Using GRIB2 in the COSMO-Model System

Ulrich Schättler

Source Code Administrator

COSMO-Model

Contents

- ➔ Structure of GRIB2
- ➔ Usage of grib_api
- ➔ Product Identifying Keys
- ➔ General Vertical Coordinate

Usage of GRIB_API (Application Programmers Interface)

- ECMWF source code for de-/encoding of GRIB1 AND GRIB2
- no internal knowledge of GRIB structure needed
- each element of a grib message has an alphanumeric name (key) that can be used to access the information linked to it (value)
- key – value approach: **shortName=T**

  typeOfLevel = hybridLayer

  topLevel = 37

  bottomLevel = 38
- How to find keys: `grib_keys -F file.grib` lists all keys of a file
- flexible – local definitions for each centre possible
  - i.e. local definition tables shortName.def for edzw (GRIB1 and GRIB2),
  - local definition of GME icosahedral grid for GRIB1
- But: needs more coordination within COSMO: for example how to handle the product identifying keys

Product Identifying Keys

- significanceOfReferenceTime (GRIB2-WMO Tab. 1.2)
- productionStatusOfProcessedData (GRIB2-WMO Tab. 1.3)
- typeOfProcessedData (GRIB2-WMO Tab. 1.4)
- generatingProcessIdentifier (local definition in GRIB1/2)
- backgroundGeneratingProcessIdentifier (local definition in GRIB2)
- typeOfGeneratingProcess (GRIB2-WMO Tab. 4.3)
- localDefinitionNumber (local section = 254, 253, 252; 250 for COSMO)
  - localNumberOfExperiment (GRIB2)
  - localInformationNumber (GRIB2) / localElementNumber (GRIB1)
  - localDecodeDate:s

Product Identifying Keys (II)

Key	Values
significanceOfReferenceTime	0 analysis 1 start of forecast 2 verifying time of forecast 3 observation time
productionStatusOf ProcessedData (possible local use)	0 operational – Routine 1 operational test – Parallelsuite 2 research – Experiments 3 re-analysis products
typeOfProcessedData	0 analysis 1 forecast 2 analysis and forecast 3 control forecast 4 perturbed forecast 5 control and perturbed forecast

Product Identifying Keys (III)

Key	Values
typeOfGeneratingProcess (local use)	0 analysis 1 initialization 2 forecast 195 interpolated analysis / forecast
backgroundGeneratingProcessIdentifier (local use)	0 main run 1 pre-assimilation 2 assimilation 3 test
generatingProcessIdentifier (local use)	Represents data base identifier

Local Use Section (localDefinitionNumber=254)

- localDefinitionNumber Identifier for content (historical: 254)
- localHostIdentifier
- localCreationDateYear/Month/Day/Hour/Minute/Second
- localValidityDateYear/Month/Day/Hour/Minute/Second
- localNumberOfExperiment Number of Experiment
- localInformationNumber
- Identifier for host system/ computer

Also need a local use section for COSMO (localDefinitionNumber = 250)

Local Use for generatingProcessIdentifier

In io_metadata.f90, Subroutine: make_grib_init

`izgeneprocid`: variable for generatingProcessIdentifier

```
SELECT CASE (ncenter)
CASE (78)    ! DWD
   izgeneprocid = Function(analyis, forecast)
CASE DEFAULT
   izgeneprocid = 255   ! not defined
END SELECT

CALL grib_set (.,generatingProcessIdentifier, izgeneprocid)
```

Local Use for productionStatusOfProcessedData

In io_metadata.f90, Subroutine: make_grib_init

```
SELECT CASE (ncenter)
CASE (78)   ! DWD
   compute nzstatus, izmodnvers as function of nvers (Namelist variable)
   CALL grib_set (.,productionStatusOfProcessedData, nzstatus)
   CALL grib_set (.,localNumberOfExperiment, izmodnvers)
CASE DEFAULT
  IF(lroutine) THEN
   CALL grib_set (.,productionStatusOfProcessedData, 0)    ! Operational
  ELSE
   CALL grib_set (.,productionStatusOfProcessedData, 2)    ! Experimental
  ENDIF
  CALL grib_set (.,localNumberOfExperiment, nvers)
END SELECT
```

