IFS DOCUMENTATION

PART VI: TECHNICAL AND COMPUTATIONAL PROCEDURES (CY25R1)

(Operational implementation 9 April 2002)

Edited by Peter W. White

(Text written and updated by members of Météo-France and the ECMWF Research Department)

Table of contents

Copyright

Part VI: TECHNICAL AND COMPUTATIONAL PROCEDURES

CHAPTER 1 Technical overview

Table of contents

1.1 INTRODUCTION

This chapter describes the high level technical structure of IFS, how it can be supplied with initial data, how the execution is controlled, and how output fields can be generated. The chapter is divided into the following sections:

1) Configurations
2) Initial data
3) Command line options
4) Namelist control
5) Post processing
6) Restart capability
7) Mass conservation

1.2 CONFIGURATIONS

TABLE 1.1

Value	Control routine	Model description
0-99	CNT1	Integration
100-199	CVA1	Variational analysis
200-299	CNT1	2-D integration
300-399	COV1	Kalman filter or predictability model
400-499	CAD1	Test of the adjoint
500-599	CTL1	Test of the tangent linear model
600-699	CUN1	Eigenvalue/vector solvers for unstable modes
700-799	CAN1	CANARI Optimal interpolation
800-899	CGR1	Sensitivity
900-999	CORMASS, INCLI0, NOEUD, EN0ED, CPREP1, CPREP2, CPREP3, CPREP5	Preparation of initial conditions/interpolations

The IFS contains many different functions within a single high level program structure. For any single execution of the program, the function is selected by means of a ***configuration*** parameter. The value of this parameter may be supplied on the IFS execution command line, or by using the namelist NCONF parameter (see the namelist NAMCT0 documentation).

The values of NCONF are divided into ranges which are handled by different high level control routines as indicated in Table 1.1:

Within these broad bands, certain values of NCONF have been defined specifically. In Table 1.2, status=O implies the configuration is in operational use at ECMWF and status=R means that it is used for research purposes.

TABLE 1.2

Value	Model description	Status
1	3-D primitive equation model	O
2	NCONF=1 & comparison with observations (LOBSC1=.TRUE.)	O
99	Dummy GPC (Identity)	
101	4-D var with 3-D P.E. model	
111	4-D var with 3-D P.E. tangent linear model	
121	4-D var with shallow water model	
122	4-D var with vorticity equation model	

TABLE 1.2

Value	Model description	Status
123	4-D var with linear gravity wave model	
131	Incremental 3-D/4-D var	O
151	3-D var	
199	4-D var with dummy GPC	
201	Shallow water model	R
202	Vorticity equation model	
203	Linear gravity wave model	
301	Kalman filter with 3-D P.E. model	
321	Kalman filter with shallow water model	
322	Kalman filter with vorticity equation model	
323	Kalman filter with linear gravity wave model	
351	Local predictability with 3-D P.E. model	
371	Local predictability with shallow water model	
372	Local predictability with vorticity equation model	
373	Local predictability with linear gravity wave model	
399	Kalman filter with dummy GPC	
401	Test of adjoint with 3-D P.E. model	R
421	Test of adjoint with shallow water model	R
422	Test of adjoint with vorticity equation model	
423	Test of adjoint with linear gravity wave model	
499	Test of adjoint with dummy GPC	
501	Test of tangent linear with 3-D P.E. model	R
521	Test of tangent linear with shallow water model	R
522	Test of tangent linear with vorticity equation model	
523	Test of tangent linear with linear gravity wave model	
599	Test of tangent linear with dummy GPC	
601	Eigenvalue/vector solver (singular vector comp.)	O
701	Optimal interpolation with CANARI	
801	Sensitivity with 3-D P.E. model	O
821	Sensitivity with shallow water model	
923	Initialisation of climatological fields	
926	Change of geometry	
951	Difference between 2 model states (CPREP2)	
952	Compute wind and gridpoint fields (CPREP3)	
953	Compute gridpoint gradient fields (CPREPAD)	

1.3 CONFIGURATION CONTROL

Within subroutines STEPO, STEPOAD and STEPOTL, extensive use is made of a character string CDCONF for controlling the logic flow of transformations between spectral and grid-point space. CDCONF is a 9 character variable where each character controls a specific sub-area within IFS as indicated in Table 1.3

