ministerie van verkeer en waterstaat

koninklijk nederlands meteorologisch instituut

Technical description LAM and OI: Limited Area Model and Optimum Interpolation analysis

Second edition

W.C. de Rooy
L.M. Hafkenscheid

Technical reports; TR-134a
Technische rapporten; TR-134a

De Bilt, 1992

de bilt 1992

postbus 201
3730 AE de bilt
wilhelminalaan 10
tel. (030) 206911
telex 47096
fax (030) 210407

publicatienummer: Technical reports=
technische rapporten; TR-134a
Division of Dynamical Meteorology

Technical description LAM and OI ; Limited Area
Model and Optimum Interpolation analysis
Second edition

First published 1991

auteur: W.C. de Rooy
L.M. Hafkenscheid

U.D.C.: 551.509.313

ISSN: 0169-1708

ISBN: 90-369-2011-6

Technical description LAM & OI

(Limited Area Model and Optimum Interpolation analysis)

LAM Version: 1.1.1
OI Version: 1.0.0
Date: 09-01-'92
Author: Wim de Rooy
Leo Hafkenscheid

INDEX

1 Introduction

The FMLAM (Fine Mesh Limited Area Model) is a numerical weather prediction model for the intermediate range (6 to 24 hours). The (FM)LAM originates from the limited area version of the ECMWF gridpoint model [1]. For the analysis LAM uses Optimum Interpolation in a procedure developed by Cats [2]. The initialisation uses a Bounded Derivative method developed by Bijlsma [3].

This paper gives the technical description of the software, needed to run the LAM system, i.e. OI analysis, initialisation and LAM forecastmodel. It is intended for use by programmers and scientists in charge of the maintenance and administration of the LAM/OI system. The scientific description is given in the references quoted above.

This paper describes LAM release 1.1.1. The differences (relating to this paper) between this release and the release (1.0.0.) described in the previous technical description are:

1 Automatical reset first guess errors
2 New installation procedure
3 Remove of scratch files

Two appendices, H and I are appended.

2 Organisation and installation of the LAM and OI systems

2.1 Basic software

All the software and data, needed to install LAM and OI is contained in two directories, indicated by the variables $SYSLAM and $SYSOI, which are of the form '<path name>/syslam111' and '<path name>/sysoi100'. The last three digits (here 111 and 100) correspond with the version number (here: 1.1.1 for LAM and 1.0.0 for OI).
The versions under development have the path name '/ontw0/ontwapl/lahafken'.

Subdirectories of $SYSLAM are:

clim	for LAM and OI climatology files
doc	for documentation files
getbdrs	for software used to extract ECMWF boundary fields
iscripts	for LAM installation scripts
rscripts	for LAM and OI run scripts
source	for LAM source files

Subdirectories of $SYSOI are:

data	for a character file used by the oi-program 'events'
iscripts	for OI installation scripts
source	for OI source files

At installation (see below) these subdirectories are copied to the environment where LAM will be installed, except 'iscripts'. The subdirectory 'rscripts' is copied to the subdirectory 'scripts'. For the contents of the subdirectories (after installation) we refer to Appendix C.

2.2 Installation

2.2.1 Installation of LAM

For installation of LAM the contents of $SYSLAM/iscripts must be copied to a working directory. The scripts in 'iscripts' are:

allexes.sc
candlam.sc
compile.sc
compload.sc
installLAM.sc
makesclim800.sc
mklamenv.sc
prelamenv
sysset

Before installation can be invoked, three environmental variables must be set, namely:

EXPCODE	op111 (for operational runs)
STATDAT	/prod0/prodapl/prodhirl (for operational runs)
DYNDAT	/prod1/prodapl/prodhirl (for operational runs)

For real time experiments the first two characters of EXPCODE should be 'rt'.

Installation is invoked by starting:

```
installLAM.sc $RUNMDL
```

where $RUNMDL is the variable name of the directory where the LAM directories should be written.
The script installLAM.sc:
-creates the file 'setlamenv' which is used to set the environmental variables needed for installation
-copies the subdirectories 'clim/LM800', 'getbdrs', 'doc', 'rscripts' (to subdir: 'scripts') and 'source'
-starts on request the script 'allexes.sc' which controls all the compilations and writes the executables produced in the directory '$RUNMDL/exe'.

The script 'allexes.sc' creates two subdirectories in the working directory where the installation is done:

obj	for object files (temporary) and libraries (permanent)
log	for loggings and compiler standard output

These subdirectories could be removed when complete compilations are required only. The installation scripts give, however, the possibility of updating the executables by 'partial compilation' i.e. by compilation of separate programs or subroutines. In this case 'obj' and 'log' should be kept.

The procedure for 'partial compilation' is:

a)

```
candlam.sc <lam block name> [<subroutine names>]
```

where <lam block name> is one of the source file names (extension included):

dyns.p8
ecpp.p8
knpp.p4
mast.p8
phys.p8
spec.p4
tsfs.p8

and [<subroutine names>] is a list of subroutine names (without extension) out of the 'lam block' specified. Only the objects of these subroutines will be compiled and replaced in 'lambd9.exe' or 'pplin.exe'. If the list of subroutines is omitted all subroutines of the block will be compiled.

b) Similarly:
compload.sc <program name> [<subroutine names>]

where <program name> is one of the source file names (extension included):

```
bounder.p
chtogf.p
daytsf.f
hextobin.p
ligrib.p
listf.p
mlsurf.p
mxhist.p
mxtims.p
petosi.p
prebd.p
prhist.p
rwexa.f
sigrib.p
testprebd.p
```

and c):
compile.sc <lib name> [<subroutine names>]

where <lib name> is one of the source file names (extension included):

```
tsf.p
utils.p
various.p
```

NOTE: 'partial compilation' is recommended for experimental use only.

2.2.2 Installation of OI

The installation of OI is done in a way similar to that of LAM.
A separate working directory should be used for copies of the scripts in '$SYSOI/iscripts'.
These scripts are:

```
allexes.sc
compile.sc
compload.sc
installOI.sc
preoienv
```

NOTE: these scripts are not identical to those in $SYSLAM/iscripts, although the same names are used.

Installation of OI is invoked by:

```
installOI.sc $RUNANA
```

where $RUNANA is the variable name of the directory where the OI directories should be written.

Appendix E gives a short summary of the installation procedure. A more complete description is given in the documentation file $RUNMDL/doc/installatie (in Dutch).

2.3 'Static' and 'dynamic' data and software.

After installation the directories $RUNMDL and $RUNANA contain all files needed to run the LAM and OI, including the necessary climatological data. These files do not change when running the model. Therefor they are referred to as 'static' data and software.
Not included, however, is the variable data such as observations, output, loggings etc. That is called 'dynamic data'.

2.3.1 'Static' data and software

After successful installation $RUNMDL contains the subdirectories:
clim
doc
exe
getbdrs
scripts
source

and $RUNANA contains the subdirectories
data
exe
source

with the contents as given in Appendix C.

NOTE: for the current operational implementation the values of $RUNMDL and $RUNANA are:
$RUNMDL = /prod0/prodapl/prodhirl/lam
$RUNANA = /prod0/prodapl/prodhirl/oi

2.3.2 'Dynamic' data

The 'dynamic' data are contained in subdirectories of the directory named $LAMDAT. The subdirectories are:

bdrs	for ECMWF boundary fields
dbas	for output GRIB files (only for non-operational runs)
log	for logging files
mars	for 'mars update' files (implementation deferred)
oc	for observation files
temp	for temporary scratch files
work	for work space

NOTE: the current operational value of $LAMDAT is: /prodl/prodapl/prodhirl

2.4 Running the LAM/OI

2.4.1 Environmental variables

With the command 'source $RUNMDL/scripts/setenvlamrun' the environmental variables are set. This command is executed at the beginning of scripts to start the LAM (resumelam.sc) or scripts which can run 'stand alone' (e.g. getOC.sc).
A listing of setenvlamrun with the current operational settings is given in Appendix A. For the operational LAM/OI the settings need not to be changed.

In principle there are two main modes for running LAM/OI: A 'real time' mode and an 'experimental' mode. This is recognised by the setting of $EXP.

a) 'real time' mode
This is the case when $EXP is of the form "opxxx" or "rtxxx", where "xxx" is preferably a three digit number. But any other three character combination for "xxx" will do as well.
"opxxx" is for operational runs (output fields in grib data base GVDB).
"rtxxx" is for non-operational real time runs (output fields in $LAMDAT/dbas).
In both cases the observation files (APL)OC<dtg of observation> ON OPER must be present on A6-dev, while the ECMWF files of the form ECMO_PQS_<dtg of ECMWF analysis>00_00000_AB must be present in the GRIB data base GVDB. (dtg is date/time in the format YYMMDDHH)

b) 'experimental' mode
This is the case when $EXP has not any form as given in case a) (output fields in $LAMDAT/dbas).
For the 'experimental' mode the observation files must be made available as $LAMDAT/oc/apl_oc<dtg of observation> or as $LAMDAT/work/LAMF_OCB_<dtg of observation>00_00000_OC
The ECMWF boundary files must be made available in the form of $LAMDAT/bdrs/ECMO_PQS_<dtg of verification>00_<3 digit forecast period>00_GB

2.4.2 output

The next thing to do is checking the script '$RUNMDL/scripts/cycle.sc'.
This script controls the analysis/initialisation/forecast cycle. Especially the output specifications for fields and gridprints (Time Series Files) should be examined if one wants something different from the operational settings.
More information is given in chapter 6: 'Postprocessing'.

2.4.3 start/suspend/stop LAM

LAM is started by submitting the script
$RMSCR/resumelam.sc [<dtg1> [<dtg2>]]
(NOTE: $RMSCR = $RUNMDL/scripts)
If no argument is given LAM resumes for the date/time 3 hours later than the last completed cycle.
If only <dtg1> is given LAM will start for this date/time; if also <dtg2> is given LAM will start with <dtg1> making 'update' cycles only until <dtg2>.
From <dtg2> on also 'forecast' cycles will be made if required.
If no analysis first-guess is available for the required date/time you are asked to type 'cold_start' to confirm that you really want a cold start.

Continuation of LAM is controlled by the last line of the file '$LAMLOG/status'.
If the first word of this line is 'go' LAM will continue normally.
If this word equals 'stop' LAM will finish the current cycle and then stop the control.
If this word equals 'wait' LAM will wait 300 seconds and then read this line again.
The script resumelam.sc appends the line 'go' to the file '$LAMLOG/status' so that LAM continues normally until 'stop' or 'wait' is appended by e.g. the command 'echo stop >> $LAMLOG/status'.

For more details we refer to the documentation files in '$RUNMDL/doc'.

3 Description of scripts and programs

3.1 Introduction

In this chapter the scripts and programs will be discussed in the same order and with the same numbering as in the script/program tree in paragraph 3.2. If you want the description of e.g. lam.sc, you first look in the script/programtree to see the corresponding number, in this case 1.6. With the aid of this number it is easy to find the description of lam.sc in paragraph 3.3.

The structure of the descriptions is the same for each program or script, namely:

-tree number -script or program-name -symbolic argument(s)

-directory: Only in case of an external program the directory of the executable is mentioned here. The directory structure of LAM scripts and programs is described in chapter 2.

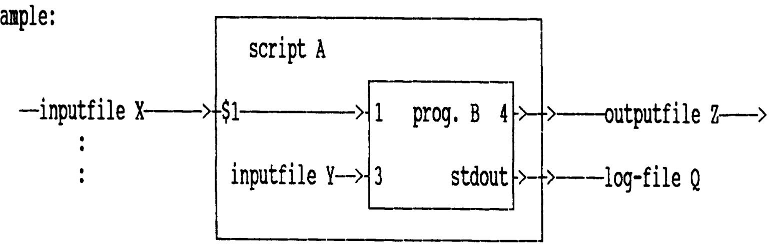
-description of arguments: -general description plus the actual given arguments when the script is used in a LAMrun.

-description of program or script

-input/output-files: the actual given input/outputfiles when prog/script is used in a LAMrun.

-schematical presentation input/output

Example:



Explanation example schematical presentation input/output

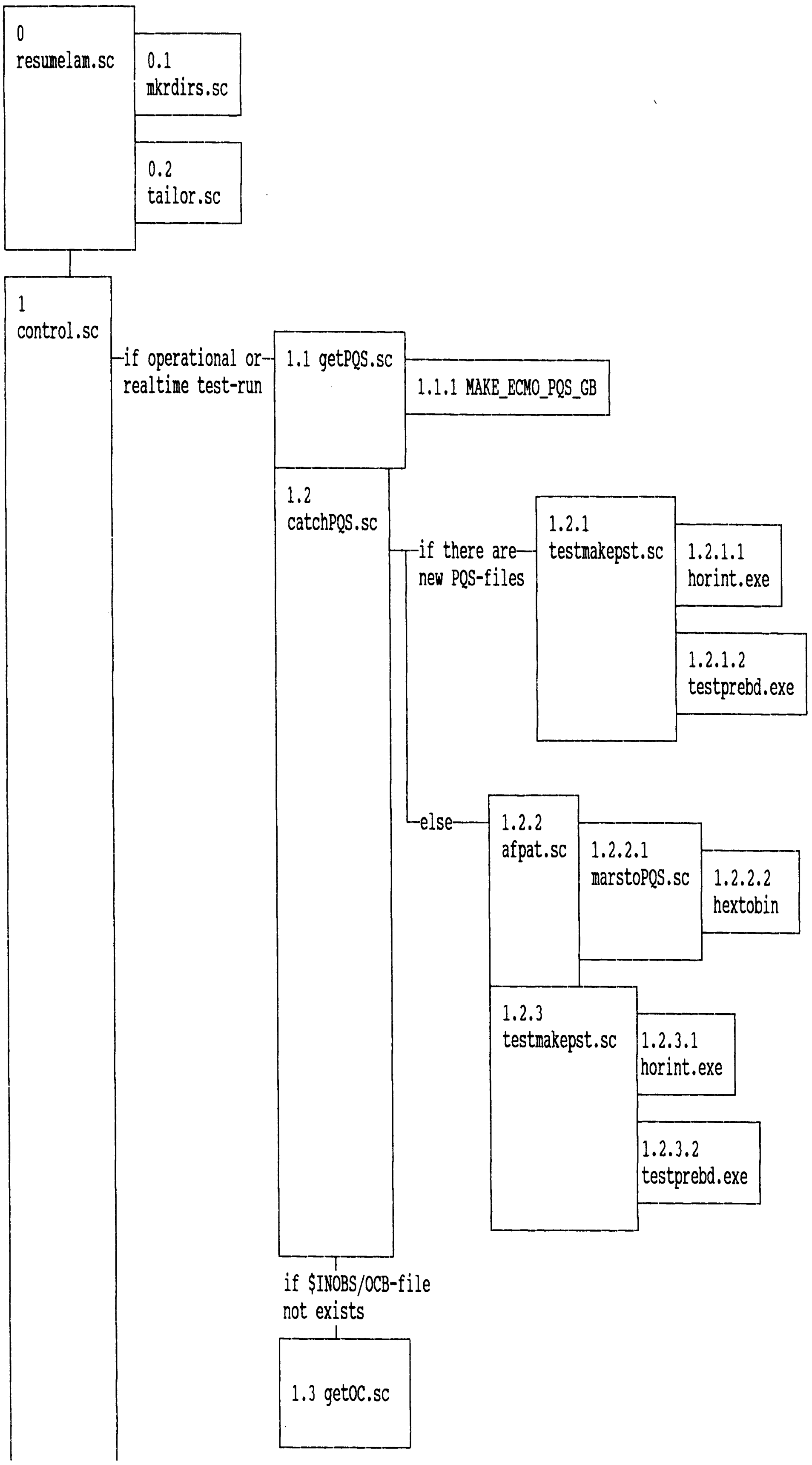
Inputfile X is called (this time given as argument $1) by the same script that called script A because the arrow of inputfile X starts outside script A. Inputfile Y however is called by script A (the arrow starts inside script A). Inputfile Y can be accessed by program B via fortran unitnumber 3.

