Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

COSMO software

fieldextra

Jean-Marie Bettems / MeteoSwiss

Jerusalem, COSMO GM, September 2017

Fieldextra

- **Core development** team

  Petra Baumann, ***Jean-Marie Bettems (SCA)***

- With **contributions** from

  Felix Ament, Axel Barleben / DWD, Philipp Glatt, Christoph Hug, Pirmin Kaufmann (original code), Guy de Morsier, Donat Perler, Florian Prill / DWD, Anne Roches, Vanessa Fundel, Martin Schraner, Balazs Szintai, André Walser

Fieldextra

- **Generic tool** to process model data and gridded observations
  - Provide a set of primitive operations, which can be combined (**toolbox**)
  - Single **Fortran 2008 program** controlled by **namelists**
  - **Read once** & **write multiple**

- Primary focus is the **production environment**
  - **Robust** (no crash, recovery after exception, scale with problem size)
  - **Safe** (sytematic check of operations and products consistency)
  - **Efficient** (time to solution, memory footprint, input/output)
  - **Versatile** (rich toolbox, code extensibility)

- Fully **documented**, incl. many examples, but…
  - **… which use cases are possible / meaningful ?**
  - **... hard to create namelists for complex problems**

Fieldextra

Case study:

Pollen downscaling from COSMO-7 to COSMO-1 (→ lateral BC)

```
&Process
  in_file="./input/lm_coarse/lffff00000000"
  out_regrid_target = "hsurf_c1"
  out_regrid_method = "linear_fit,square,4.5"
  out_file="./tmp_output/downscaling_c7_2_c1_0000.g1", out_type="GRIB1",
/
&Process in_field="AMBR", tag="ambr_c7" /
...

&Process tmp2_field="ambr_c7", tag="ambr_c1",
        voper = "intpl_k2z++,  hfl_c1,lnp;  extpl_down,constant,hfl_c7_ke,ambr_c7_ke"
        voper_use_tag = "p_c7,hfl_c7,hfl_c1,hfl_c7_ke,qv_c7_ke"                    /
```

- *More complex interpolation operators can be used / implemented*
- *About 2x faster than INT2LM on a single Kesch node (for this specific application)*

Fieldextra

Recent releases

- **v12.3.1** (12.08.2016)
- **v12.4.0** (02.12.2016 – Common COSMO GRIB API environment)
- **v12.5.0** (21.02.2017)
- **v12.6.0** (25.07.2017)

→ 2-3 major releases each year

Fieldextra

Availability

- **Full code and documentation** on GitHub
  https://github.com/MeteoSwiss-APN/fieldextra
- **Standalone package** on COSMO web site (main releases only!)
  http://www.cosmo-model.org/content/support/software/default.htm
- **Full installation** at ECMWF on cca (special UNIX group cfxtra)
  /perm/ms/ch/ch7/projects/fieldextra

Fieldextra

About backward compatibility

- **Backward compatibility not guaranteed**
  (i.e. old namelist not always compatible with newest code)
- But… all required namelist modifications documented in COMPATIBILITY_ISSUES_* files

→ It is advised to regularly update the software!

Fieldextra

Some highlights since COSMO GM 2016

- Full implementation of **COSMO GRIB API policy**
  (vendor & COSMO, as external resources)
- Update to **GRIB API 1.20.0**, with adaptive entropy coding support
  (C1 output reduced by a factor 2.5, processing cost increased by a factor 1.5)
- **ASCII input**
  (easy creation of GRIB or NetCDF files, incl. meta-information)
- Support missing input in time series, interpolation in **time dimension**
- **Consolidation** of functionality and of code
  (GRIB1, GRIB2… thermodynamic operators… time operators…)

- **See** *https://github.com/MeteoSwiss-APN/fieldextra-wiki/wiki/History*
- ***See*** *https://github.com/MeteoSwiss-APN/fieldextra/blob/master/admin/HISTORY*