TABLE 1.3

Character position	Sub-area	Description
1	IOPACK	IO handling
2	LTINV	Inverse Legendre transform
3	FTINV	Inverse Fourier transform
4	CPG	Grid point computations
5	POS	Post processing
6	OBS	Comparison with observations
7	FTDIR	Direct Fourier transform
8	LTDIR	Direct Legendre transform
9	SPC	Spectral space computations

The characters used to represent each function are specified as follows, where a 'null' or 'blank' character means ***do nothing***:

iopack (CDCONF(1:1)):

TABLE 1.4

Character	Description
A	Write out model level post-processed data
B	Retrieve trajectory information
C	Write out pressure level post-processed data
I	Store/write out increment (incremental 3D/4D-var)
J	Read increment (incremental 3D/4D-var)
R	Read restart file
T	Store trajectory
L	Lagged model level post-processing
V	Read in the inputs for sensitivity job
F	"A" + "R"
E	FullPos
U	FullPos
Y	FullPos

TABLE 1.4

Character	Description
M	FullPos
Z	FullPos

Inverse Legendre Transform (CDCONF(2:2)):

TABLE 1.5 SPECTRAL DATA

Derivatives	SPA3		SPA5		SPA7	
	Yes	No	Yes	No	Yes	No
Fourier data T0	A	G	B	H	C	I
Fourier data T5	D	J	E	K	F	L

CDCONF(2:2) = P is used by FullPos

Inverse Fourier Transform(CDCONF(3:3)):

TABLE 1.6 GRIDPOINT DATA

Derivatives	T0		T5		T1	
	Yes	No	Yes	No	Yes	No
Fourier data T0	A	B				I
Fourier data T5			C	D		

CDCONF(3:3) = P is used by FullPos

Grid-point calculations for the model (CDCONF(4:4)):

TABLE 1.7

Character	Description
A	"Normal "timestep
B	Additional computations for post-processing surface fields
E or F	Adiabatic NNMI iteration
M or X	NNMI iteration or initial fluxes

Post-processing (CDCONF(5:5)): —If vertical part of FullPos or "old style" p.p. (LLVFP=true)

:

TABLE 1.8

Character	Description
A	Pressure level post-processing
H	Height (above orography) level post-processing
T	Potential temperature level post-processing
V	Potential vorticity level post-processing
M	Model level post-processing
S	Eta level post-processing
L	End of vertical post-processing

Gridpoint computations for analyses (CDCONF(6:6)):

TABLE 1.9

Character	Description
A	Add squares of gridpoint values (analyses error calc.)
B	Subtract squares of point values (analyses error calc.)
Y	Modifies the background errors to have a prescribed global mean profile and (optionally) to be separable
Z	Generate background errors of humidity
G or W	Normalization by standard deviations of background error
X	Multiplication by standard deviations of background error (inverse of above)
I	Grid point calculations for CANARI
C or V	Computation of observation equivalents (GOM-arrays)

Direct Fourier transform (CDCONF(7:7)):

TABLE 1.10

Character	Description
A	Standard transform
B	Pressure level post-processing
C	Model level post-processing step
P	FullPos

Direct Legendre Transform (CDCONF(8:8)):

TABLE 1.11

Character	Description
A	Standard transform
B	Pressure level post-processing
C	Model level post-processing step
P	FullPos
T	Tendencies (result in SPT.. arrays)
G	Like A but goes from vorticity and divergence in Fourier space to spectral space, instead of starting from wind components

Spectral Space Computations (CDCONF(9:9)::

TABLE 1.12

Character	Description
P	Filtering of Full-Pos fields
A	Semi-implicit + horisontal diffusion
F	Filtering of spectral fields
I	Only semi-implicit (for NMI)

1.4 INITIAL DATA

The starting conditions are supplied to the IFS in a number of files which follow a specific naming convention. The file names used and their contents vary according to the model configuration being run. In the following descriptions an ***experiment identifier*** (xxid) consisting of any 4 alphanumeric characters is used. Note that the ***Arpege*** version of the IFS, selected by setting LECMWF=false in namelist NAMCT0, uses different file names and file formats and is not documented here.