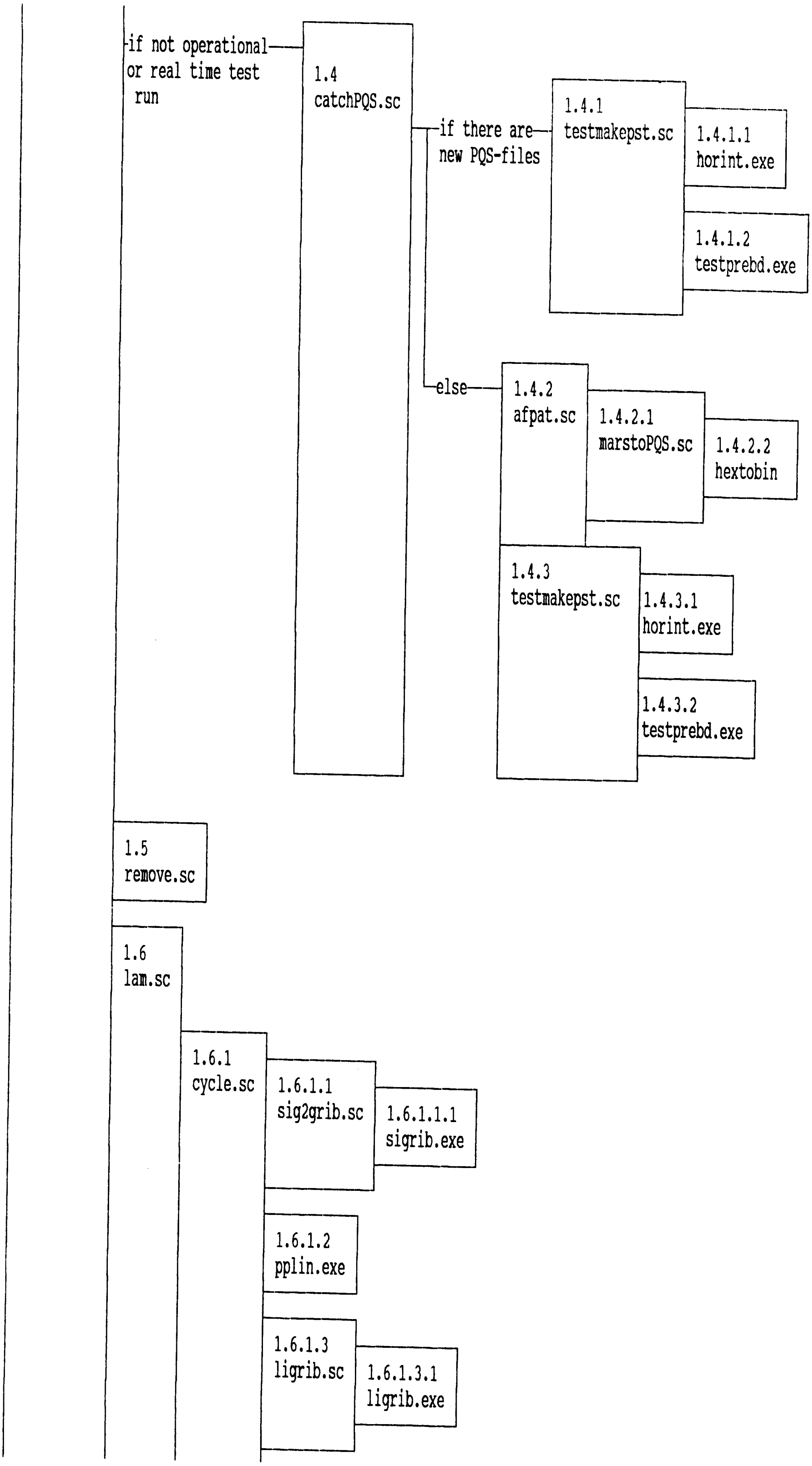
The standard output of prog. B is redirected (by script A) to log-file Q. Standard output and standard input is abbreviated as respectively stdout and stdin.
Colon's beneath the input or outputfile means that the in-or output consists of a number of files of those type.

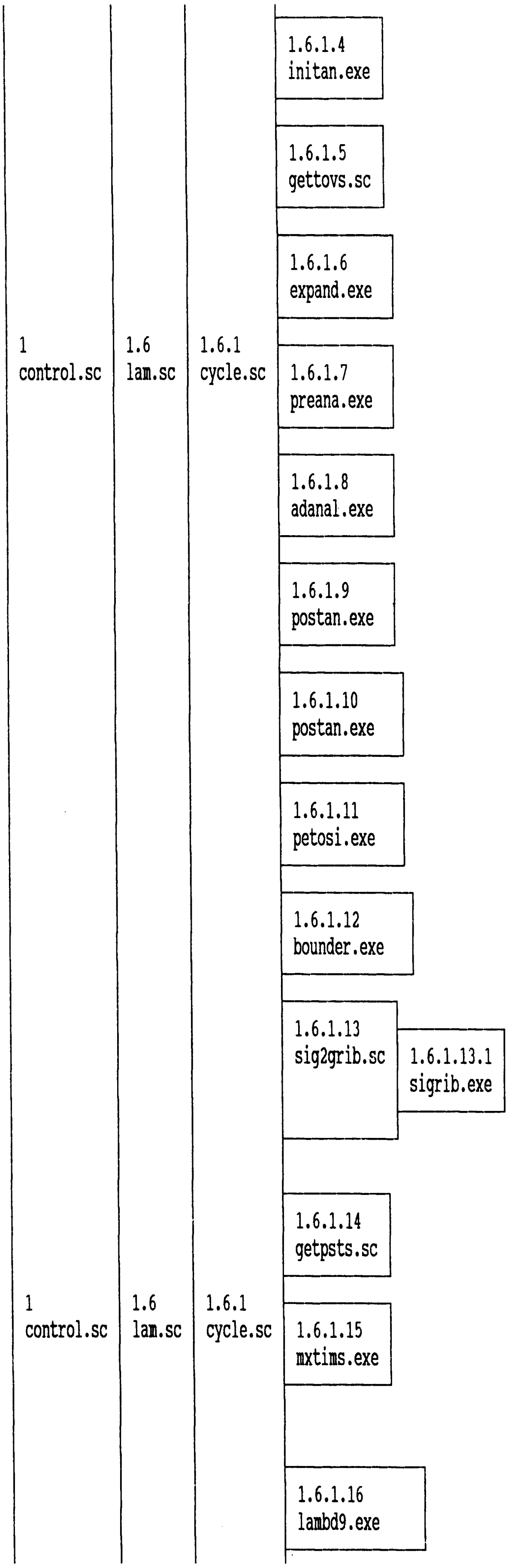
The symbolic arguments are written in Unix-notation ,i.e. $1 is the first argument ,$2 the second etc.. An argument is optional when it is notated between square brackets.
$$ is in unix the string which contains the processnumber. It is used in temporarily directories and files.

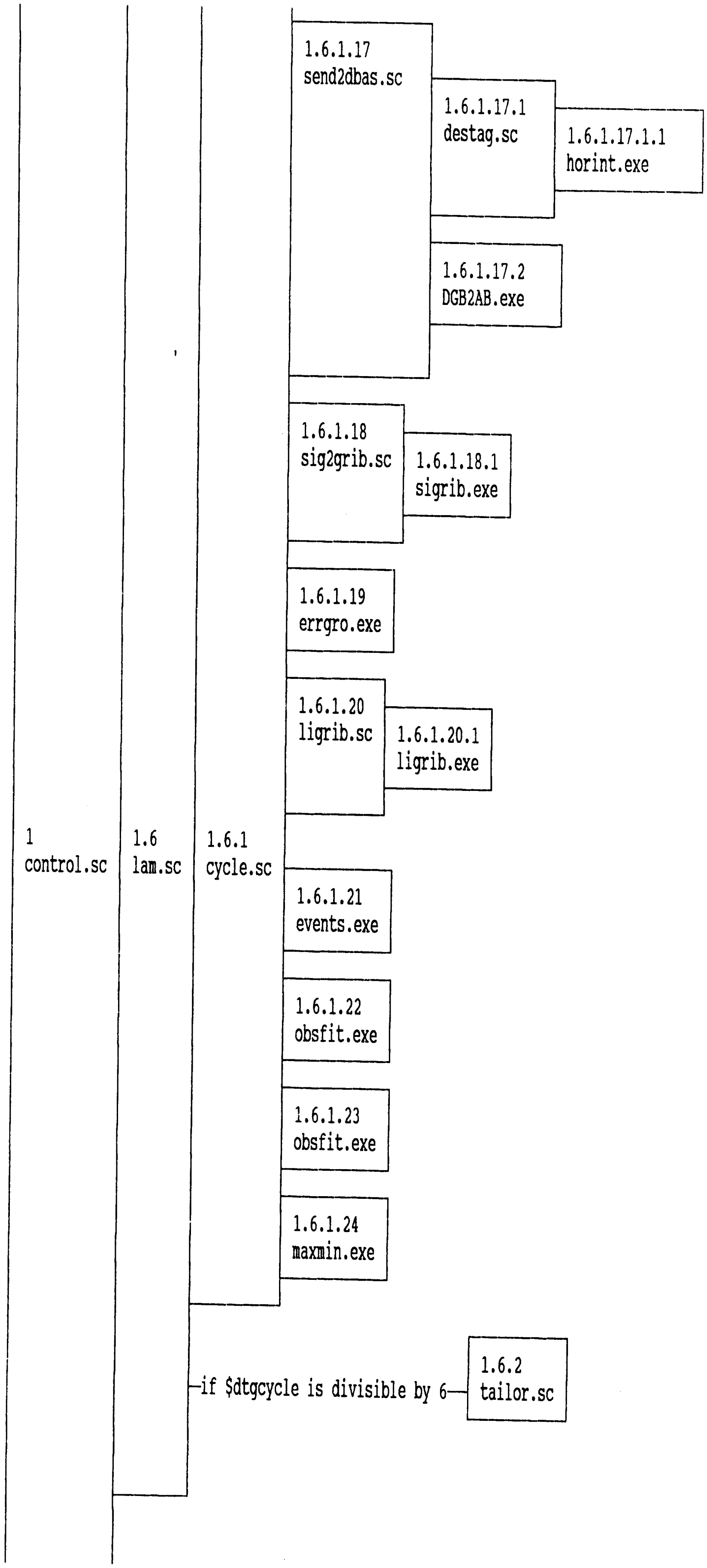
The directories are notated as environmental variables (see Appendix A). The directory structure of the programs and scripts is explained in chapter 2. The datafilenames used in this chapter (for example: FMT_GB) are described in Appendix D.

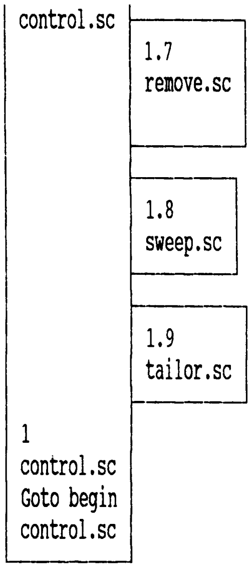
3.2 Script/program-tree




if not operational or real time test run
1.4 catchPQS.sc
if there are new PQS-files
1.4.1 testmakepst.sc
1.4.1.1 horint.exe
1.4.1.2 testprebd.exe
else
1.4.2 afpat.sc
1.4.2.1 marstoPQS.sc
1.4.2.2 hextobin
1.4.3 testmakepst.sc
1.4.3.1 horint.exe
1.4.3.2 testprebd.exe
1.5 remove.sc
1.6 lam.sc
1.6.1 cycle.sc
1.6.1.1 sig2grib.sc
1.6.1.1.1 sigrib.exe
1.6.1.2 pplin.exe
1.6.1.3 ligrib.sc
1.6.1.3.1 ligrib.exe


1.6.1.4
initan.exe
1.6.1.5
gettovs.sc
1.6.1.6
expand.exe
1
control.sc
1.6
lam.sc
1.6.1
cycle.sc
1.6.1.7
preana.exe
1.6.1.8
adanal.exe
1.6.1.9
postan.exe
1.6.1.10
postan.exe
1.6.1.11
petosi.exe
1.6.1.12
bounder.exe
1.6.1.13
sig2grib.sc
1.6.1.13.1
sigrib.exe
1.6.1.14
getpsts.sc
1
control.sc
1.6
lam.sc
1.6.1
cycle.sc
1.6.1.15
mxtims.exe
1.6.1.16
lambd9.exe


1
control.sc
1.6
lam.sc
1.6.1
cycle.sc
1.6.1.17
send2dbas.sc
1.6.1.17.1
destag.sc
1.6.1.17.1.1
horint.exe
1.6.1.17.2
DGB2AB.exe
1.6.1.18
sig2grib.sc
1.6.1.18.1
sigrib.exe
1.6.1.19
errgro.exe
1.6.1.20
ligrib.sc
1.6.1.20.1
ligrib.exe
1.6.1.21
events.exe
1.6.1.22
obsfit.exe
1.6.1.23
obsfit.exe
1.6.1.24
maxmin.exe
if $dtgcycle is divisible by 6
1.6.2
tailor.sc


control.sc
1.7
remove.sc
1.8
sweep.sc
1.9
tailor.sc
1
control.sc
Goto begin
control.sc

3.3 Description of the scripts and programs

-0 <u>resumelam.sc</u> [$1] [$2]

arguments: Default (calling resumelam.sc without an argument) is to start the new lam-run 3 hours after the last completed run, but:
-<u>If $2 is present</u> ,it is the dtg of the first forecast and $1 is the first updatecycle. All cycles till $2 are updatecycle's.
-<u>if only $1 is present</u> ,it is the start-dtg and the first dtg divisble by $FRFC (forecast frequency, e.g. 6h) will be a forecast cycle.

description: With this script you can start a new lam-run or resume an interrupted run. First the environmental variables for running LAM and analysis cycles are set in setenvlamrun (see Appendix A).
The script gives a warning if there's no useful BQS and/or FST-field. If no FST-field is available the script asks you to type "cold_start". If you do so a copy of a PST-field is used as first guess. When no PST file is found (and $EXP=rtXXX or opXXX) resumelam.sc tries to get the required PQS-files and converts them to PST-files (all cases) using getPQS.sc (for description see 1.1) and catchPQS.sc (see 1.2). The date at which the cold start is made is written in $LAMLOG/coldstarts.
At the end of resumelam.sc "go" is appended to the file $APLSMS/status, the (scratch-)directories $TEMP/$$ $WRKDAT/$$ and $LAMLOG/$$ are removed and control.sc is invoked.