Local Use for typeOfGeneratingProcess

In io_metadata.f90, Subroutine: make_grib_init

```
SELECT CASE (ncenter)
CASE (78)   ! DWD
 IF (leps) THEN
  CALL grib_set (izgrbid,'typeOfGeneratingProcess',  4) ! Ensemble Forecast
 ELSE
   IF (ptr_to_out%lanalysis) THEN
     CALL grib_set (izgrbid,'typeOfGeneratingProcess', 202) ! Nudging
   ELSEIF (ptr_to_out%lsfc_ana) THEN
     CALL grib_set (izgrbid,'typeOfGeneratingProcess',   0) ! External Ana.
  etc.
CASE DEFAULT
  only default settings are used
END SELECT
```

Local Use

- ➔ We want to document local use on the Web Page!
  - ➔ yes, I know: I said that also last year
- ➔ But we need to know about your settings of special local keys
- ➔ We also want to document the local definition tables: shortname.def, etc.

General Vertical Coordinate

typeOfLevel = 150

Why a new vertical coordinate?

- As a non-hydrostatic model, COSMO needs a special vertical grid: fixed in space and time
- Also post-processing programs have to be aware of this grid (or the HHL)
- But the algorithm to compute it, is rather complex (not just $a_k + b_k * p_s$)

- Therefore a proposition was made to WMO, to introduce a new typeOfLevel=150
- To process atmospheric data using that typeOfLevel, another 3D field is necessary: the HHL fields
- If a product has typeOfLevel=150, then there are 6 additional meta data in the Product Definition Section, which replace the vertical coordinate parameters

numberOfVGridUsed	to identify a special vertical coordinate (`ivctype`)
nlev	number of levels of the HHL file
uuidOfVGrid:	unique universal identifier to ensure correct identification of HHL

Current Situation and New Solution

- ➔ INT2LM and the COSMO-Model both compute the HHL fields and the reference atmosphere p_0.
- ➔ The necessary vertical coordinate parameters (for HHL) and the reference atmosphere parameters are given to
  - ➔ INT2LM by Namelist variables
  - ➔ COSMO-Model by GRIB1 (or NetCDF) meta data: but this always was a non-standard GRIB usage!
- ➔ New Solution:
  - ➔ The new generalized vertical coordinate does not know meta data for vertical coordinate parameters and for the reference atmosphere.
  - ➔ HHL and full pressure P are transferred from INT2LM to COSMO and within the assimilation cycle from COSMO to COSMO by the initial laf-file, but with a higher precision (24 bits packing rate).

Consequences

- Reference atmosphere parameters are no more available
  - `irefatm, p0sl, t0sl, dt0lp, delta_t, h_scal`
  - cannot compute the reference pressure p_0
- Solution:
  - New namelist variables in the COSMO-Model for the reference atmosphere parameters (in group `/LMGRID/`)
  - which reference atmosphere is used does not depend on the reference atmosphere used in INT2LM!
  - COSMO-Model can still compute the reference pressure p_0

Consequences (II)

- Vertical coordinate parameters are no more available
  - `ivctype, vcoord, vcflat, svc1, scv2, nfltvc`
  - cannot compute the height of half levels `HHL`
- Solution:
  - Transfer `HHL` within the initial laf-file
  - But do we need the vertical coordinate parameters for other purposes?
  - COSMO-Model computes two kind of vertical coordinate parameters out of `vcoord`
    - `vcoord%vert_coord`: height of levels above mean sea level
    - `vcoord%sigm_coord`: reference pressure above mean sea level (normalized to [0,1])
  - Both variants are used in the COSMO-Model

Vertical Coordinate Parameters: Special Use

- Get the model level, which is about 8000 m above surface (in fact: the model level 8000 m above mean sea level is taken, because `vcoord%vert_coord` is used). This level is used for all grid points (e.g. to compute the snow fall limit). But what about the Himalaya?
- Now you could search in every column for the level, which is 8000 m above surface. Would be the correct solution, but more complicated to program.
- Alternative: After reading the initial file, the COSMO-Model stores a „reference profile“ in a special 1D variable: `hhl_prof(0:ke+1)`
  - The lowest grid point above mean sea level is taken for that reference profile
  - If there is a sea-point in the model domain, `hhl_prof` just contains the `vcoord%vert_coord` parameters (height of half levels above mean sea level)
  - This reference profile could be taken for the task above