1.4.1 Forecast model, NCONF=1

The initial fields are supplied in GRIB format and contained in the files ICMSHxxidINIT, ICMGGxxidINIT (and optionally also ICMGGxxidINIUA and ICMCLxxidINIT. The following tables indicates what fields are required, where the number of model levels is NFLEV. Some fields are conditional on the setting of model switches which are described in more detail in the section ***Namelist*** Control

1.4.1 (a) ICMSHxxidINIT. This contains upper-air spectral format fields on model levels.

.

TABLE 1.13

ECMWF GRIB code	IFS variable	Description	Levels	Condition
129	NGRBZ	Geopotential	1	not LGPOROG
152	NGRBLNSP	Log surface pressure	1	
130	NGRBT	Temperature	NFLEV	
133	NGRBQ	Specific humidity	NFLEV	LSPQ or LGPQ and not LGPQIN
138	NGRBVO	Vorticity (relative)	NFLEV	
155	NGRBD	Divergence	NFLEV	

Notes:

- LGPOROG=false (default) if input orography is spectral
- LSPQ=true to keep humidity as a spectral variable (default=false)
- LGPQ=true to keep humidity as a grid-point variable (default=true)
- LGPQIN=true if humidity is input in grid-point space

1.4.1 (b) ICMGGxxidINIT. This contains surface fields on the model Gaussian grid.

TABLE 1.14

ECMWF GRIB code	IFS variable	Description	Condition
129	NGRBZ	Geopotential (at the surface orography)	If not provided as spectral field
139	NGRBSTL1	Surface temperature level 1	
140	NGRBSWL1	Soil wetness level 1	
141	NGRBSD	Snow depth	
160	NGRBSDOR	Standard deviation of orography	
161	NGRBISOR	Anisotropy of subgrid scale orography	
162	NGRBANOR	Angle of subgrid scale orography	
163	NGRBSLOR	Slope of subgrid scale orography	
170	NGRBSTL2	Soil temperature level 2	
171	NGRBSWL2	Soil wetness level 2	

TABLE 1.14

ECMWF GRIB code	IFS variable	Description	Condition
172	NGRBLSM	Land/sea mask	
173	NGRBSR	Surface roughness	
174	NGRBAL	Albedo	
183	NGRBSTL3	Soil temperature level 3	
184	NGRBSWL3	Soil wetness level 3	
198	NGRBSRC	Skin reservoir content	
199	NGRBVEG	Percentage of vegetation	
148	NGRBCHAR	Charnock parameter	LWCOU and LWCOU2W (coupled wave model)
233	NGRBASQ	Apparent surface humiidity	
234	NGRBLSRH	Logarithm of surface roughness length for heat	
235	NGRBSKT	Skin temperature	
236	NGRBSTL4	Soil temperature level 4	
237	NGRBSWL4	Soil wetness level 4	

1.4.1 (c) ICMGGxxidINIUA. This contains upper air fields in grid point space.

TABLE 1.15

ECMWF GRIB code	IFS variable	Description	Levels	Condition
133	NGRBQ	Specific humidity	NFLEV	LGPQIN=true
246	NGRBCLWC	Cloud liquid water content	NFLEV	LCLDPIN=true
247	NGRBCIWC	Cloud ice water content	NFLEV	LCLDPIN=true
248	NGRBCC	Cloud cover	NFLEV	LCLDPIN=true
203	NGRBO3	Ozone mixing ratio (EC prognostic ozone)	NFLEV	LGPO3=true

Notes:

- Specific humidity can be input in spectral form (LGPQIN=false) even when LGPQ=true
- If the cloud parameters (246,247,248) are not input (LCLDPIN=false), they are initialised to zero
- LGPO3=true for ozone as a prognostic variable (grid-point field).

1.4.1 (d) ICMCLxxidINIT. Climate fields.

Contains surface fields, in the model Gaussian grid, to be used in 'perfect surface' long integrations. In such integrations the surface conditions subject to seasonal variation (which would normally be defined by the data assimilation and kept constant during the forecast) are changed regularly during the integration, based on the values

contained in the file.

Note that the fields below have to follow a pattern, otherwise the model fails in the first time step. They should be in ascending time order, and the time spanned by them should be large enough to cover the model integration period. Furthermore, they should come at regular intervals, either every *n*th day (LMCCIEC=.FALSE.), or every month (LMCCIEC=.TRUE., see NAMCC below).