-0.1 mkrdirs.sc

description: In this script some environmental variables (containing directories) are made if they doesn't already exist.

-0.2 tailor.sc $1 $2

arguments: remove first part of file $1 (full path name
           required), but keep last $2 lines. When tailor.sc
           is called from resumelam.sc the actual given
           arguments are: $1 = $LAMLOG/controlzero.log
                          $2 = 40

description: see arguments

-1 control.sc $1 [$2]

arguments: -If $2 is present ,it is the dtg of the first forecast and $1 is the first update cycle. All cycles till $2 are updatecycle's.
-if only $1 is present ,it is the start-dtg and the first dtg divisble by $FRFC (forecast frequency, e.g. 6h) will be a forecast cycle.

description: In the beginning of this script the status is determined (by reading $APLSMS/status) to take the proper action. The status can be:

go :just go on with the script (most usual)
wait:sleep for some seconds
stop:stop (exit 2)

For operational runs ($EXP=opXXX) or real time tests ($EXP=rtXXX) control.sc starts looking for PQS-files in $GVDB and observationfiles at the a6 (production) 2 hours after $dtgcycle (=date time group of the cycle). All this is described in 1.1 getPQS.sc till 1.3 getOC.sc. If the OCB and PQS-files are available a cycle (analysis, initialisation and forecast) is submitted.
If $EXP <> opXXX or rtXXX then the program is looking for PQS-files continuously. If they are available lam.sc is started. Then the OCB-file has to be present in the directory $INOBS otherwise lam.sc will be stopped (exit 17) and run again after 300 seconds.

Once a month (2nd at 12UTC) the first guess errors (BQS-file) are reset to climatological values (rtxxx and opxxx runs only). This is done because of stability reasons.
All normal messages in control.sc are redirected (by resumelam.sc) to control.log. This log-file reports the wall clock time and the dtg of the cycle. It also reports the first PST file (if available) in a list of PST files.
controlzero.log is the errorfile logging of control.sc. The standard output of control.sc is redirected to this file by resumelam.sc.

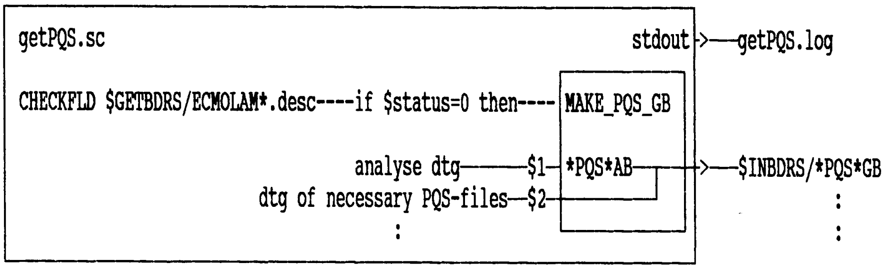
output-logfile: $LAMLOG/control.log
$LAMLOG/controlzero.log

-1.1 getPQS.sc $1

arguments: This argument is given to determine which PQS-files have to be processed. The actual given argument is $dtgcycle.

description: In this script the necessary PQS_GB-files are extracted from the gribfields database using MAKE_PQS_GB. But first the PQS fields are checked with external CHECKFLD. CHECKFLD checks the PQS-asimof files using the descriptor files in $GETBDRS. These *.desc files are made by $RMSCR/makeECMOLAMdesc.sc (see chapter 5) and contain information about the quantities which are necessary for a LAM run. If $status=0 after running CHECKFLD MAKE_PQS_GB is invoked.
The standard output of getPQS.sc is appended to getPQS.log

outputfiles: $INBDRS/ECMO*PQS*GB
logfile: $LAMLOG/getPQS.log


getPQS.sc
stdout
>—getPQS.log
CHECKFLD $GETBDRS/ECMOLAM*.desc----if $status=0 then----
MAKE_PQS_GB
analyse dtg———$1
dtg of necessary PQS-files—$2
*PQS*AB
>—$INBDRS/*PQS*GB
:
:
:

-1.1.1 MAKE_ECMO_PQS_GB $1 $2

directories: executable can be made using the makefile
$GETBDRS/make_ecmo_pqs_gb.mk
source: $GETBDRS/make_ecmo_pqs_gb.f

arguments: $1 = dtganalyse = the dtg of the PQS_AB file (normally previous day 12 o'clock).
$2 = forecast (hours). This argument is changing during one getPQS.sc run (loop), so all necessary PQS_GB-files (+00, +06, ...+42) are extracted from the PQS_AB file.

description: This program produces the ECMO_PQS_{dtg}00_{forecast period}00_GB files (determined by $1 and $2) from the ECMO_PQS_{dtg}00_00000_AB file using subroutine getfld.

input: $GVDB/ECMO_PQS_{dtg}00_00000_AB
output: $INBDRS/ECMO_PQS_{dtg+forecast}00_00000_GB

-1.2 catchPQS.sc [$1]

arguments: When this argument is given it is the first (oldest) PQS dtg to be processed. All PQS files are processed if no argument is given. The actual given argument is: $1 = $dtgpst (=$dtgcycle-3h)

description: At the beginning of this script the environmental variables are set. This script treats PQS files and lamg-ecmwf-output files.

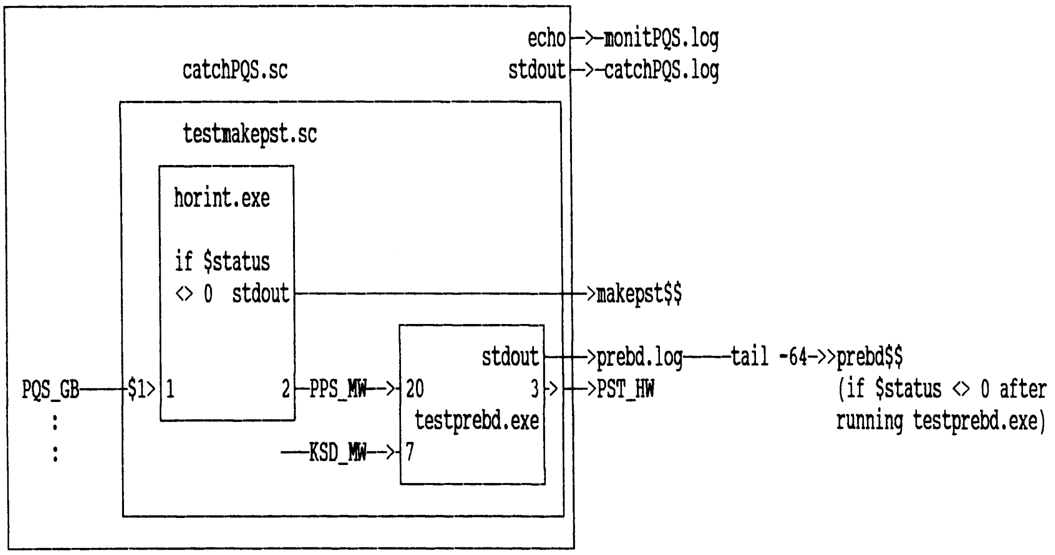
1 When PQS files are available in the directory $INBDRS the files are converted one by one to PST-files. Subsequently PQS- and PST-files (contain boundary conditions interpolated to the LAM grid) are moved to $WRKDAT.

2 When no PQS-files are detected in the directory $INBDRS the marsupdate cyclus is invoked. First $ECFILES/lamg_ecmwfoutput_** are converted to PQS-files in the directory $INSURF. These PQS-files contain only surface data and will be notated as *PQS*(surf) in this report. Actually the filename is the same for a 1- or 14 level PQS-file.
The PQS(surf) file is merged with the already existing PST-files from $WRKDAT.
The standard output of catchPQS.sc is redirected to catchPQS.log by control.sc. Which PQS-files are treated and in which mode they are treated (mars or normal mode) is written in monitPQS.log. In monitPQS.log you can also see which "old" PST file is removed from $WRKDAT.

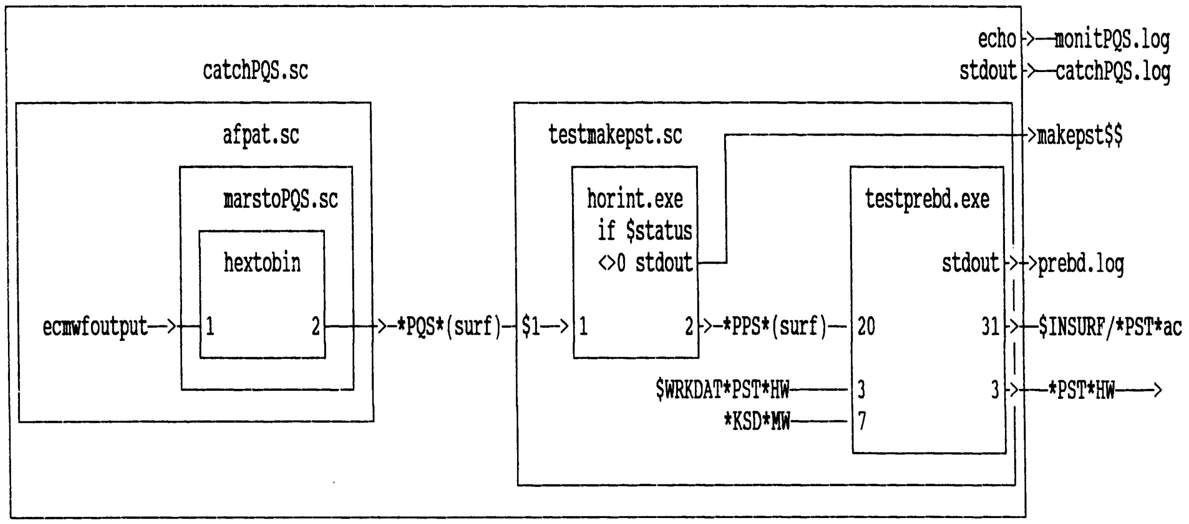
input/output-files: output: $INSURF/*PST*ac (in case of marsupdate)
$WRKDAT/*PST*_HW

logfile: $LAMLOG/monitPQS.log

1 When PQS files are available in the directory $INBDRS



2 When no PQS-files are detected in the directory $INBDRS (mars update cyclus)



-1.2.1 testmakepst.sc $1 [$2]

argument: $1 is the name (inclusive dtg) of the ECMO PQS-file which will be converted in this script. If $2='pqs' then the "1.2.1 testmakepst.sc" description (and all descriptions of programs called from 1.2.1 testmakepst.sc) is valid. Else the 1.2.3 (marsupdate-) description of testmakepst.sc is valid. So here the actual given second argument is 'pqs'.

description: In this script PST_HW files in LAMgrid are created from PQS_GB and KSD_MW files. The standard output of horint.exe and prebdy is redirected by testmakepst.sc to respectively makepst$$ and prebd.log.

input/outputfiles: input: $INBDRS/*PQS*GB
output: $INSURF/*PST*ac
$WRKDAT/*PST*_HW
logfiles: $LAMLOG/makepst$$
$LAMLOG/prebd.log
$LAMLOG/prebd$$ (only if $status <> 0 after running testprebd.exe)

-1.2.1.1 horint.exe (external)

directories: $DHORI/horint.exe

description: When this program is executed in this place ,it interpolates the PQS_GB file bilinear to the LAM grid. The outputfile is in MBW gridcode. The u and v windcomponents are staggered.

input/outputfiles:
input: see 1.2.1 testmakepst.sc
output: $INBDRS/*PPS*_MW
logfiles: $LAMLOG/makepst$$ (This log-file is removed if $status = 0 after running horint.exe)

1.2.1.2 testprebd.exe

description: This program prepares the PST-historyfile ,which contains the lateral boundary conditions, merged with climatological data from *KSD*MW, for a LAM-run.
The standard output of testprebd.exe is redirected to prebd.log. If $status <> 0 after running testprebd.exe then the last 64 lines of prebd.log are written in prebd$$.

input/outputfiles: input: $INBDRS/*PPS*_MW
$RMCLM/$AREA/*KSD*_MW
output: see 1.2.1 testmakepst.sc
logfiles: $LAMLOG/prebd.log
$LAMLOG/prebd$$ (only if $status <> 0 after running testprebd.exe)

1.2.2 afpat.sc (part of marsupdate cyclus (see page 20))

description: Afpat is an abbreviation for Automatic File Processing After Transfer. The hexadecimal lamg_ecmwfoutput_** files (only containing surface data) are converted to binair *PQS*GB files (notated as *PQS*(surf) in this paper).

outputfiles: $INSURF/*PQS*GB (surf)

