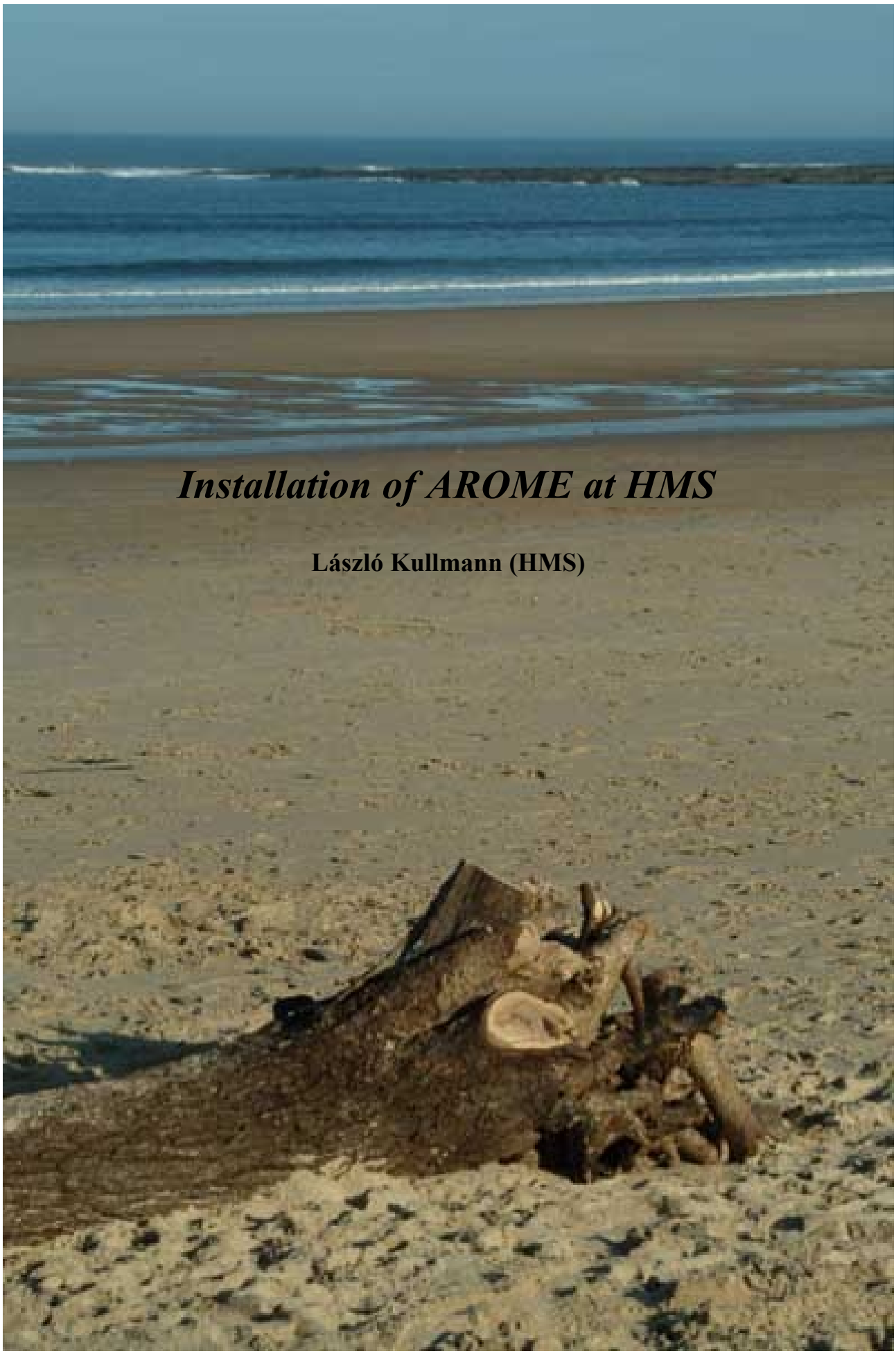

# *Installation of AROME at HMS*

**László Kullmann (HMS)**

## 1. Summary

We describe briefly the installation and the related problems (sec 2.) of AROME prototype (export version: cy29t1_t2, v05 created by GCO in Toulouse on the 5th of April) on the IBM computer at HMS. In sec. 3 we give some the technical details how we run AROME at HMS. The installation was done on IBM p655 cluster (AIX 5.2). We used the FORTRAN compiler, xlf 8.1.1.8 and the c compiler, xlc 6.0.0.10. We used gmkpack.6.1 to make the compilation. After correcting some bugs in the code the model is running and the comparison of the results obtained in Toulouse with the local run shows good agreement.