Fieldextra

About code performance… *another motivation for upgrading!*

- Systematic **performance monitoring** during code development, e.g.
  - '*benchmark*' mix of standard MCH products
  - '*fabec*' interpolation on standard levels for flight control
  - '*parallel prod*' mix of differential operators & convection indices
- Incremental **improvement** of performances observed with new releases, both sequential and parallel

	Benchmark [s] (3x3)	Fabec [s] (1x4)	Parallel prod [s] (3x4) (2x12)
v12.1.0 (12.2015)	39	61	39
v12.3.0 (06.2016)	34	59	25
v12.7.0 (08.2017)	29 **(-25%)**	52 **(-15%)**	24 **(-38%)** 17 **(-56%)**

Fieldextra

Towards v13.0.0 (→ Q1 2018)

features

- Add **vertical cross section** along arbitrary path (generate NetCDF output)
- Add **NetCDF** import (for COSMO type output)

environment

- Provide more **commented examples** (→ user friendliness)
- Improve **regression tests & automatic build** (→ code quality)
- Examine **software life cycle** & further planning (→ sustainability)
  - *Many possible development lines*
    (optimization, code consolidation / clean-up, simplify usage…)
  - *Organize workshop with stakeholders?*

- **See** *https://github.com/MeteoSwiss-APN/fieldextra-wiki/wiki/Planning*
- ***See*** *https://github.com/MeteoSwiss-APN/fieldextra/milestones*

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

GRIB support:
GRIB API

Jean-Marie Bettems / MeteoSwiss

Jerusalem, COSMO GM, September 2017

GRIB API

GRIB API is a software developed at ECMWF to process GRIB records.

GRIB API is composed of the encoding / decoding engine, and of one or more sets of decoding rules (*definition files*).

The **Common COSMO GRIB 2 Policy** coordinates the usage of the GRIB API in the COSMO consortium (short names, model names, local codes…).

Besides the ECMWF distribution, a set of **COSMO definition files** is maintained to support the processing of GRIB 2 records by the different COSMO software.

GRIB API is now replaced by **eccode**, which should be backward compatible.

→ **Migration plan will follow DWD planning**

GRIB API

- **Content of latest set of COSMO definition files v1.20.0.2**
  - latest set of definition files prepared at DWD (status June 2017)
  - proposed modifications for new COSMO output (as communicated to the TAG by Uli S.)
  - pollen information
  - definitions for VGUST_DYN_10M, VGUST_CON_10M, VABSMX_10M, RCLD
  - support of COSMO-LEPS local coding
  - See https://github.com/C2SM-RCM/libgrib-api-cosmo-resources

- **Poor *OpenMP* performances of GRIB API 1.20.0**
  - already bad in1.13.1 but worse now, resulting in poor performances of parallel GRIB 2 write
  - *software support ticket SUP-2089 has been opened at ECMWF*

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

Sharing COSMO software

GitHub

Jean-Marie Bettems / MeteoSwiss

Jerusalem, COSMO GM, September 2017

GitHub

GitHub is a **web based service** for source code management, based on git

Git is a distributed **version control system** (think SVN…)