1.2.2.1 marstoPQS.sc $1 $2 $3

arguments: $1 is the filenamebase
$2 is the date of analysis
$3 is the forecast period
So the inputfilename is $1$2$3

description: In this script hextobin is called with the correct input- and outputfile.

input/outputfiles: input: $ECFILES/lamg_ecmwfoutput_**
output: see afpat.sc

1.2.2.2 hextobin

description: This program converts the hexadecimal ecmwfoutput surface-file into a binair gribcode file containing the surface parameters. This *PQS*(surf) file will be merged with the already existing PST-file in 1.2.3 testmakepst.sc.

outputfile: $INSURF/*PQS*GB (surf)

-1.2.3 testmakepst.sc $1 [$2]

argument: $1 is the name (inclusive dtg) of the ECMO PQS-(surf) file which will be converted in this script.
Here the actual given second argument is 'mars' so the 1.2.3 (marsupdate-) version of testmakepst.sc is described.

description: In this script PST_HW files (with updated surface climatology) in LAMgrid are created from PQS_GB (surf), KSD_MW and already existing PST-files. The standard output of horint.exe and prebdy is redirected by testmakepst.sc to respectively makepst$$ and prebd.log.

```
input/outputfiles:  input:  $INSURF/*PQS*_GB    (surf)
                    output: $INSURF/*PST*ac
                            $WRKDAT/*PST*HW
                    logfiles: $LAMLOG/makepst$$
                            $LAMLOG/prebd.log
                            $LAMLOG/prebd$$ (only if $status
                            <> 0 after running testprebd.exe)
```

-1.2.3.1 horint.exe (external)

directory: $DHORI/horint.exe

description: When this program is executed in this place ,it interpolates the PQS_GB (surf) file bilinear to the LAM grid. The outputfile is in MBW gridcode. The u and v windcomponents are staggered.

input/outputfiles:
input: *PQS*GB (surf)
output: $INBDRS/*PPS*_MW
logfiles: $LAMLOG/makepst$$ (This log-file is removed if $status = 0 after running horint.exe)

-1.2.3.2 testprebd.exe

description: In 1.2.3.2 testprebd.exe the PPS*(surf) file which contains climatological ecmwf surface data is merged with the already existing $WRKDAT/*PST*HW file.
The standard output of testprebd.exe is redirected to prebd.log. If $status <> 0 after running testprebd.exe then the last 64 lines of prebd.log are written in prebd$$.

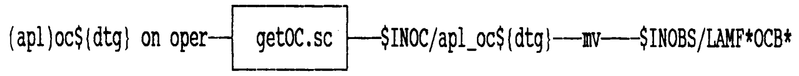
input/outputfiles: input: $INBDRS/*PPS*_MW(surf)
$RMCLM/$AREA/*KSD*_MW
$WRKDAT/*PST*HW
output: see 1.2.3 testmakepst.sc

logfiles: $LAMLOG/prebd.log
$LAMLOG/prebd$$ (only if $status <> 0 after running testprebd.exe)

1.3 getOC.sc $1

arguments: $1 is the dtg of the cycle

description: This script is called only during an operational run ($EXP =opxxx) or a real time test ($EXP=rtXXX). It is run after 1.1 getPQS.sc and 1.2 catchPQS.sc if more then 2 hours are elapsed since $dtgcycle (verification time) and the $INOBS/*OCB* file isn't available.
First the environmental variables are set. The script looks at the A6 Production for the oc-file. If the file is found it is copied to the convex (dir = $INOC) using $LOCBIN/getf.



-1.4 catchPQS.sc
(+ all his child processes = 1.4.*)

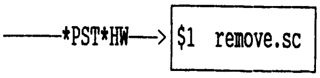
description: see 1.2 catchPQS.sc
All child processes called from 1.4 catchPQS.sc are exactly the same as those called from 1.2 catchPQS.sc.

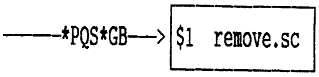
-1.5 remove.sc $1 [$2]

arguments: <u>The first argument</u> defines which files (namely $TEMP/*$1*) will be removed.
<u>The second argument</u> defines how many of those files are kept at least for each verification-time (default is 1 file).
When the script is called (twice) from this place in control.sc the following arguments are given:
- $1=PST and $2=1
- $1=PQS and $2=1

description: see arguments

inputfiles: $TEMP/*PST*HW
$TEMP/*PQS*GB





1.6 lam.sc $1 [$2]

arguments: $1 is the dtg of the cycle.
If $2 is present and equals 'update' then an 'update only' cycle is submitted ,i.e. only a 3 hours forecast will be made for producing a first guess for the next analysis (currently extracted to 6 hours for 'quickupdate')
If no $2 is present then a 30 hours forecast is made if the dtg of the cycle is divisble by $FRFC (forecast frequency, e.g. 6h) ,which is set in setenvlamrun (appendix A).
The second argument 'update' is actually given to lam.sc if $dtgcycle (=first argument control.sc) < $dtgupdate (last argument control.sc). Else only $1 ($dtgcycle) is given as argument.

description: This script submits one LAM-cycle if the observation file is found. The environmental variables are set. If $QUEUE is set (in setenvlamrun) then the script cycle.sc is submitted in queue $QUEUE.
The standard output of lam.sc is redirected (by control.sc) to lam.log. This log-file reports the wall clock time and the dtg of the cycle. It also reports the first PST file (if available) in a list of PST files and an error-message if the cycle is stopped.

output -logfile: $LAMLOG/lam.log

-1.6.1 cycle.sc $1 $2 [$3]

arguments: $1 is the dtg for the following cycle. The updatemode is set if the second or third argument is Update. If there are three arguments , then the second one is the name of the experiment ($EXP).

If the updatemode in lam.sc is set (second argument of lam.sc = update) ,then the arguments of cycle.sc are:

```
      $1=dtgcycle
      $2=$EXP
      $3=Update
else:
      $1=dtgcycle
      $2=$EXP
```

description: this script runs one complete LAM-cycle including analysis, initialisation, modelrun and post-processing. At the end of cycle.sc the temporary ($$) directories and files are removed.
The standard output of cycle.sc is redirected (by lam.sc) to cycle.log. In this log-file it is possible to see where the cycle-run went wrong. The file contains the starting- and finishing times of the fortran programs called in cycle.sc.
It also reports: -the links made by getpsts.sc
-which FST and FMT files are produced by lambd9.exe
-the name (=dtg) of the outputfile which contains physical information of the run
-possible errors
-some other information

Beside cycle.log, which is filled with the standard output of cycle.sc ,some other logfiles are used in cycle.sc. A selection of the physical input (namelists) and output of the programs called in cycle.sc ,is stored in the file $LAMLOG/$dtg (abbreviated as $outp in cycle.sc). The file $cwd/$$/bdrun (abbreviated as $outx in cycle.sc) contains the same information as $outp plus the names of some datafiles used in the run. $outx however is removed at the end of cycle.sc.

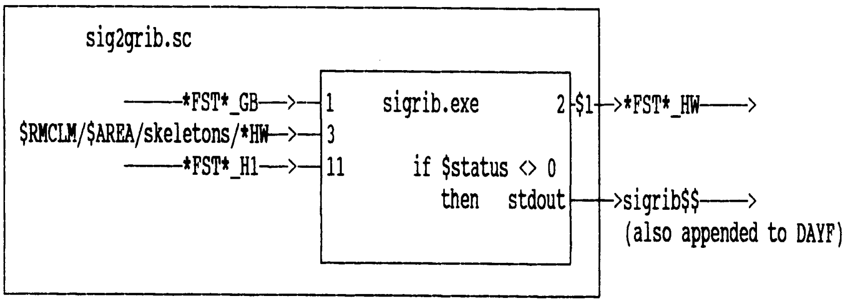
logfiles: $LAMLOG/cycle.log
$LAMLOG/$dtg
$cwd/$$/bdrun

1.6.1.1 sig2grib.sc $1 $2 [$3]

arguments: $1 is the (standard) σ-file to be packed or unpacked. The outputfile is also $1 but with trailing _HW replaced by GB or vice versa. In case of packing, a descriptor-recordfile is given with filename $1 and trailing _HW replaced by H1.
$2 is the areacode
$3 is either P (pack (default)) or U (unpack).
When this script is called from this place in cycle.sc the arguments are:
$1=$WRKDAT/*FST*HW
$2=$AREA
$3=U

description: This script packs or unpacks a σ-file (see arguments) with the use of sigrib.exe. When a σ-file has to be unpacked, a skeleton-file is necessary. This skeleton is found in $RMCLM/$AREA/skeletons/*HW. When sig2grib.sc cannot find one or more of the inputfiles it is reported in the logfile sigrib$$. Sig2grib.sc also redirects the standard output of horint.exe to the file sigrib$$. If $status=0 after running horint.exe then sigrib$$ is removed.

input/outputfiles: output: $WRKDAT/*FST*HW
logfile: $RMLOG/$EXP/sigrib$$



1.6.1.1.1 sigrib.exe

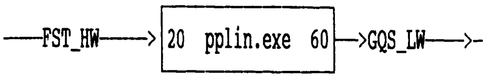
description: see 1.6.1.1 sig2grib.sc

input/outputfiles: input: $WRKDAT/*FST*GB
$RMCLM/$AREA/skeletons/*HW
$WRKDAT/*FST*H1
output: see 1.6.1.1 sig2grib.sc

1.6.1.2 pplin.exe

description: this program interpolates the guessfield on σ-levels in history-format to a guessfield on p-levels in lineformat (using subroutine SITOLP).

input/output:
$WRKDAT/*FST*HW
$cwd/$$/*GQS*LW



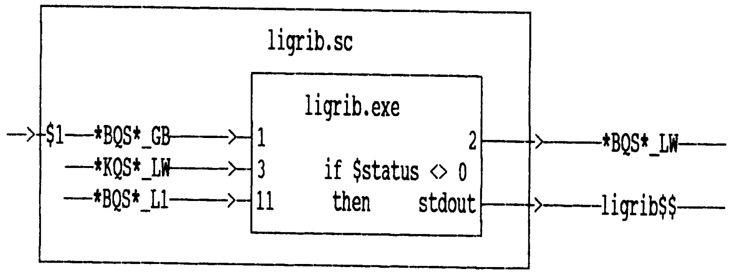
1.6.1.3 ligrib.sc $1 $2 [$3]

arguments: $1 is the title of the gribcode file to be produced (or to be used as input when unpacking). The title of the lineformat file is also $1 but with trailing GB replaced by LW, that of the descriptor recordfile is obtained by replacing GB by L1.
During unpacking the script will look for a KQS_LW file as skeleton file in all subdirectories of $3 (default ~ $WRKDAT).
$2 is U for unpacking or P for packing (default). If $2 is neither U or P then $2 is used as lookdirectory for skeletonfiles.
ligrib.sc is called from cycle.sc with the following arguments:

```
$1=$WRKDAT/*BQS*_GB
$2=U
$3=$RMCLM/$AREA
```

description: This script packs or unpacks a lineformat-file (using ligrib.exe). The standard output of ligrib.exe is redirected to ligrib$$.

input/outputfiles: output: $WRKDAT/*BQS*_LW
logfile: $RMLOG/$EXP/ligrib$$



1.6.1.3.1 ligrib.exe

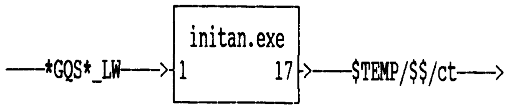
description: see 1.6.1.3 ligrib.sc

input/outputfiles: input: $WRKDAT/*BQS*_GB
$WRKDAT/*BQS*_L1
$RMCLM/$AREA/*KQS*_LW
output: see 1.6.1.3 ligrib.sc

1.6.1.4 initan.exe

description: This program prepares the analysis control file (ct).

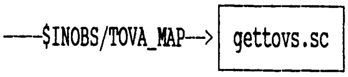
input/outputfiles: input: $cwd/$$/*GQS*_LW
output: $TEMP/$$/ct



1.6.1.5 gettovs.sc $1 $2 (not yet implemented)

arguments: This script generates a list of $INOBS/TOVA_MAP_ (satellite-) files with a start time between $1 and $2+30 minutes and establishes symbolic links of those files with fort.71 ,72 ,73 .....
When this script is called from cycle.sc the arguments are:
$1 = $dtg-2hours ($dtg is first argument of cycle.sc).
$2 = $dtg + 1hour

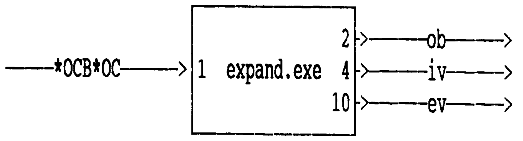
inputfiles: $INOBS/TOVA_MAP_



1.6.1.6 expand.exe

description: This program puts the observationfile (in characters) and TOVS into the right form ,i.e. in Convex words ,for further analysis.

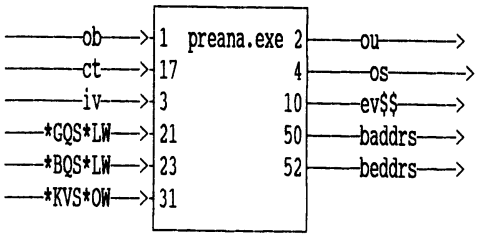
input/outputfiles: input: $INOBS/*OCB*OC
output: $TEMP/$$/ob
$TEMP/$$/iv (inventory)
$TEMP/$$/ev



1.6.1.7 preana.exe

description: Preana (preanalysis) creates observation increments normalised by first-guess errors and checks against first-guess.

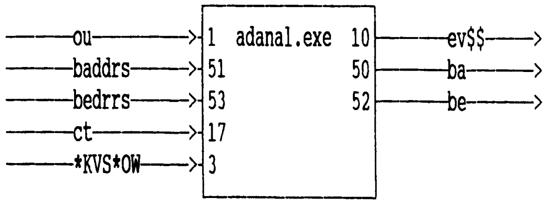
```
input/outputfiles: input:  $TEMP/$$/ob
                           $TEMP/$$/ct
                           $TEMP/$$/iv
                           $cwd/$$/*GQS*LW
                           $WRKDAT/*BQS*LW
                           $RMCLM/*KVS*OW
                   output: $TEMP/$$/ou
                           $TEMP/$$/os
                           $RMSCR/ev$$ (contents will be
                                     appended to $TEMP/$$/ev)
                           $TEMP/$$/baddrs
                           $TEMP/$$/beddrs
```



1.6.1.8 adanal.exe

description: This program performs the 3-dimensional multivariate optimum interpolation of mass and wind.