Vertical Coordinate Parameters: Still used

- ➔ But the vertical coordinate parameters `vcoord%vert_coord` are still used in the nudging (latent heat nudging and nudging)
- ➔ And also the pressure coordinates `vcoord%sigm_coord` are still used
  - ➔ nudging, convection, radiation, stochastic physics (also spectral nudging)

- ➔ The next trick: GRIB2 knows the „`firstFixedSurface`“ and also the „`secondFixedSurface`“ (specify „`first`“ for levels, and both for layers)
  - ➔ For `HHL`, we only have to specify the „`firstFixedSurface`“ with the number of the level `k`
  - ➔ We can use the „`secondFixedSurface`“ to specify the height of this level above mean sea level (the `vcoord%vert_coord(k)` for level `k`)

- ➔ But still it would be good to check, whether these values are really needed!

Vertical Coordinate Parameters: One more Problem

- Still used in the nudging: `vcoord%vcflat`
  - Height, where levels become flat
  - Given a vertical reference profile for some grid point, we can compute `vcflat` for `ivctype=2`, because of the easy formula

$$hhl_{ijz} = a(z) + b(z) \cdot hsurf_{ij} = z + \frac{vcflat - z}{vcflat} \cdot hsurf_{ij}$$

  - `vcflat` is then stored in `hhl_prof(0)`
  - This is NOT the case for the SLEVE coordinates (`ivctype=3/4`). Here we can only give an estimation (calculate it as for `ivctype=2`)

How to run a Forecast using GRIB2

without data assimilation

- ➔ INT2LM computes the initial and boundary data
  - ➔ provide namelist input for reference atmosphere and vertical coordinate parameters (as usual)
  - ➔ set namelist variable `lnewVGrid=.TRUE.` (in group `/LMGRID/`): then `HHL` for the fine COSMO grid is computed and a new UUID is set)
  - ➔ if it is a COSMO(coarse) ⇒ COSMO(fine) interpolation, and the coarse data are also in GRIB2, INT2LM has to read `HHL` fields for the coarse grid.
    - ➔ specify a file containing `HHL` with `yin_hhl='lfff00000000c'`
    - ➔ this file has to be in directory `yinext_cat`
  - ➔ HHL and all atmospheric variables are written with `typeOfLevel=150` (generalVertical) and contain the new UUID

How to run a Forecast using GRIB2 (II)

without data assimilation

- ➔ The COSMO-Model reads fields
  - ➔ provide namelist input for the reference atmosphere parameters in group (`/LMGRID/`)
  - ➔ it reads the `HHL` fields from the laf-file and checks that all atmospheric fields have the same UUID
  - ➔ it computes a reference atmosphere using the reference atmosphere parameters specified, but does not compute `HHL`
  - ➔ it reads the boundary fields and checks that all atmospheric fields have the same UUID

How to run a Forecast using GRIB2 (III)

with data assimilation

- The initial atmospheric data are provided by a COSMO run using nudging and INT2LM only computes boundary data. But COSMO (nudging run) and INT2LM have to use the same `HHL` with the same UUID.
- For INT2LM
  - provide namelist input for reference atmosphere and vertical coordinate parameters (as usual)
  - set namelist variable `lnewVGrid=.FALSE.` (in group `/LMGRID/`) and specify a `HHL` file for the fine COSMO grid:
    - `ylm_hhl = 'COSMO_HHL_name.g2'` (has to be in the directory `ylmext_cat`)
    - then `HHL` for the fine COSMO grid is not computed but read from the specified file and the UUID from the fields read are taken.
  - Rest: is the same as before (including COSMO-Model forecast run)

How to run a Forecast using GRIB2: Summary

- ➔ with `lnewVGRID=.TRUE.`: you can run your forecast system nearly the way, as it was with GRIB1 (but the yin_hhl file for coarse COSMO grid in INT2LM)
- ➔ with `lnewVGRID=.FALSE.`:
  - ➔ can also be chosen for forecasts without data assimilation
  - ➔ we recommend to produce an additional file `COSMO_HHL_with_a_name`, that can be used by INT2LM daily (until you change the horizontal and / or the vertical grid.

Thank you

very much

for your

attention