GitHub offers …

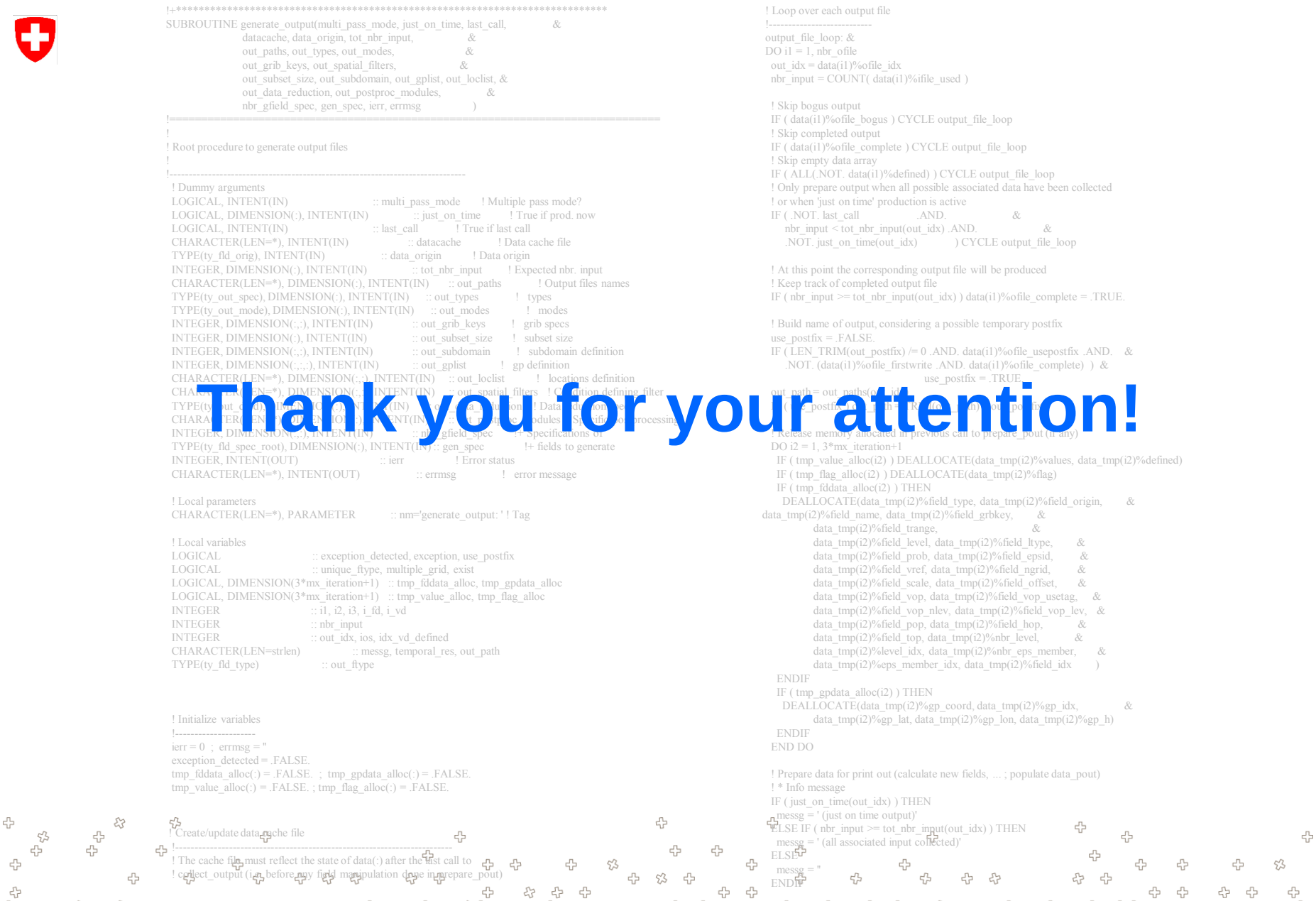
- **Easy to use information platform**
  markup language, integrated WiKi and PDF reader, commits history…
- Tools for **sharing code development in a distributed environment**
  discussion board, code review …
- Integration with **automatic testing procedure** (Jenkins)
- High level of **security**
  https://help.github.com/articles/github-security/

- Source code is managed with **Git**, which is **open source**, and not dependent on GitHub… so, no risk to loose your work

COSMO software on GitHub

- **libgrib-api-cosmo-resources** (C2SM-RCM org)
  - *common COSMO definition files for GRIB API*
- **EXTPAR** (C2SM-RCM org)
  - *COSMO software for generation of external parameters*
- **Fieldextra** (MeteoSwiss-APN org)
  - *COSMO postprocessing software*
- **COSMO-POMPA** (MeteoSwiss-APN org)
  - *most recent GPU version of the COSMO code*
- **COSMO-PRERELASE** (MeteoSwiss-APN org)
  - *most recent **development** branch of the COSMO code*
  - *essential for **coordinating** multiple branches of development*