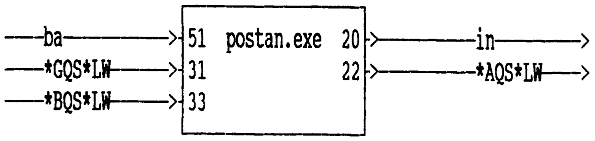
```
input/outputfiles: input:  $TEMP/$$/ou
                           $TEMP/$$ct
                           $TEMP/$$/baddrs
                           $TEMP/$$/beddrs
                           $RMCLM/$AREA/*KVS*OW
                   output: $RMSCR/ev$$ (contents will be
                                     appended to $TEMP/$$/ev)
                           $TEMP/$$/ba
                           $TEMP/$$/be
```



1.6.1.9 postan.exe

description: When this program is called from cycle.sc for the first time it reorders and denormalises the analysis output of adanal (ba) to obtain the analysed fields (AQS) and increments (in).

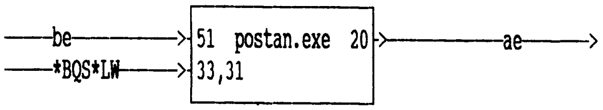
```
input/outputfiles: input:  $TEMP/$$/ba
                           $cwd/$$/*GQS*LW
                           $WRKDAT/*BQS*LW
                  output:  $TEMP/$$/in
                           $cwd/$$/*AQS*LW
```



1.6.1.10 postan.exe

description: This time the program reorders the analysis error output of adanal (be) to obtain analysiserrors.

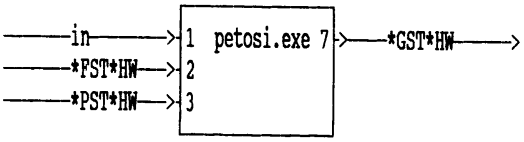
```
input/outputfiles: input:  $WRKDAT/*BQS*LW (is linked by 2
                                           unitnumbers to
                                           postan)
                           $TEMP/$$/be
                  output:  $TEMP/$$/ae
```



1.6.1.11 petosi.exe

description: This program performs the vertical interpolation to σ (model-) levels of the analysis increments on p-levels (in), and adds them to the guessfield. A PST file (dtg = latest valid ECMWF time) is added for the surface climatology. The complete analysis output on σ -levels in history format is written in GST_HW.

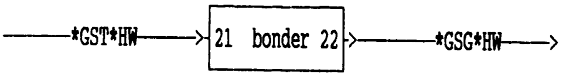
```
input/outputfiles: input:  $TEMP/$$/in
                           $WRKDAT/*FST*HW
                           $WRKDAT/*PST*HW
                   output: $WRKDAT/*GST*HW
```



1.6.1.12 bounder.exe

description:Initialisation of analysis with the bounded derivative-method.

```
input/outputfiles: input:  $WRKDAT/*GST*HW
                   output: $WRKDAT/*GSG*HW
```

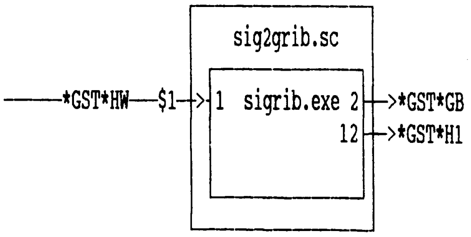


1.6.1.13 sig2grib.sc $1 $2 [$3]

```
arguments: general description arguments see 1.6.1.1.
           Now the arguments are: $1 = $WRKDAT/LAMF*GST*HW
                                  $2 = $AREA
                                  $3 = P
```

description: see 1.6.1.1

```
input/outputfiles: input:  $WRKDAT/*GST*HW
                   output: $WRKDAT/*GST*GB
                           $WRKDAT/*GST*H1
```



1.6.1.13.1 sigrib.exe

see 1.6.1.1.1 and 1.6.1.13

1.6.1.14 getpsts.sc $1 $2 $3 [$4]

arguments: This script generates a list of $4/LAMF_PST*HW files ($4 is default $WRKDAT) with a verification time starting at $1+6 (if $1 is even) ,$1+3 (if $1 is odd) ,increasing by 6. It establishes links of those files with fort. ($2+1), fort. ($2+2) ,etc.. The maximum number of timesteps to be examined is given in $3.
The actual given arguments are:
$1 = $dtg
$2 = 90
$3 = 6
$4 = $WRKDAT

input/outputfiles: input: $WRKDAT/*PST*HW
output: $WRKDAT/*PST*HW (now linked to unit numbers 91 ,92 , etc.

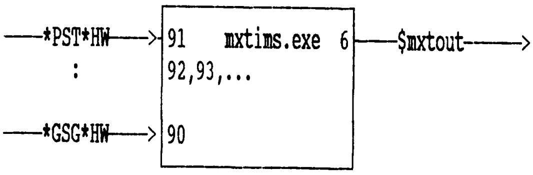
1.6.1.15 mxtims.exe

description: This program calculates the maximum absolute windcomponents in a number of historyfiles (in this case the PST historyfiles from getpsts.sc and the GSG_*${dtg}*HW file). From this maximum value the maximum time step for the fmlam-model is calculated for which the stability criterion is fulfilled.

The output of this program is put in an array named mxtout.

mxtims.exe	cycle.sc
ioutm -->	mxtout[1] = max. time step (default =450s)
iousph -->	mxtout[2] = max. number of time steps per hour (default=8)
ioumxv -->	mxtout[3] = max. absolute u and v windcomponents (default=0m/s)

input/outputfiles: input: $WRKDAT/*PST*HW
$WRKDAT/*GSG*HW
output: $cwd/$$/$mxtout



1.6.1.16 lambd9.exe

description: This program is the forecast model. It performs the following steps:

a) create a file of initial and lateral boundary conditions (subroutine MBDF).

b) forecast (controlled by namelist NEWRUN).

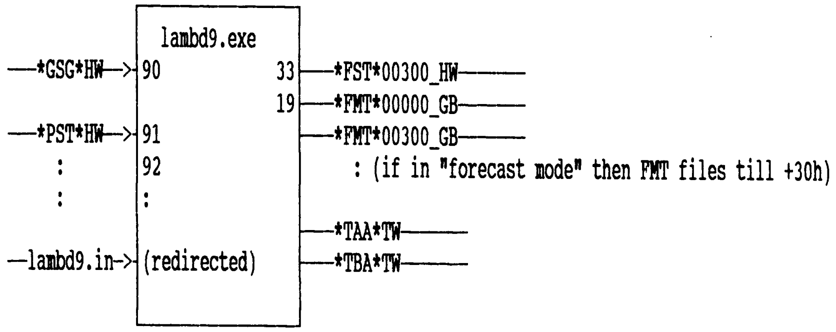
c) in-model postprocessing (controlled by namelist POSTIN)

Possible outputfiles from lambd9.exe:

TAA | _____timeserie-files
TBA |

*FMT*GB (internally packed to GB-format)
*FST*HW (σ 'history' file)

You can change the kind of variables and the area that are written in the TAA and TBA files by changing the variables ending in respectively A or B in the namelist NEWRUN. You can also change the contents of the FMT files by changing the variables in the namelist POSTIN. The namelists are created in cycle.sc. First the namelists POSTIN (A, B, C, D) are set for each forecast period seperately. So it is possible to specify the output (in FMT-files) for each forecast period. After this the general (for every forecast period) settings for POSTIN and the namelist NEWRUN are created. All the namelists are appended to the file lambd9.in.



```
input/outputfiles: see also the description.
            input: $WRKDAT/*GSG*HW
                   $WRKDAT/*PST*HW
           output: $WRKDAT/*FST*HW
                   $WRKDAT/*FMT*GB
                   $WRKDAT/*TAA*TW
                   $WRKDAT/*TBA*TW
                   standard printer output
```

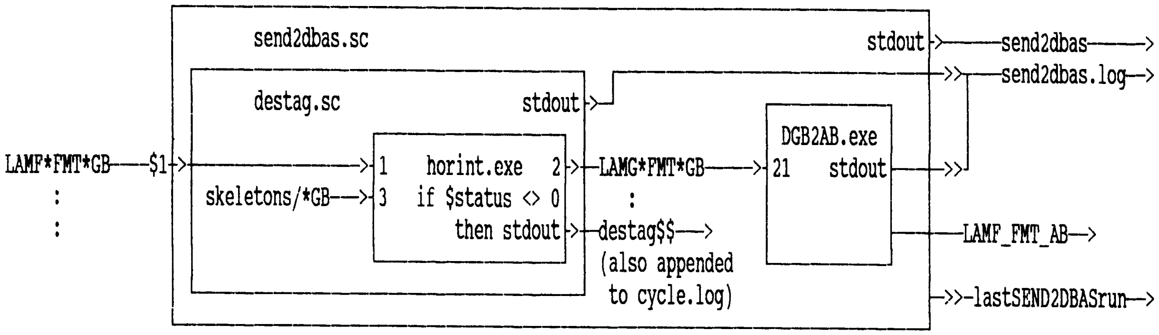
1.6.1.17 send2dbas.sc $1

argument: $1 is an array ($fmtl) which contains the FMT*GB files of the last forecast or updatecycle.

description: This script inserts the contents of the files from the list $1 into an asimof-file.

The standard output of destag.sc and DGB2AB.exe is redirected to send2dbas.log by send2dbas.sc. The analysedtg, the number of processed files and the date is written to the file lastSEND2DBASrun. The files not found by send2dbas.sc are written in send2dbas.

input/outputfiles: input: $WRKDAT/LAMF_FMT*GB
output: $GVDB/LAMF_FMT*AB
logfiles: $LAMLOG/send2dbas.log.
$APLSMS/lastSEND2DBASrun
$LAMLOG/error/send2dbas



1.6.1.17.1 destag.sc $1 $2

arguments: $1 is the title of the gribcode inputfile. The title of the outputfile is also $1 but with LAMF replaced by LAMG.
$2 is the areacode which defines the directory of the skeletonfiles.

description: This script extracts and destaggers (using horint.exe) gribcode fields and changes filetitle *LAMF* into *LAMG*.

```
input/outputfiles: input:  see 1.6.1.17 send2dbas.sc
                   output: $WRKDAT/*LAMG*FMT*GB
                   logfile: $RMLOG/$EXP/destag$$
```

1.6.1.17.1.1 horint.exe (external)

directory: see 1.2.1.1 horint.exe

description: For the internal modelrun the u and v components are located exactly between the gridpoints. This is not useful for most users and that's why horint.exe is used to destagger a file ,i.e. to interpolate the u and v components to the gridpoints by horizontal interpolation. A skeletonfile is used to define the grid. The standard output of horint.exe is redirected to destag$$ by destag.sc. If the $status = 0 after running horint.exe destag$$ is removed. If $status <> 0 destag$$ is kept and appended to cycle.log.

```
input/outputfiles: input:  $WRKDAT/LAMF_*FMT*GB
                           $RMCLM/$AREA/skeletons/*GB
                   output: $WRKDAT/LAMG_*FMT*GB
                   logfile: $RMLOG/$EXP/destag$$
```

1.6.1.17.2 DGB2AB.exe $1 (external)

directory: $DGBAB/DGB2AB.exe

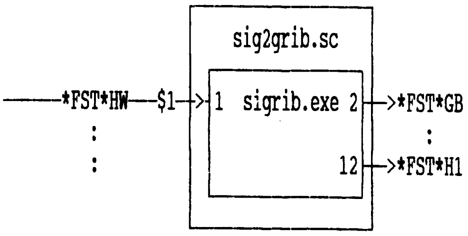
description: This program puts the LAMG_GB files into the LAMF_AB asimof file. The gribcodefields are read one by one. The product Definition Block (PDB) of each field is decoded. Then the field is written to the asimof file with that PDB as key. Administration is written in the send2dbas.log file.

```
input/outputfiles: input:  $WRKDAT/LAMG_FMT_GB
                   output: $GVDB/LAMF_FMT_AB
                   logfile: $LAMLOG/send2dbas.log
```

1.6.1.18 sig2grib.sc $1 $2 [$3]

arguments: see 1.6.1.1 sig2grib.sc for general description arguments. This time the given arguments are:

```
$1 = $fstl  (filelist of *FST*HW files)
$2 = $AREA
$3 = P
```



description: see 1.6.1.1 sig2grib.sc

```
input/outputfiles: input:  $WRKDAT/*FST*HW
                   output: $WRKDAT/*FST*GB
                           $WRKDAT/*FST*H1
```

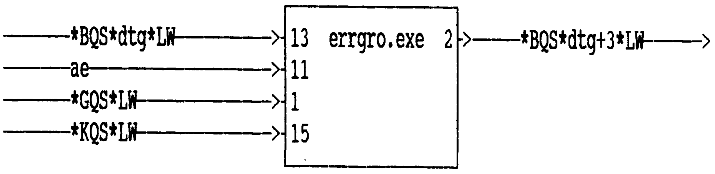
1.6.1.18.1 sigrib.exe

see 1.6.1.1.1

1.6.1.19 errgro.exe

description: This is a simple "forecast" model for the first guess error of the next cycle.

```
input/outputfiles: input:  $cwd/$$/*GQS*LW
                           $TEMP/$$/ae
                           $WRKDAT/*BQS*dtg*LW
                           $RMCLM/$AREA/*KQS*LW
                   output: $WRKDAT/*BQS*dtg+3*LW
```

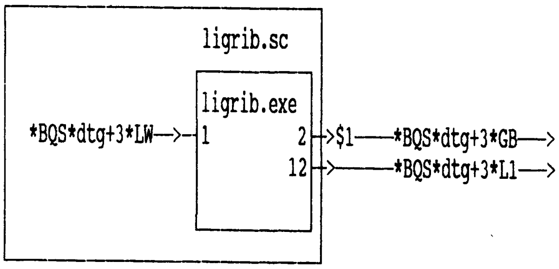


1.6.1.20 ligrib.sc $1 $2 [$3]

arguments: General description arguments see 1.6.1.3
ligrib.sc. This time the arguments are:
$1 = $nb (BQS_GB file with dtg+3)
$2 = P
$3 = $RMCLM/$AREA

description: This time the script packs the first guess-errorfile (BQS) for the next cycle (dtg+3).

input/outputfiles: output: $WRKDAT/*BQS*dtg+3*LW
$WRKDAT/*BQS*dtg+3*L1



1.6.1.20.1 ligrib.exe

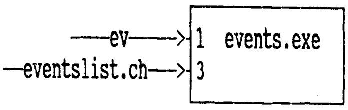
description: see 1.6.1.20 ligrib.sc

input/outputfiles: input: $WRKDAT/*BQS*dtg*LW
output: see 1.6.1.20 ligrib.sc

1.6.1.21 events.exe

description: This program generates a list of observation events ,discarded observations and rejected data. This list (=stdout) is inserted in the file $LAMLOG/$dtgcycle by cycle.sc (see also 1.6.1 cycle.sc logfiles).