TABLE 1.16

ECMWF grib code	IFS variable	Description	Condition
139	NCLIGC(1)	Surface temperature layer 1	LMCCEC .and. .NOT.LMCC04
174	NCLIGC(2)	Albedo	LMCCEC .and. .NOT.LMCC04

1.5 COMMAND LINE OPTION

The primary way to control options within IFS is the ***namelist*** input file. However, since there are a very large number of options, it is convenient to be able to specify certain standard configurations in a simple way. This is achieved by supplying a small number of ***Unix*** style flags on the command line:

e.g. MASTER -e abcd -v ecmwf -t 900 -f d10 -a sli

The available flags and their mappings on to namelist variables are as follows:

TABLE 1.17

Option	Namelist variable	Description
-c	NCONF	Job configuration
-v	LECMWF	Model version: **ecmwf** or **meteo**
-e	CNMEXP	Experiment identifier (no more than 4 characters)
-t	TSTEP	Time step (seconds) default set according to model resolution and advection scheme
-f	NSTOP	Forecast length: **dxxxxx** - run for xxxxx days **hxxxxx** - xxxxx hours txxxxx - xxxxx timesteps
-a	LSLAG LVENIN	Advection scheme - **eul** : Eulerian **sli** : interpolating semi-Lagrangian **slni** : non-interpolating in the vertical, semi-Lagrangian
-m	LUELAM	Model type: **arpifs**: ARPEGE/IFS **aladin** : ALADIN

Notes:

- If the forecast length is specified in units other than timesteps, then the value of the timestep must be given.
- Command line arguments override any namelist switches that are set.
- Either both options -v and -e must be used together, or neither must be used.

1.6 NAMELIST CONTROL

Namelist input is provided in a text file **fort.4**. Within this file, the namelists can be in any order. The file is read multiple times by IFS to extract the namelist parameters in the order that the IFS code reads them. The general format is:

```
&NAME1
keyword=value,
..
/
&NAME2
/
```

All namelists must always be present in **fort.4**. It is not necessary, however, for a namelist to have any contents (see example of NAME2 above).

1.6.1 Index of Namelists

TABLE 1.18

Name	Description	Read in subroutine
nacobs	CANARI	defrun
nactdo	CANARI	not used
nactex	CANARI	canali
nacveg	CANARI	canali
naephy	ECMWF physics	su0phy
naerad	ECMWF radiation	suecrad
naimpo	CANARI	canali
nalori	CANARI	canali
nam926	Change of orography	su926
nam_distributed_vectors	Initialize chunksize for Distributed Vectors	sumpini
namafn	Full-Pos	suafn
namana	CANARI	canali:defrun
namcfu	Flux accumulation control	sucfu
namchk	Gridpoint evolution diagnostics	suechk
namcla		incli0
namclb		incli0

TABLE 1.18

Name	Description	Read in subroutine
namclc		incli0
namcli		incli0
namclo		incli0
namclp		incli0
namclr		incli0
namcls		incli0
namclt		incli0
namclw		incli0
namcok	CANARI	canali
namcos	General cost function control	not used
namct0	Control parameters, constant during model run	suct0:sumpini
namct1	Overiding control switches	su1yom
namcva	General variational analysis control	sualctv
namddh	Diagnostic (horizontal domain)	sunddh
namdfi	Digital filtering control	suedfi
namdif	Difference two model states	sudif
namdim	Dimension/truncation	sudim
namdmsp	Satellite data descriptor	getsatid
namdphy	Physics dimension	sudim
namdyn	Dynamics and hyperdiffusion	sudyn
namfft	FFT	sufft
namfpc	Full-Pos	sufpc
namfpd	Full-Pos	sufpd
namfpdyh	Full-Pos	sufpdyn
namfpdyp	Full-Pos	sufpdyn
namfpdys	Full-Pos	sufpdyn
namfpdyt	Full-Pos	sufpdyn
namfpdyv	Full-Pos	sufpdyn
namfpf	Full-Pos	sufpf
namfpg	Full-Pos	sufpg1
namfpios	Full-Pos	sufpios
namfpphy	Full-Pos	sufpphy
namfpsc2	Full-Pos	sufpsc2
namgem	Transformed sphere (geometry/coordinate definition)	sugem1a:inclib
namgms	Satellite data descriptor	getsatid
namgoes	Satellite data descriptor	getsatid