## 2. Compilation of AROME

The compilation was done with gmkpack. Some minor changes had to be performed on the software:

- Changing the **xlf_wrapper_loc** script, since we were unable to run the code compiled with omp optimization.
- Changing **aux/libspack.sh** script because in the original version all the project libraries were recreated even if one didn't modified the code under that project.
- Modify **aux/unsxrpack.sh** by adding underscore to the file names.

We compiled the code in 3 steps:

1. Setup for compilation: explicit interface routines, dependencies, compilation list. This step was run on 1 proc.
2. Compilation of the source. This was done on 8 proc. The compilation took 4260 *s* on the IBM cluster.
3. Creating libraries and binary. This step was also run on 1 proc. In the script **ics_arome** one should set ICS_START=2 and ICS_STOP=1 to avoid recompilation of the source. After submitting the first time this script the linking will fail due to undefined symbols. Therefore one should place files (0 byte) under src/unsxref/verbose/ with the name of undefined symbols (no trailing underscore should be used), and resubmit the script.
4. Some routines had to be modified in order to be able to run AROME.

### 2.1. Problem with explicit interface in latlon_grid

The subroutine **mse/internals/latlon_grid.mnh** calls some subroutines (e.g. *latlon_gridtype_conf_proj*) with the optional argument: PDIR. (This variable is also optional in *latlon_grid*.) When running the program, PDIR will never be present in *latlon_gridtype_conf_proj* however it is present in *latlon_grid* and the subroutine *latlon_gridtype_conf_proj* is called with argument PDIR.

It turned out that the explicit interface for the subroutine *latlon_gridtype_conf_proj* does not exist in *latlon_grid*. To solve the problem new modules were written containing the interfaces and which are then used by *latlon_grid*:

mse/module/modi_latlon_gridtype_cartesian.mnh

mse/module/modi_latlon_gridtype_conf_proj.mnh

mse/module/modi_latlon_gridtype_lonlat_reg.mnh

### 2.2. Problem of surfex I/O

When writing the surfex file: AROMOUT_{hhh}.lfi the program stops if one runs on more than one processor. The STOP commands are in the *write_surf[xx]_aro* subroutines, where [xx] stands for e.g. x1, x0, n0, depending what kind of variable (scalar, 1 dimensional real vector, etc)

should be written to the file.

These subroutines call the routine *fmwrit* with the argument KRESP which is an output error code indicating whether the writing was successful (KRESP=0) or not (KRESP/=0). After there is a condition: IF(KRESP/=0) ... STOP, i.e. if there were a problem with writing the program stops.

The routine *fmwrit* is only called by the first processor (in case of IYPROC=1), i.e. the variable KRESP is initialized only in this case. However the condition checking the value of KRESP is not only done for the first processor. Of course if KRESP were initialized to zero there would be no problem. We tried to compile the code with **–qinitauto** (which should initialize all variable to 0) but it did not work only when we compiled with -O0 flag. So the solution is to change the code in order to call the condition checking KRESP only in case of IYPROC=1.

### 2.3. Problem connected to the value of NPROMA

To run AROME we used first the same namelist that was used in Toulouse. However we encountered a strange problem. If we set NPROMA=1500 (which was used in Toulouse) after 3 hour integration when reading the coupling file the program aborts in *arp/setup/sugridua.F90* subroutine at the line:

IF( MAXVAL( GFL(:,:,YTKE%MP,:) ) = = 0.0_JPRB .AND. MINVAL( GFL(:,:,YTKE%MP,:) ) = = 0.0_JPRB)

The problem is in MAXVAL() calculation. If one reduces NPROMA (e.g. to the value 1000) then it does not abort. It seems that if the array is too large (since the first dimension of the array GFL is NPROMA) the MAXVAL function does not work properly. After discussing with the french colleagues it turned out that on IBM one should use smaller NPROMA values. Indeed for NPROMA<500 the above problem disappeared.