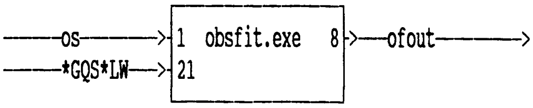
input/outputfiles: input: $TEMP/$$/ev
$RADAT/eventslist.ch



1.6.1.22 obsfit.exe

description: When this program is called for the first time in cycle.sc it produces a printoutput which shows the fit between firstguess and observations.

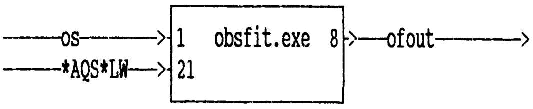
```
input/outputfiles: input:  $TEMP/$$/os
                           $cwd/$$/*GQS*LW
                   output: $TEMP/$$/ofout
```



1.6.1.23 obsfit.exe

description: When this program is called for the second time in cycle.sc it produces a printoutput which shows the fit between analysis and observations.

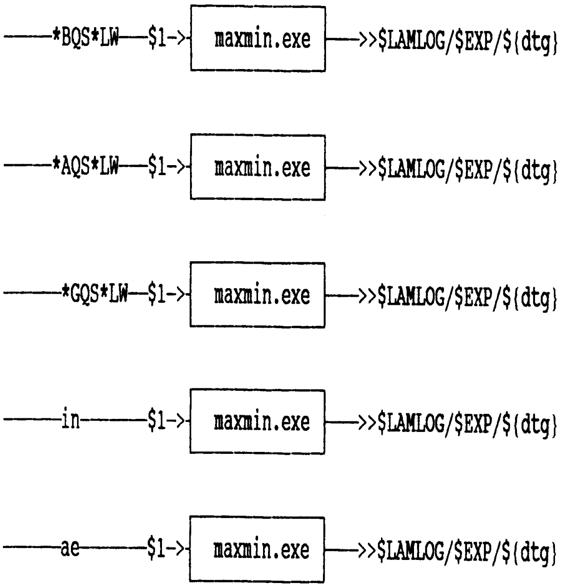
```
input/outputfiles: input:  $TEMP/$$/os
                           $cwd/$$/*AQS*LW
                   output: $TEMP/$$/ofout
```



1.6.1.24 maxmin.exe

description: This program determines from a lineformat-file the maximum and minimum value ,the position of these values ,the mean and standard deviation ,and sends them to the $LAMLOG/$dtg file and the standard printeroutput. The program is called 5 times from cycle.sc ,every time with a different inputfile.

```
input/outputfiles: input:   $WRKDAT/*BQS*LW
                            $cwd/$$/*AQS*LW
                            $cwd/$$/*GQS*LW
                            $TEMP/$$/in
                            $TEMP/$$/ae
                   output:  $LAMLOG/$EXP/${dtg}
```



1.6.2 tailor.sc

arguments: description see 0.2 tailor.sc
$LAMLOG/autobdries 256

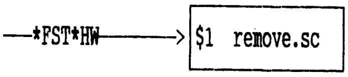
description: see 0.2. This script is invoked if $hour is divisible by 6.

1.7 remove.sc

arguments: see 1.5
When this script is called from this place in control.sc the following arguments are given:
$1=FST $2=2

description: see 1.5

inputfiles: $WRKDAT/*FST*HW



1.8 sweep.sc $1 $2 $3

arguments: This script removes all files $WRKDAT/*$1* with a dtg more than $3 hours older than $2. Example:

```
$RMSCR/sweep.sc PST 90041212 24
```

removes all files $WRKDAT/LAMF_PST_dtg000_???00_HW for which dtg is less than 90041112.
The actual given arguments are (sweep.sc is called 9 times):

$1 = PQS	$2 = $dtgcycle	$3 = 24
$1 = PST	$2 = $dtgcycle	$3 = 6
$1 = FMT	$2 = $dtgcycle	$3 = 24
$1 = FST	$2 = $dtgcycle	$3 = 12
$1 = GST	$2 = $dtgcycle	$3 = 12
$1 = OCB	$2 = $dtgcycle	$3 = 24
$1 = TAA	$2 = $dtgcycle	$3 = 72
$1 = TBA	$2 = $dtgcycle	$3 = 72
$1 = BQS	$2 = $dtgcycle	$3 = 24

description: At the beginning of this script the environmental variables are set. Rest see arguments.

1.9 tailor.sc $1 $2

arguments: general description see 0.2
The actual given arguments when tailor.sc is called (10 times) from control.sc are:

$1 = $LAMLOG/cycle.log	$2 = 2000
$1 = $APLSMS/lastLAMrun	$2 = 10
$1 = $APLSMS/lastSEND2DBASrun	$2 = 10
$1 = $LAMLOG/send2dbas.log	$2 = 1500
$1 = $LAMLOG/prebd.log	$2 = 1000
$1 = $LAMLOG/catchPQS.log	$2 = 200
$1 = $LAMLOG/getOC.log	$2 = 500
$1 = $LAMLOG/lam.log	$2 = 200
$1 = $LAMLOG/getPQS.log	$2 = 600
$1 = $LAMLOG/monitPQS.log	$2 = 100
$1 = $APLSMS/status	$2 = 10

description: see 0.2 tailor.sc

4 Use of surface climatology

In addition to the observation file (*OCB*) or ECMWF file (*PQS*) LAM needs the following parameters:

surface soil wetness
deep soil temperature ┐ (for layer between surface-
deep soil wetness ┘ and climatological deep layer)
surface radiation
albedo
climatological deep soil temperature
climatological deep soil wetness

These climatological parameters are given in

```
$RMCLM/$AREA/LAMF*KSD*{month}*MW
```

For every month there's one KSD file (see appendix C). The KSD files have been made by running a mars update cycle for every 15 th of every month in 1990 (except Jan. Feb. and March which are from 1991). During a LAM run the information in a KSD file can be overwritten by new ECMWF climatological data if an mars update cycle is invoked (see par.3.3 1.2 catchPQS.sc and 1.2.2 till 1.2.3.2).
Note: at the present these 'mars update files' are not available.
The climatological data in a KSD file is merged (1.2.1.2 testprebd.exe) together with the boundary conditions from the ECMWF (*PQS* file) into a $INBDRS/*PST*HW file which is used as ultimate input file for the forecast model.

5 Retrieval of ECMWF boundary files

The LAM is a limited area model so some variables at the boundaries of this area have to be prescribed during a forecast. These variables are extracted from an ECMWF asimof file ($GVDB/ECMO*PQS*AB) and copied to a gribfile named $INBDRS/ECMO*PQS*GB. The ECMWF variables are only given every six hours. During the forecast (lambd9.exe) these values are interpolated to LAM timesteps.
The variables to be extracted from the asimof file are:

gribcode	description	
104	geopotential height	[10m]
112	specific humidity	[0.1 g/kg]
123	wind component	[m/s]
124	wind component	[m/s]

for the levels: 10, 30, 50, 70, 100, 150, 200, 250, 300, 400, 500, 700, 850 and 1000 hPa

and

gribcode	description	
104	temperature	[°C]
151	snow depth	[cm]

at ground level [0m]

The script $GETBDRS/makeECMOLAMdesc.sc has been used to produce $GETBDRS/ECMOLAM{forecastperiod}.desc files (see appendix C). These files describe all the variables mentioned above. In 1.1 getPQS.sc subroutine $LOCBIN/CHECKFLD uses these *.desc files to check if the wanted variables are available in $GVDB/*PQS*AB (see also 1.1 getPQS.sc). If they are not then 1.1.1 MAKE_ECMO_PQS_GB isn't called and the LAM will use older PQS values or waits (if there are no older PQS values available for that perticular verification time).
Actually the *.desc files are the same for all ECMWF forecast periods (from +00 till +72 hours).
makeECMOLAMdesc.sc is not called in a LAM run because normally the variables to be extracted from the $GVDB/*PQS*AB file doesn't change.

The $INBDRS/*PQS*GB files still have to be interpolated to LAM (LM800) grid and merged with climatological information (see paragraph 3.3 1.2 or 1.4 catchPQS.sc and chapter 4 use of surface climatology). This ultimate boundary file is called $WRKDAT/*PST*HW which is used as input for the forecastmodel 1.6.1.16 lambd9.exe.

6 Postprocessing

Normal meteorological output of LAM is in two forms:
1- GRIB formatted fields
2- Time Series Files in TSF format (BUFR compatible)

The contents of these output files are controlled by the settings in $RMSCR/cycle.sc (See Appendix A for meaning of shell variables).

6.1 Fields

Output fields are written to the GRIB data base 'GVDB' for operational runs, or to the local data base '$LAMDAT/dbas' for experimental runs. Fields for all required forecast periods are written in the same file, which is labeled by the analysis time (APL file type 'FMT*AB').

Contents: (see $RMSCR/cycle.sc section 6.6)
The array variable 'pphour' contains a list of forecast periods expressed in hours, for which output fields are required.
Array 'ppcont' contains a list of file names specifying the required contents for the corresponding forecast period in 'pphour'. (section 6.62 in cycle.sc). In cycle.sc section 6.61 these contents are defined by namelist 'POSTIN'.

The meaning of the relevant variables in &postin is:

NMFD	NO. FIELDS AT MULTIPLE LEVELS
NFDML(10)	FIELD CODES (MULTIPLE LEVEL FIELDS)
NMLV	NO. OF LEVELS FOR MULT. LVL. FIELDS
NLVML(30)	LEVELS (MULTIPLE LEVEL FIELDS)
NSFD	NO. FIELDS AT SINGLE LEVEL
NFDSL(30)	FIELD CODES (SINGLE LEVEL FIELDS)
NLVSL(30)	LEVELS (SINGLE LEVEL FIELDS)
N2D	NO. 2-D FIELDS ON MODEL GRID
NGPCL(2,20)	FIELD CODE/LEVEL PAIRS (2-D FIELDS)

Appendix F gives a summary of the field and level codes that can be used.

6.2 Time Series Files

LAM has two kinds of time series files:
type A for surface and upper air data (APL file type *TAA*TW , see appendix D).
type B for surface data only (APL file type *TBA*TW).
Both file kinds use a fixed list of variables. These variables, together with their BUFR element descriptors is given in Appendix G.
The time resolution and the grid points for which TSF's are required are controled in cycle.sc by the following variables of namelist &NEWRUN :

NFRTA	:	FREQUENCY OF TSF WRITE UPS IN TIME STEPS (use shell variable $hr[] for frequency expressed in hours)
NITTA	:	INPUT TYPE FOR COORDINATES: 0: NORMAL LAT LON ELSE: GRIDPOINTS (relative to north-west corner)
NIUTA	:	TSF FILE UNIT NUMBER
NLATGA	:	ROW NUMBER OF GRID POINT (for NITTA .NE. 0)
NLONGA	:	POSITION OF GRID POINT IN ROW (for NITTA .NE. 0)
NRGPA	:	NUMBER OF GRID POINTS IN TSF (max 32)
TLATNA	:	NORMAL GRID LATITUDE OF GRID POINT (for NITTA .EQ. 0)
TLONNA	:	NORMAL GRID LONGITUDE OF GRID POINT (for NITTA .EQ. 0)

The number of co-ordinates given must correspond to NRGPA

The same holds for the variables NFRTB ... TLONNB for the TSF file 'TBA'.

Note: a reasonable frequency is 1 hour for type 'TAA' and 1 time step for 'TBA'.

Appendix A Environmental variables

In this appendix you can read the listing of setenvlamrun which is self-explaining.

setenvlamrun

```
#
# setenvlamrun
#
# set environmental variables for running LAM and analysis
# cycles
#
# interface:  source $RMSCR/setenvlamrun
#----------------------------------------------------------------
#
# setenvlamrun contains a number of commands that is used for
# creating an environment for running the lam system.
# The directories used for the lam are all given by
# environmental variabeles.
#
# Setenvlamrun must be present with the proper contents in the
# directory where the script is present with which the lam is
# started.
# Setenvlamrun is executed by a number of scripts that can run
# 'stand alone'.
#
# When running the lam in a new environment a number of
# settings must be adapted at installation. These settings are
# concentrated in chapter 1 (of this script).
#
#     The experiment code can be chosen:
#     e.g. setenv EXP op002 for operational runs.
#     $EXP indicates a.o. where the loggings can be found.
#      If $EXP has a value of the form "opxxx", where "xxx" is
#     free, then the run is a operational run, otherwise not.
#
#  On the base there is RUNMDL :e.g.
#                               /prod0/prodapl/prodhirl/lam
#          and          RUNANA  :e.g.
#                               /prod0/prodapl/prodhirl/oi
#     RUNMDL and RUNANA must be adapted so that the scripts
#     etc. can be found in the proper directories
#     subdirectories of RUNMDL are: exe, scripts, clim and
#                                   getbdrs.
#     subdirectories of RUNANA are: exe, data
#
#     Directories for data (input, output, scratch) are in
#     general subdirectories of $LAMDAT
#     $LAMDAT must be set.
#  WARNING: full path name of $LAMDAT should not have more
#  than 26 chars!
```

```
#
#      Directories for data bases have a default value for
#      operational runs.
#      The settings here are valid for experiment runs.
#      In general it is not necessary to change the settings
#      for area code, level code etc.
#      The frequency of forecasts is set by FRFC;
#      FRFC = 12 means e.g. that a 30 hours forecast is made
#      for analysis time 00 and 12 UTC.
#
# WARNING: full path name of $WRKDAT should not have more than
# 32 chars including final slash !!
#---------------------------------------------------------------
#
#  1.   set variable full path names
#
#  1.1 set experiment code
#
setenv EXP op002
#
#  1.2 set stem of 'lamtree' and 'analysis tree' (oi)
#
setenv RUNMDL /prod0/prodapl/prodhirl/lam
setenv RUNANA /prod0/prodapl/prodhirl/oi
#
#   1.3  directories for input data, work data , output data
and loggings
#
setenv LAMDAT /prod1/prodapl/prodhirl
setenv FRSTFG $LAMDAT/work
#
#  1.4 data bases (operational, for experiments in chapter
#      3.))
#
setenv ECFILES /prod0/prodapl/prodhirl/ecmwffiles
setenv GVDB /prod1/prodgvzg/GVDB
setenv TRDB /prod1/prodgvzg/TRDB
#
#  1.5 queues
#
setenv QUEUE pqhirlam
#===============================================================
#
#  2. set area code, level code, frequency of forecast runs,
printer
#
setenv AREA LM800
setenv LEVS L11
setenv FRFC 6
setenv PRINTER wolp
```

```
#
#  3.  directories for input data, work data , output data
#
setenv INBDRS $LAMDAT/bdrs
setenv INSURF $LAMDAT/mars
setenv INOC $LAMDAT/oc
setenv INOBS $LAMDAT/work
setenv WRKDAT $LAMDAT/work
setenv WRKDAT2 $LAMDAT/work
setenv TEMP $LAMDAT/temp
setenv PUTDBAS $LAMDAT/dbas/
setenv GETDBAS $LAMDAT/dbas/
#
#  3.1 loggings
#
setenv LAMLOG $LAMDAT/log
setenv APLSMS $LAMLOG
#
#  4. directories where scripts, objects etc necessary for
#  running the model and the analysis can be found
#
#  4.1 for running the model
#
setenv RMSCR $RUNMDL/scripts
setenv RMCLM $RUNMDL/clim
setenv RMEXE $RUNMDL/exe
setenv GETBDRS $RUNMDL/getbdrs
#
#  4.2 for running the analysis
#
setenv RAEXE $RUNANA/exe
setenv RADAT $RUNANA/data
#
#  4.3 additional objects/libraries (temporarily)
#      $DHORI for horint.exe
#
setenv LOCBIN /usr/local/bin
setenv DHORI $LOCBIN
setenv DGBAB $LOCBIN
#
#
#  5.  other settings
#
```

