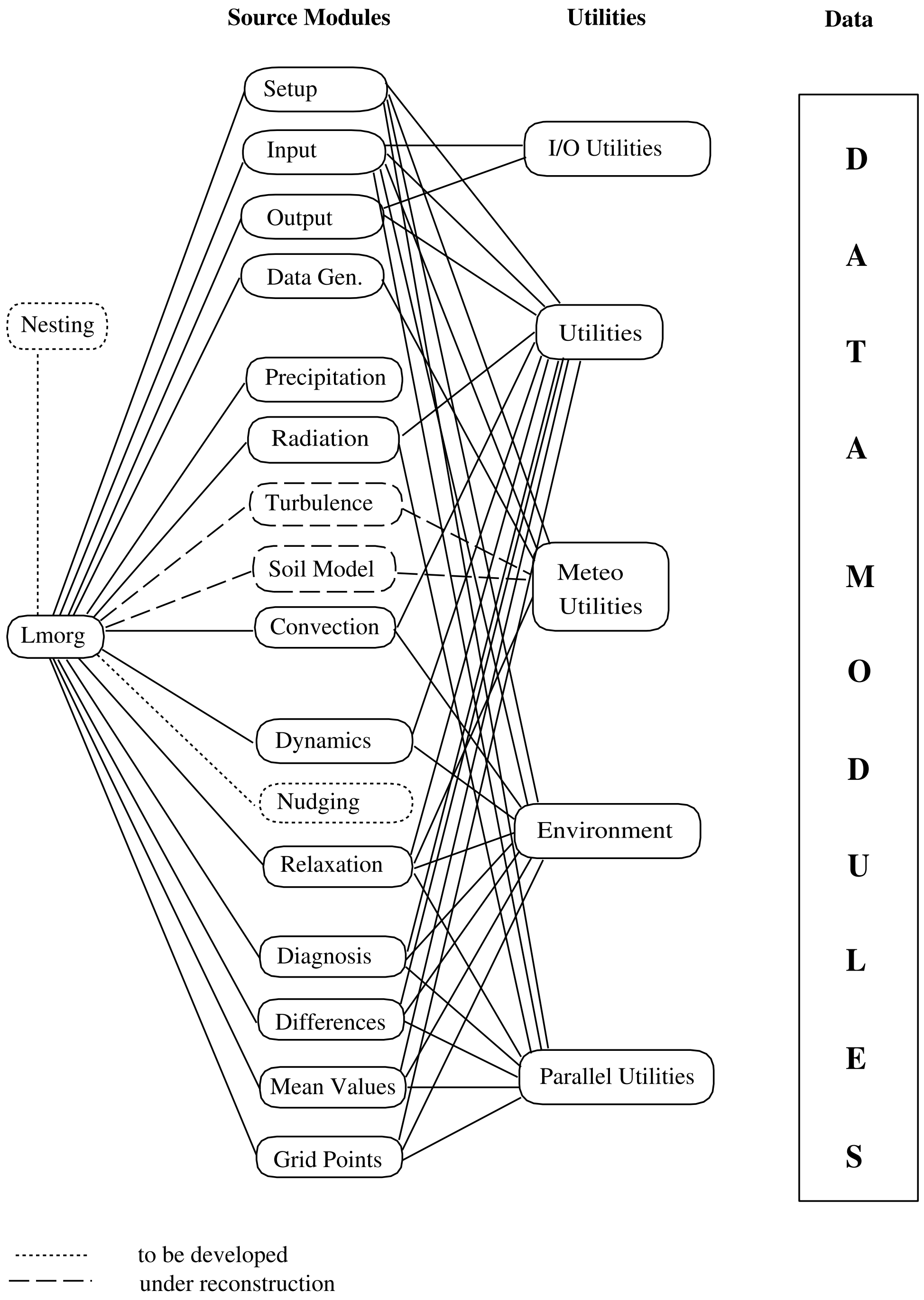

Figure 3.1: Structure of the LM

Chapter 4

Description of the Source Modules

4.1 Setup of the Model

4.2 Input / Output

4.3 Dynamics

4.4 Grid Scale Precipitation

4.5 Radiation

4.6 Turbulence

4.7 Soil Model

4.8 Convection

4.9 Diagnostic Computations

Chapter 5

Description of the Utility Modules

5.1 Utilities

5.2 Meteorological Utilities

5.3 Environment of the Model – Parallelization

5.4 Parallel Utilities

5.5 I/O Utilities

Chapter 6

Subroutine Glossary

Bibliography

??? (1997). Source code access and restrictions. Technical report, Deutscher Wetterdienst, Postfach 100465, 63004 Offenbach, Germany.

Andrews Ph., e. a. (1995). *European Standards for Writing and Documenting Exchangeable Fortran 90 Code.* UKMO.

Kalnay E., e. a. (1989). Rules for interchange of physical parameterizations. *Bull. A.M.S.*, 70, No. 6:620 – 622.

Tiedtke, M. (1989). A comprehensive mass flux scheme for cumulus parameterization in large-scale models. *Mon. Wea. Rev.*, 117:1779 – 1799.