We faced however other problems related to the variable NPROMA. We discovered that the results are slightly different when running with two different NPROMA value (e.g. 20 and 40). It turned out that the problem comes from microphysics, if one switches off microphysics (LMICRO=.F. in namelist) then the two runs for different NPROMA values give the same result. One source of the problem was found and this is the following:

In subroutine *rain_ice_sedimentation* (which is inside subroutine *rain_ice*) the following calculation is performed:

```
IF( ISEDIM >= 1 ) THEN
  PRRS(:,:,:) = PRRS(:,:,:) * ZTSTEP
  ...
  PRRS(:,:,:)=PRRS(:,:,:) / ZTSTEP
ENDIF
```

In the above expression the value ISEDIM is bigger then 0 only if one of the element of PRRS exceeds a prescribed minimum value ($10^{-20}$). The first dimension of array PRRS has the size of NPROMA. Let us assume that we run with NPROMA=20 and NPROMA=40. Let us further assume that all the values in PRRS(1:20,:,:) (i.e. for all the values which belong to the first dimension less then 21) are below the threshold. In this case ISEDIM will be 0 if we run with NPROMA=20 so the array PRRS will not be touched. For NPROMA=40 however ISEDIM will be 1 and the array PRRS will be first multiplied by ZTSTEP and at the end of the IF condition will be divided by ZTSTEP. The values of PRRS belonging to the first dimension less then 21 will be not touched during the calculation so those values will be just multiplied and then divided by ZTSTEP. The strange behaviour occurred that the multiplication and division did not give the same value! The difference is small but this small difference will start to grow during the integration. We don't know yet whether this is a compiler bug or not the right compilation options were set.

The code was changed in the way that the multiplication and the division was put outside the IF condition, i.e.:

```
PRRS(:,:,:) = PRRS(:,:,:) * ZTSTEP
IF( ISEDIM >= 1 ) THEN
  ...
ENDIF
PRRS(:,:,:) = PRRS(:,:,:) / ZTSTEP
```

In this case the difference between the two runs with two different NPROMA values still exist (but it is smaller) which indicates that there is still some problem at some other place in the code.

## 3. Running AROME at HMS (technical details)

1. The AROME model is running at HMS on a domain covering just Hungary (see Table I.) with 2.5km horizontal resolution and 49 vertical levels. The integration for 36 hours takes around 7 hour on 16 processors on IBM p655 cluster.

Table I. AROME domain properties

| NDLUN | NDLUX | NMSMAX | NDGL | NDGUX | NSMAX |
|---|---|---|---|---|---|
| 250 | 240 | 124 | 160 | 150 | 79 |
| ELONC | ELATC | ELON0 | ELAT0 | EDELX | EDELY |
| 19.55 | 47.33 | 19.55 | 47.33 | 2488.67 | 2488,67 |

We describe here what are the main steps to run AROME.

1. First an ALADIN forecast should be run to produce initial as well as lateral boundary condition for AROME.
2. One has to run an ee927 configuration on the ALADIN forecast files to interpolate them to AROME geometry. One has to add ozone, and aerosol fields to the output, which is done by an external program, **INIOZOAER** (this step will not be needed in the future since the ARPEGE coupling files contain these fields). One also needs to reinitialize TKE (turbulent kinetic energy) field to bigger value than zero. This latter is done by the external program, **protke**.
3. One has to create a special initial surface file (which is in LFI format) since the externalized surface code can only read this special format. This step can at the present stage of the code only be done in Toulouse:

- Create a PGD file (for every month of the year) containing the physiographic fields. One needs to do this step only once for a given domain.
- Convert the ALADIN +0h forecast to GRIB format.
- Create the surface initial file (TEST.lfi) using the ALADIN initial file (in GRIB format) and the PGD file.

4. After creating the initial and LBC files one can run AROME.
5. Note that the upper air fields will be written to the output files ICMSH*EXPR*+00?? (as usually) while the surface fields will be written to ARMOUT_.???.lfi file (which is <u>not</u> an FA file).
6. To visualize the surface fields we have several options but at HMS we use 2 of them currently:

- Use the MESO-NH program: **diaprog**. One has to convert first the AROME surface output file to *dia.lfi* format with the **conv2dia** program.

- At HMS we use the HAWK visualization system. The advantage is that it is easier to compare the results with other model output or with observations. The HAWK system reads the data in latlon grid and in netcdf format. A small program was created to interpolate the field from lambert to regular latlon grid and to convert to netcdf format.

7. To visualize the upper-air fields we also use the HAWK system. Latlon fullpos should be run on the forecast output and than convert the PF file to netcdf format. Only adiabatic fullpos can be run since some surface fields are missing from the ICMSH files.

CONTENTS