Appendix B Documentation files

This paper is intended for use by programmers and scientists. The documentation files in directory $RUNMDL/doc are primarily intended for operators (production) and systemmanagers. These files (all written in Dutch), summarised in the file "read_me", are:

1.1 diagnose This file contains a decision tree with complete explanation how to check the progress of the LAM or how to handle if something went wrong.

1.1.1 diatree stripped version of 1.1 diagnose (this time tree with minimum of explanation)

1.1.2 summary This descision tree is specially intended for the operators . It is a short summarised version of the 1.1 diagnose tree but with complete explanation and referring to interface options. The LAMinterface is a menu driven program for operators to stop, start and check the LAM.

1.2 beschrijving Some general information about the LAM and the relations to other processes. Also information about who to call in case of problems.

1.2.2 standaard General information about the LAM and relations to other processes in standard layout.

2.1 identifikatie Very compact global information about the LAM (also a short description of disk space usage).

2.2 resources Description of disk space, memory, CPU time and software usage at the Convex.

2.3 installatie Here the procedure to install the LAM is described. In this paper a description (in English) of the install procedure can be found in chapter 2 and appendix E (summary).

2.4 scratch files use of scratch files during a LAM run

relnotes Release notes. Description of the differences between LAM release 111 and preceding releases.

Appendix C Contents of $RUNMDL and $RUNANA and their subdirectories.

As mentioned in chapter 2 the basic software to generate the LAM can be found in $SYSLAM and $SYSOI (optimum interpolation part of the LAM also used by 'quickupdate').
During the install procedure (chapter 2 and appendix E) two directories are created which contain all the static software and static data. E.g. output data or boundary fields from the ECMWF are not static data, this data will change in time. Climatological data on the other hand is static data. The two ('static') directories are:

$RUNMDL containing files specifically meant for the LAM.

$RUNANA containing files concerning the LAM but these files can also be used by other products.

The subdirectories and contents of $RUNMDL are:

$RUNMDL/clim This directory is used for climatological data (see chapter 4). For the present LAM this data is put in subdirectory LM800 which is the area code. Contents of $RUNMDL/clim/LM800:

$RUNMDL/clim/LM800

see for description files in accordance with the APL file name conventions Appendix D

```
LAMF_KQS_0001000000_00000_LW
LAMF_KQS_0002000000_00000_LW
      :           :
      : for every       :
      : month     :
      :           :
LAMF_KQS_0001200000_00000_LW
LAMF_KSD_0001000000_00000_MW
LAMF_KSD_0002000000_00000_MW
      :           :
      : for every       :
      : month     :
      :           :
LAMF_KSD_0001200000_00000_MW
LAMF_KVS_0000000000_00000_OW
lcsmask           (land coast sea mask)
lsoro                    (land   sea
orography)

statz0            (surface roughnesses
                  of synopstations,
                  will be used if getobs
                  is implemented in the
                  LAM)
```

$RUNMDL/clim/LM800/skeletons

files in this directory aren't used as normal datafiles. This files are used whenever the grid has to be defined. E.g.:

unpacking gribfiles (see 3.3 1.6.1.1 sigrib.sc using *PST*HW)

destaggering a file (see 3.3 1.6.1.17.1.1 horint.exe using *KSD*GB)

Contents of directory clim/skeletons:

ECMO_PST_8904250000_01200_HW
LAMF_KSD_0000000000_00000_GB

$RUNMDL/doc

the documentation files (see appendix B)

beschrijving
diagnose
diatree
identifikatie
installatie
read_me
resources
standaard
summary

$RUNMDL/exe

all (LAM) executables namely:

bounder.exe
daytsf.exe
hextobin.exe
lambd9.exe
ligrib.exe
listf.exe
mxhist.exe
mxtims.exe
petosi.exe
pplin.exe
prhist.exe
rwexa.exe
sigrib.exe
testprebd.exe

$RUNMDL/getbdrs

files concerning the production of boundary files (PQS_GB) (see chapter 5)

ECMOLAM000.desc
ECMOLAM006.desc
:
next forecastperiod (+6 hours)
:
ECMOLAM066.desc
ECMOLAM072.desc
MAKE_ECMO_PQS_GB
makeECMOLAMdesc.sc
makePQSGB.sc
make_ecmo_pqs_gb.f
make_ecmo_pqs_gb.mk
make_ecmo_pqs_gb.o

$RUNMDL/scripts all (LAM) unix scripts

afpat.sc
catchPQS.sc
control.sc
cycle.sc
daytsf.sc
destag.sc
elt.sc
getOC.sc
getPQS.sc
getpsts.sc
gettovs.sc
lam.sc
ligrib.sc
listf.sc
listgb.sc
marstoPQS.sc
mkrdirs.sc
mxhist.sc
prhist.sc
remold.sc
remove.sc
resumelam.sc
rwexa.sc
send2dbas.sc
setenvlamrun
sig2grib.sc
sweep.sc
swoop.sc
tailor.sc
test.sc
testmakepst.sc

$RUNMDL/source all (LAM) fortran sources
Some of these sources have the extension p, p4 or p8. This means that they have to be preprocessed (using ax_a9.f) before they can be compiled. *.p8 files have to be compiled with double precision. Extension p is equal to extension p4.

add10.p
ax_a9.f
bounder.p
chtogf.p
daytsf.f
dyns.p8
ecpp.p8
hextobin.p
knpp.p4
lamcom.p
ligrib.p
listf.p
main.p4
mast.p8
mlsurf.p
mxhist.p
mxtims.p
petosi.p
phys.p8
prebd.p
prhist.p
rwexa.f
sigrib.p
spec.p4
testprebd.p
tsf.p
tsfs.p8
utils.p
various.p

The subdirectories and contents of $RUNANA are:

$RUNANA/data containing eventslist.ch (input file for 1.6.1.21 events.exe)

$RUNANA/exe

OI executables

adanal.exe
errgro.exe
events.exe
expand.exe
initan.exe
maxmin.exe
obsfit.exe
postan.exe
preana.exe

$RUNANA/source

OI sources
The file comdeck.p has the extension p. This means that it has to be pre-processed (using ax_a9.f) before it can be compiled.
Most other files have the extension .C. This means that they are compressed using the, standard Unix, command compact. They can be decompressed using uncompact <filename>. This is done automatically by the installation scripts.

adanal.C
ax_a9.f
comdeck.p
errgro.C
events.C
expand.C
initan.C
lennrc.f
maxmin.C
obsfit.C
olympus.C
postan.C
preana.C
sendgrib.C
tovs.C
various.C

Appendix D Datafilenames

Datafilenames used inside the APL (automatic production line) are restricted to some conventions.
Format of a filename for the LAM:

LAMF_TTT_YYMMDDHHmm_HHHmm_FR

TTT	=	file type (see below)
YYMMDDHHmm	=	verification date/time
HHHmm	=	forecastperiod (hours/minutes) if FR = AB then HHHmm = 00000
FR	=	file format (first character) and representation type (second character)

Files from the ECMWF for the LAM (PQS or PST) start with ECMO (instead of LAMF).

-The filetypes are represented by three capital characters. The filenames used in this paper are:

O Observations:
OCB = Observation file (produced by getobs).

A or G Analysis:
AQS = Analysis (full fields) on 10 pressure levels.
GST = Uninitialised analysis on 11 model (σ) levels.
GSG = Initialised analysis on 11 model levels.
GQS = First guess on pressure levels.

B First guess errors:
BQS = first guess errors on 10 pressure levels.

F Forecasts:
FST = First guess (three hours forecast) on 11 model levels.
FMT = Forecast on mixed levels (model, pressure, mean sea level etc.), interpolated from 11 model levels.

K Climatology:
KLC = Climatology (analysis) on 10 pressure levels.
KQS = Climatology standard deviations on 10 pressure levels.
KVS = Vertical correlation matrix (10 pressure levels).
KSD = Temperature and moisture of soil, albedo, landsea mask and orography.

P Preprocessing:
PQS = ECMWF field on pressure levels.
PPS = As PQS, but on LAM grid.
PST = As PPS, but on 11 model levels.

T Time serie files:
TAA = Standard surface data and upper air data.
TBA = Standard surface data only.

MAP = satellite files

-The file format and form of representation are represented by two capital characters:

Indicator of fileformat (first character):
B = BUFR
G = GRIB
H = history-file
L = lineformat
M = MBW grid
O = observations
T = Time Serie File

Indicator form of representation (second character):

B = bits
C = characters
W = words
1 = descriptor file

Appendix E Installation procedure

Brief summary of the installation procedure for LAM and OI.

Suppose LAM and OI basis software are in the directories $SYSLAM and $SYSOI and suppose the directories for the 'static' software and data are:
$RUNMDL and $RUNOI (see chapter 2).
Installation is then done by the following commands:

```
For LAM:
% mkdir $RUNMDL
% mkdir instlam    (working dir for the installation)
% cd instlam
% cp $SYSLAM/iscripts/* .
% setenv EXPCODE op111 (for operational runs)
% setenv STATDAT /prod0/prodapl/prodhirl (for operat. runs)
% setenv DYNDAT /prod1/prodapl/prodhirl (for operational runs)
% installLAM.sc $RUNMDL
% compile          (on request if compilations are required)

For OI:
% mkdir $RUNANA
% mkdir instoi    (working dir for the installation)
% cd instoi
% cp $SYSOI/iscripts/* .
% installOI.sc $RUNANA
```

For more information see: Chapter 2 : Organisation and installation of the LAM and OI systems.

Appendix F Fields and level codes for postprocessing

Fields and level codes for postprocesing:
NOTE: in namelist &postin the ec-codes should be used.

THE FOLLOWING FIELD CODES ARE USED:-

ec-code	variabele	units	pres?	surf?	MSL?	sigm?	grib code
1	= GEOPOTENTIAL	m**2/s**2	x	x			102
2	= TEMPERATURE	K	x			x	104
3	= U-VELOCITY	m/s	x			x	123
4	= V-VELOCITY	m/s	x			x	124
5	= HUMIDITY MIX. RATIO	kg/kg	x			x	114
6	= PRESSURE	mbar		x	x		101
7	= VERTICAL VELOCITY	Pa/s	x				140
9	= PRECIP. WATER CONTENT	m					147
10	= VORTICITY	1/s	x				130
11	= SURFACE TEMPERATURE	K		x			104
12	= SOIL WETNESS	m		x			147
13	= SNOW DEPTH	m		x			151
14	= LARGE SCALE RAIN	m		x			150
15	= CONVECTIVE RAIN	m		x			150
16	= SNOW FALL	m		x			150
29	= RELATIVE HUMIDITY	fract	x	x			113
30	= D(PS)/DT	Pa/s		x			141
36	= CLOUD COVER	fract		x		x	179
37	= U AT 10 METRES	m/s		x			123
38	= V AT 10 METRES	m/s		x			124
39	= T AT 2 METRES	K		x			104
40	= TD AT 2 METRES	K		x			110

level codes:	pres? :	pressure(mbar)*10
	surf?:	-100
	MSL?:	-200
	sigm?:	highest model level = -NLEV (now:-11)
		:
		:
		lowest model level = -1

Appendix G Variables in LAM Time Series Files

Type TAA: single level elements and multi level elements
Type TBA: single level elements only

```
BUFR element descriptors used in FMLAM:     890322
--------------------------------------
Single level elements:
------------------------------------------------------------

        1   :LAND/SEA; LAND=0, SEA=1
     ISLE(1)=008012

        2   :SURFACE (GEOPOTENTIAL) HEIGHT  (M)
     ISLE(2)=010001

        3   :MEAN SEA LEVEL PRESSURE (PA)
     ISLE(3)=010051

        4   :PRECIPITATION  (M)
     ISLE(4)=013011

        5   :CONVECTIVE PRECIPITATION  (M)
     ISLE(5)=055132

        6   :SNOW FALL  (M)
     ISLE(6)=013012

        7   :SNOW HEIGHT  (M)
     ISLE(7)=013013

        8   :ROUGHNESS LENGTH  (M)
     ISLE(8)=055110

        9   :TOTAL CLOUD COVER  (FRACTION)
     ISLE(9)=055039

        10  :HIGH CLOUD COVER  (FRACTION)
     ISLE(10)=055031

        11  :MEDIUM CLOUD COVER  (FRACTION)
     ISLE(11)=055032

        12  :LOW CLOUD COVER  (FRACTION)
     ISLE(12)=055033

        13  :2M TEMPERATURE  (K)
     ISLE(13)=012004

        14  :2M DEW POINT  (K)
     ISLE(14)=012006

        15  :10M WIND DIRECTION  (DEG)
     ISLE(15)=011011
```

```
    16  :10M WIND SPEED  (M/S)
ISLE(16)=011012

    17  :SOLAR ANGLE  (DEG)
ISLE(17)=055240

    18  :NET SURFACE RADIATION (W/M**2)  UPW. POS.
ISLE(18)=055141

    19  :SURFACE PRESSURE  (PA)
ISLE(19)=055100

    20  :SURFACE TEMPERATURE  (K)
ISLE(20)=055120

    21  :SURFACE RELATIVE HUMIDITY  (%)
ISLE(21)=055130


-------------------------------------------------------------

Multi level elements on all model levels:
-------------------------------------------------------------


    1   :PRESSURE  (PA)
IMLE(1)=007004

    2   :TEMPERATURE  (K)
IMLE(2)=012001

    3   :POTENTIAL TEMPERATURE  (K)
IMLE(3)=055020

    4   :RELATIVE HUMIDITY  (%)
IMLE(4)=013003

    5   :WIND DIRECTION  (DEGREES TRUE)
IMLE(5)=011001

    6   :WIND SPEED  (M/S)
IMLE(6)=011002

    7   :CLOUDS  (FRACTION)
IMLE(7)=055030

    8   :EQUIVALENT POTENTIAL TEMPERATURE (THETA E)
IMLE(8)=055022

-------------------------------------------------------------
```

Appendix H Gridpoints for which Time Series Files are made

LAM has two kinds of time series files:
type A for surface and upper air data (APL file type *TAA*TW).
type B for surface data only (APL file type *TBA*TW).