TABLE 1.18

Name	Description	Read in subroutine
namgrib	GRIB coding descriptor	sugrib
namini	Overriding switches for initialization	sueini
namiomi	Minimisation i/o scheme	suiomi
namios	I/O control	suios
namjg	Assimilation, first guess constraint	sujb
namjo	Jo control	defrun
namkal	Kalman filter	sufkal
namkap	Kalman filter	sufkap
namlcz	Lanczos eigensystem	sulcz
namleg	Legendre polynomials	suleg
namlfi	LFI description (Arpege)	sulfi
nammars	Conversion from grib to ARPEGE file	cprep1
nammcc	Climate version	sumcc:sudim
nammeteosat	Satellite data descriptor	getsatid
nammtt	Cray specific,diagnostic	sumtth
nammul	Multi-tasking control	sumul
namnmi	Normal mode initialisation	sunmi
namnud	Nudging	sunud
namobs	observation control	defrun
namoph	Permanent file information	suoph
nampar0	Parallel version control	sumpini
nampar1	Parallel version control	sump0
namphy0	Arpege atmospheric physical parameters	suphy0
namphy1	Arpege ground physics parameters	suphy1
namphy2	Arpege vertical physics definition	suphy2
namphy3	Arpege radiation physical constants	suphy3
namphyds	Physics fields setup	suphyds
namphy		not used
nampm3d	3D Local Predictability Model	supm3a:canali
namppc	Post-processing control	supp
nampre	CANARI	canami
namrad15	Arpege climate version of ECMWF radiation	suecrad15
namrcf	Restart control file	reresf:wrresf
namres	Restart time parameters	sures
namrgri	Reduced grid description	surgri
namrinc	Incremental variational description	surinc

TABLE 1.18

Name	Description	Read in subroutine
namrip	Real time parameters	surip
namscc	observation screening control	defrun
namsens	Sensitivity job	suvar
namsimphl	Arpege linear physics parameterization	su0phy
namskf	Simplified Kalman Filter	suskf
namstoph	Parameterisation top limits / mesospheric drag parameters	surand1
namtoph	Mesospheric drag parameterization (Arpege)	sutoph
namtovs	Satellite data descriptor	getsatid
namtrajp	ECMWF linear physics	su0phy
namvar	Variational assimilation	suvar
namvdoz	Arpege physics	suphy1
namvfp	Variable LARCHFP	suvar
namvrtl	Switches for variational assimilation	suvar
namvv1	vertical co-ordinate descriptor	suvert
namvwrk	IO-scheme for trajectory	suvwrk
namxfu	Instantaneous flux control	suxfu
namzdi	Zonal diagnostic	suzdi
naphlc	Switches for simple physics	su0phy

The following tables describe namelist parameters which can be controlled by utilising the above namelists. 'type' indicates the Fortran variable type where L=logical, I=integer,R=real,C=character. Parentheses () indicate that the variable is an array. Logical variables (L) are used as switches to control a process.

TABLE 1.19 NAEPHY: ECMWF PHYSICS

Name	Type	Description	Default
LAGPHY	L	Call physics package in lagged mode	
LEAERO	L	Climatological aerosols	
LECOND	L	Large scale condensation	
LECUMF	L	Mass-flux convection scheme	
LEDCLD	L	Diagnostic cloud scheme	
LEDYNT	L	Use dynamical tendencies as input to physics	
LEDBUG	L	Debugging prints	
LEEVAP	L	Evaporation of precipitation	
LEGWDG	L	Gravity wave drag	
LEOZOC	L	Climatological ozone	

TABLE 1.19 NAEPHY: ECMWF PHYSICS

Name	Type	Description	Default
LEO3CH	L	ozone chemistry	
LEPCLD	L	Prognostic cloud scheme	
LEPHYS	L	ECMWF physics	
LEQNGT	L	Negative humidity fixer	
LERADC	L	Interpolation scheme for radiation	
LERADI	L	Radiation scheme	
LERADS	L	Interactive surface radiative properties	
LESHCV	L	Shallow convection	
LESICE	L	Interactive sea ice temperature	
LESURF	L	Interactive surface processes	
LEVDIF	L	Vertical diffusion	
LEZERO	L	Pure T-DT inputs within all physics routines	
LWCOU	L	Coupled wave model	
LWCOU2W	L	Coupled wave model with 2-way interaction	