Private GitHub access provided by C2SM / ETHZ



```
!+*****************************************************************************
SUBROUTINE generate_output(multi_pass_mode, just_on_time, last_call,              &
                           datacache, data_origin, tot_nbr_input,                 &
                           out_paths, out_types, out_modes,                       &
                           out_grib_keys, out_spatial_filters,                    &
                           out_subset_size, out_subdomain, out_gplist, out_loclist, &
                           out_data_reduction, out_postproc_modules,              &
                           nbr_gfield_spec, gen_spec, ierr, errmsg                )
!=============================================================================
!
! Root procedure to generate output files
!
!-----------------------------------------------------------------------------
 ! Dummy arguments
 LOGICAL, INTENT(IN)                          :: multi_pass_mode       ! Multiple pass mode?
 LOGICAL, DIMENSION(:), INTENT(IN)            :: just_on_time          ! True if prod. now
 LOGICAL, INTENT(IN)                          :: last_call             ! True if last call
 CHARACTER(LEN=*), INTENT(IN)                 :: datacache             ! Data cache file
 TYPE(ty_fld_orig), INTENT(IN)                :: data_origin           ! Data origin
 INTEGER, DIMENSION(:), INTENT(IN)            :: tot_nbr_input         ! Expected nbr. input
 CHARACTER(LEN=*), DIMENSION(:), INTENT(IN)   :: out_paths             ! Output files names
 TYPE(ty_out_spec), DIMENSION(:), INTENT(IN)  :: out_types             !   types
 TYPE(ty_out_mode), DIMENSION(:), INTENT(IN)  :: out_modes             !   modes
 INTEGER, DIMENSION(:,:), INTENT(IN)          :: out_grib_keys         !   grib specs
 INTEGER, DIMENSION(:), INTENT(IN)            :: out_subset_size       !   subset size
 INTEGER, DIMENSION(:,:), INTENT(IN)          :: out_subdomain         !   subdomain definition
 INTEGER, DIMENSION(:,:,:), INTENT(IN)        :: out_gplist            !   gp definition
 CHARACTER(LEN=*), DIMENSION(:), INTENT(IN)   :: out_loclist           !   locations definition
 CHARACTER(LEN=*), DIMENSION(:), INTENT(IN)   :: out_spatial_filters   ! Definition defining filter
 TYPE(ty_out_data_reduction), DIMENSION(:), INTENT(IN) :: out_data_reduction ! Data reduction to be
 CHARACTER(LEN=*), DIMENSION(:), INTENT(IN)   :: out_postproc_modules  ! Modules for post processing
 INTEGER, DIMENSION(:,:), INTENT(IN)          :: nbr_gfield_spec       !+ Specifications of
 TYPE(ty_fld_spec_root), DIMENSION(:), INTENT(IN) :: gen_spec          !+ fields to generate
 INTEGER, INTENT(OUT)                         :: ierr                  ! Error status
 CHARACTER(LEN=*), INTENT(OUT)                :: errmsg                !   error message

 ! Local parameters
 CHARACTER(LEN=*), PARAMETER                  :: nm='generate_output: ' ! Tag

 ! Local variables
 LOGICAL                                      :: exception_detected, exception, use_postfix
 LOGICAL                                      :: unique_ftype, multiple_grid, exist
 LOGICAL, DIMENSION(3*mx_iteration+1)         :: tmp_fddata_alloc, tmp_gpdata_alloc
 LOGICAL, DIMENSION(3*mx_iteration+1)         :: tmp_value_alloc, tmp_flag_alloc
 INTEGER                                      :: i1, i2, i3, i_fd, i_vd
 INTEGER                                      :: nbr_input
 INTEGER                                      :: out_idx, ios, idx_vd_defined
 CHARACTER(LEN=strlen)                        :: messg, temporal_res, out_path
 TYPE(ty_fld_type)                            :: out_ftype