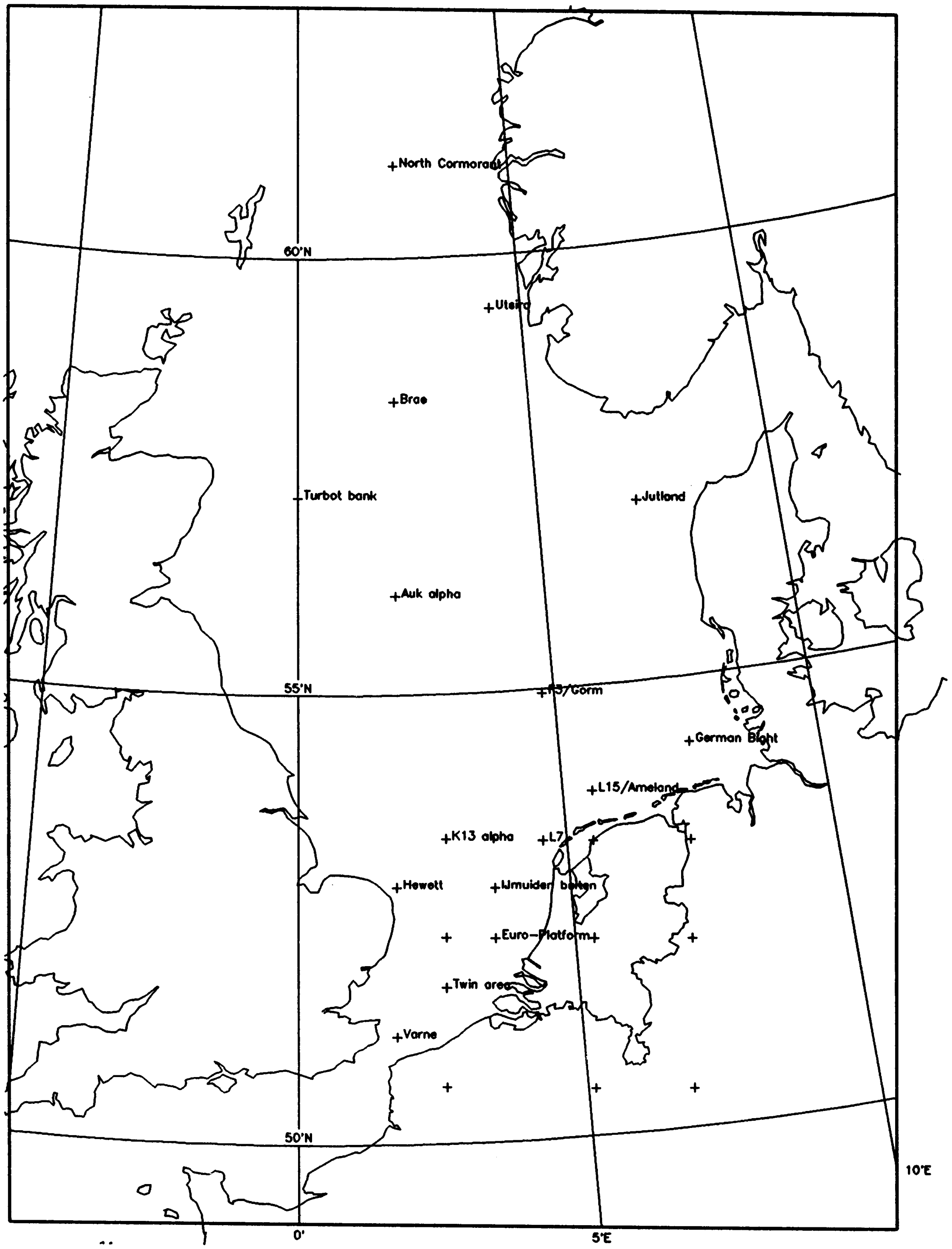
Gridpoint positions *TAA*TW file:

Nearby Station	Latitude	Longitude
North Cormorant/Brent	61,08	2.27
Utsira	59,38	4,32
Brae	58,33	2,09
Turbot bank	57,25	0
Jutland	57,04	7,08
Auk alpha	56,13	1,97
F3/Gorm	54,95	4,77
German Bight	54,25	7,51
L15/Ameland	53,81	5,56
K13 alpha	53,37	2,75
L7	53,30	4,57
	53,26	5,49
	53,16	7,30
Leman Area/Hewett	52,83	1,81
IJmuiden buiten	52,79	3,61
	52,27	2,67
Euro-Platform	52,24	3,56
de Bilt	52,17	5,34
	52,06	7,10
Twin area	51,72	2,64
Varne	51,19	1,73
	50,62	2,57
	50,52	5,13
	50,42	6,82

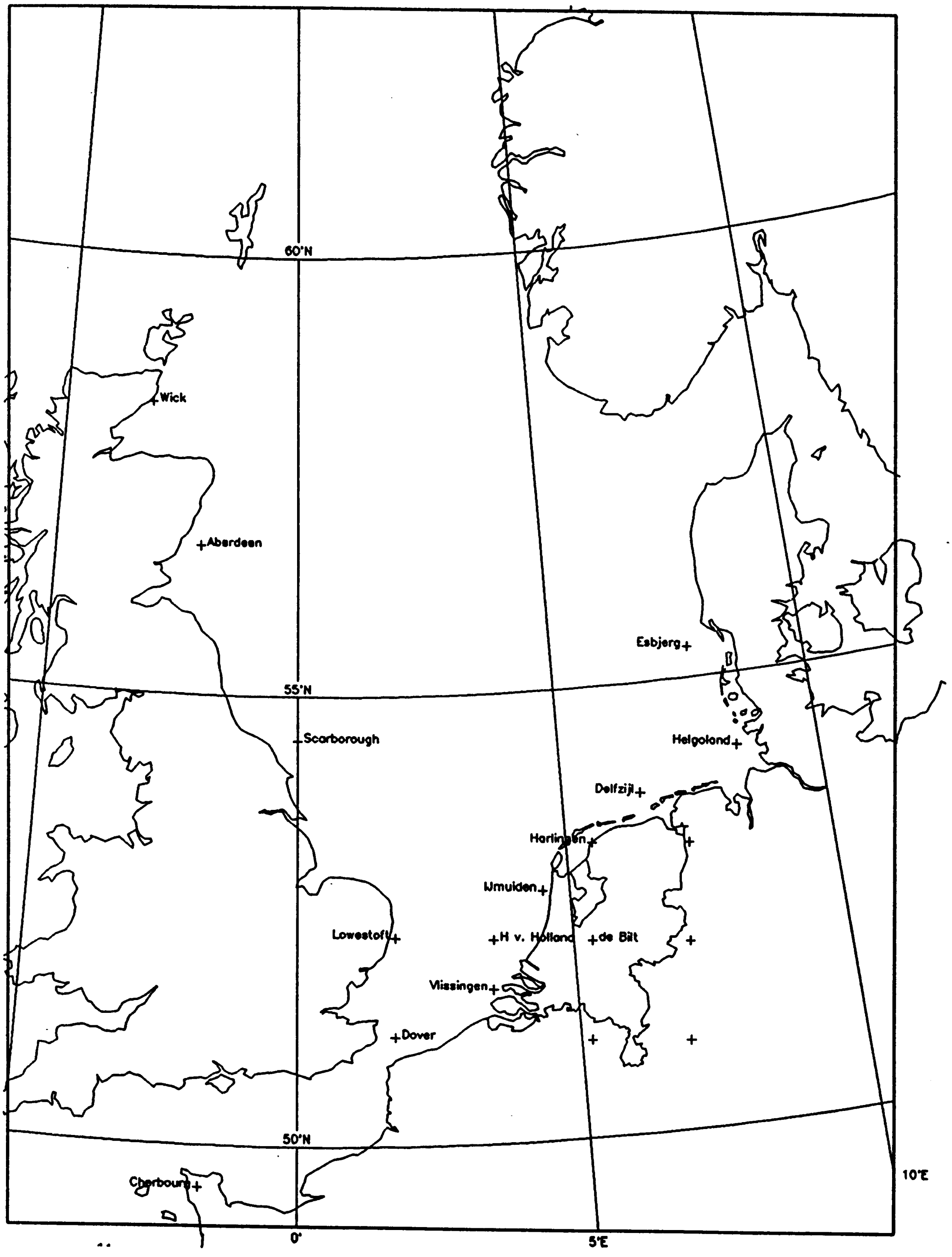
Gridpoint positions *TBA*TW file:

Nearby Station	Latitude	Longitude
Wick	58,31	-3,14
Aberdeen	56,68	-2,00
Scarborough	54,50	0
Lowestoft	52,29	1,78
Dover	51,19	1,73
Esbjerg	55,34	7,73
Helgoland	54,18	8,44
Delfzijl	53,76	6,49
Harlingen	53,26	5,49
IJmuiden	52,76	4,51
Hoek van Holland	52,24	3,56
Vlissingen	51,69	3,51
Cherbourg	49,54	-1,67
De Bilt	52,17	5,34
	51,07	5,19
	50,97	6,91

GRIDPOINT POSITIONS *TAA*TW FILES AND SOME NEARBY STATIONS



GRIDPOINT POSITIONS *TBA*TW FILES AND SOME NEARBY STATIONS



Appendix I List of postprocessed fields

Fields for all required forecast periods are written in the same file, which is labeled by the analysis time (APL file type 'FMT*AB').

In cycle.sc the array variable 'pphour' contains a list of forecast periods expressed in hours, for which output fields are required.
Array 'ppcont' contains a list of names referring to the required contents for the corresponding forecast period in 'pphour'. The required contents are defined by namelist 'POSTIN'.

```
pphour=(00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30)
ppcont=(AA AE AD AB AD AD AA AE AE AB AE AE AA AE AE AB AE AE AA AE AE AB AE AE AA AE AE AB AE AE AA)
```

The meaning of the relevant variables in &POSTIN is:

NMFD	NO. FIELDS AT MULTIPLE LEVELS
NFDML(10)	FIELD CODES (MULTIPLE LEVEL FIELDS)
NMLV	NO. OF LEVELS FOR MULT. LVL. FIELDS
NLVML(30)	LEVELS (MULTIPLE LEVEL FIELDS)
NSFD	NO. FIELDS AT SINGLE LEVEL
NFDSL(30)	FIELD CODES (SINGLE LEVEL FIELDS)
NLVSL(30)	LEVELS (SINGLE LEVEL FIELDS)
N2D	NO. 2-D FIELDS ON MODEL GRID
NGPCL(2,20)	FIELD CODE/LEVEL PAIRS (2-D FIELDS)

Appendix F gives a summary of the ec-field and level codes that can be used.

```
cat << end > AA
 &POSTIN
 NMFD=5,NFDML=1, 2, 3, 4, 29,
 NMLV=4,NLVML=5000, 7000, 8500, 10000,
 NSFD=4,
 NFDSL=6,    7,    3,    4,
 NLVSL=-200, 5000, 2500, 2500,
 N2D=25,NGPCL= 6,-100, 11,-100, 14,-100, 15,-100, 29,-100,
        37,-100, 38,-100, 39,-100, 40,-100,
        2,-1, 2,-2, 2,-3, 2,-4, 2,-5,
        5,-1, 5,-2, 5,-3, 5,-4, 5,-5,
        3,-1, 3,-2, 3,-3,
        4,-1, 4,-2, 4,-3,
 &END
end
```

```
cat << end > AB
 &POSTIN
 NMFD=0,
 NMLV=0,
 NSFD=2,
 NFDSL=6,    2,
 NLVSL=-200, 8500,
 N2D=10,
 NGPCL= 14,-100, 15,-100, 37,-100,  38,-100,
         3,-1, 3,-2, 3,-3,
         4,-1, 4,-2, 4,-3,
 &END
end


cat << end > AE
 &POSTIN
 NMFD=0,
 NMLV=0,
 NSFD=1,
 NFDSL=6,
 NLVSL=-200,
 N2D=2,
 NGPCL= 37,-100,  38,-100,
 &END
end


cat << end > AD
 &POSTIN
 NMFD=0,
 NMLV=0,
 NSFD=1,
 NFDSL=6,
 NLVSL=-200,
 N2D=4,
 NGPCL= 37,-100,  38,-100,
      3, -1,    4, -1,
 &END
end
```

References

[1] Louis, J.F. et al, 1982: ECMWF Forecast Model Documentation Manual, Volumes 1 & 2.

[2] Cats, G.J., 1984: A scheme for mass and wind analysis on a limited area using multivariate threedimensional optimum interpolation: scientific documentation and first evaluation.
KNMI, T.R. 46

[3] Bijlsma, S.J., Hafkenscheid, L.M., 1986: Initialisation of a limited area model: a comparison between the nonlinear normal mode and bounded derivative methods.
Monthly weather review, Vol. 114, no. 8 (1986);
p. 1445-1455