 ! Initialize variables
 !---------------------
 ierr = 0  ;  errmsg = ''
 exception_detected = .FALSE.
 tmp_fddata_alloc(:) = .FALSE.  ;  tmp_gpdata_alloc(:) = .FALSE.
 tmp_value_alloc(:) = .FALSE. ; tmp_flag_alloc(:) = .FALSE.


 ! Create/update data cache file
 !------------------------------
 ! The cache file must reflect the state of data(:) after the last call to
 ! collect_output (i.e. before any field manipulation done in prepare_pout)
                                 _
```

```
 ! Loop over each output file
 !---------------------------
 output_file_loop: &
 DO i1 = 1, nbr_ofile
  out_idx = data(i1)%ofile_idx
  nbr_input = COUNT( data(i1)%ifile_used )

  ! Skip bogus output
  IF ( data(i1)%ofile_bogus ) CYCLE output_file_loop
  ! Skip completed output
  IF ( data(i1)%ofile_complete ) CYCLE output_file_loop
  ! Skip empty data array
  IF ( ALL(.NOT. data(i1)%defined) ) CYCLE output_file_loop
  ! Only prepare output when all possible associated data have been collected
  ! or when 'just on time' production is active
  IF ( .NOT. last_call                    .AND.                    &
      nbr_input < tot_nbr_input(out_idx) .AND.                    &
      .NOT. just_on_time(out_idx)           ) CYCLE output_file_loop

  ! At this point the corresponding output file will be produced
  ! Keep track of completed output file
  IF ( nbr_input >= tot_nbr_input(out_idx) ) data(i1)%ofile_complete = .TRUE.

  ! Build name of output, considering a possible temporary postfix
  use_postfix = .FALSE.
  IF ( LEN_TRIM(out_postfix) /= 0 .AND. data(i1)%ofile_usepostfix .AND.   &
      .NOT. (data(i1)%ofile_firstwrite .AND. data(i1)%ofile_complete)  ) &
                                                  use_postfix = .TRUE.
  out_path = out_paths(out_idx)
  IF ( use_postfix ) out_path = TRIM(out_path)//TRIM(out_postfix)
  ! Release memory allocated in previous call to prepare_pout (if any)
  DO i2 = 1, 3*mx_iteration+1
   IF ( tmp_value_alloc(i2) ) DEALLOCATE(data_tmp(i2)%values, data_tmp(i2)%defined)
   IF ( tmp_flag_alloc(i2)  ) DEALLOCATE(data_tmp(i2)%flag)
   IF ( tmp_fddata_alloc(i2) ) THEN
    DEALLOCATE(data_tmp(i2)%field_type, data_tmp(i2)%field_origin,       &
data_tmp(i2)%field_name, data_tmp(i2)%field_grbkey,       &
               data_tmp(i2)%field_trange,                              &
               data_tmp(i2)%field_level, data_tmp(i2)%field_ltype,     &
               data_tmp(i2)%field_prob, data_tmp(i2)%field_epsid,      &
               data_tmp(i2)%field_vref, data_tmp(i2)%field_ngrid,      &
               data_tmp(i2)%field_scale, data_tmp(i2)%field_offset,    &
               data_tmp(i2)%field_vop, data_tmp(i2)%field_vop_usetag,  &
               data_tmp(i2)%field_vop_nlev, data_tmp(i2)%field_vop_lev, &
               data_tmp(i2)%field_pop, data_tmp(i2)%field_hop,         &
               data_tmp(i2)%field_top, data_tmp(i2)%nbr_level,         &
               data_tmp(i2)%level_idx, data_tmp(i2)%nbr_eps_member,    &
               data_tmp(i2)%eps_member_idx, data_tmp(i2)%field_idx      )
   ENDIF
   IF ( tmp_gpdata_alloc(i2) ) THEN
    DEALLOCATE(data_tmp(i2)%gp_coord, data_tmp(i2)%gp_idx,              &
               data_tmp(i2)%gp_lat, data_tmp(i2)%gp_lon, data_tmp(i2)%gp_h)
   ENDIF
  END DO

  ! Prepare data for print out (calculate new fields, ... ; populate data_pout)
  ! * Info message
  IF ( just_on_time(out_idx) ) THEN
   messg = ' (just on time output)'
  ELSE IF ( nbr_input >= tot_nbr_input(out_idx) ) THEN
   messg = ' (all associated input collected)'
  ELSE
   messg = ''
  ENDIF
```

Thank you for your attention!