TABLE 1.20 NAERAD: ECMWF RADIATION

Name	Type	Description	Default
LECSRAD	L	Clear-sky radiation diagnostics	FALSE
LEPO3RA	L	Pass the prognostic ozone to radiation code	FALSE
LERAD6H	L	Radiative computations every 1 hour during the first 6 hours	TRUE
LERADHS	L	Radiation is computedon a coarser sampled grid	TRUE
LLGEOSE	L	Simulate LW window channel radiance in GOES-E geometry	FALSE
LLGEOSW	L	Simulate LW window channel radiance in GOES-W geometry	FALSE
LLGMS	L	Simulate LW window channel radiance in GMS geometry	FALSE
LLINDSA	L	Simulate LW window channel radiance in INSAT geometry	FALSE
LLMTO	L	Simulate LW window channel radiance in METEOSAT geometry	FALSE
LOIEBCU	L	Ice cloud optical properties from Ebert–Curry	TRUE
LOIOULI	L	Ice cloud optical properties from Ou-Liou	FALSE
LOISUSH	L	Ice cloud optical properties from Sun-Shine	FALSE
LONEWSW	L	New style of cloud optical properties	TRUE
LOWCAS	L	Water cloud optical properties from Slingo	FALSE

TABLE 1.20 NAERAD: ECMWF RADIATION

Name	Type	Description	Default
LOWCYF	L	Water cloud optical properties from Fouquart	TRUE
LRADLB	L	Radiation coarse grid to be load balanced (when NRINT > 1)	TRUE
IGEO	I	Number of geostationary satellites to be simulated	0
NAER	I	Aerosol radiative effects (on=1, off=0)	1
NCSRADF	I	If LECSRAD true, accumulated or instantaneous flux (1 or 2)	1
NLW	I	Number of LW spectral intervals	6
NMODE	I	Radiation code configuration: flux=0, radiance=1	0
NOVLP	I	Cloud overlap index: 1=max-ran, 2=maximum, 3=random	1
NOZOCL	I	Ozone climatotogy 0=Geleyn, 1=Fortuin-Langematz	1
NRADC2F	I	Coarse-to-fine interpolation	2
NRADF2C	I	Fine-to-coarse interpolation	2
NRADFR	I	Frequency of full radiation, < in hours, > in time-steps	-3
NRADPFR	I	Ffrequency of radiation diagnostic listing prints	0
NRADPLA	I	Listing print output every NRADPLA latitude lines	15
NRINT	I	Interpolation distance (in points along a latitude line)	4
NRPROMA	I	Vector length for radiation calculations	
NSW	I	Number of SW spectral intervals	2

TABLE 1.21 NAMCT0: CONTROL PARAMETERS, CONSTANT DURING MODEL RUN

Name	Type	Description	Default
CFCLASS	C*2	Class for use by fields database	FALSE
CFDIRLST	C*120	Path of postprocessing listing file	TRUE
CFPATH	C*200	Prefix for trajectory spectral data filename	
CFPNCF	C*6	Name of Full-POS control file (configuration 927)	
CNDISPP	C*120	Directory for display files ('ifs.stat' and 'ifs.disp')	
CNMEXP	C*16	Name of the experiment	
CNPPATH	C*120	Directory of postprocessing namelist files	
CTYPE	C*2	Type for use by fields database	
LADJORO	L	Initialise increments	

TABLE 1.21 NAMCT0: CONTROL PARAMETERS, CONSTANT DURING MODEL RUN

Name	Type	Description	Default
LAGHIS	L	Lagged physics output on history files	
LAPRXPK	L	Approximate definition of full levels	
LARPEGEF	L	Use ARPEGE files	FALSE
LCANARI	L	Controls CANARI optimal interpolation	
LCOORD	L	Model coordinates written on trajectory files	
LCVROSS	L	Restirct contributions to gradient to that of Rossby modes	
LECMWF	L	Use model version of ECMWF	TRUE
LELAM	L	Controls ALADIN geometry	
LFBDAP	L	Diagnostics written on trajectory files	
LFDBOP	L	Write post-processed output fields to fields database	TRUE
LFPOS	L	Use Full-POS	
LFROG	L	Use leapfrog scheme, otherwise Matsuno	
LGRAV	L	Control of gravity waves included	
LGRBOP	L	Output in GRIB	
LGUESS	L	Term of first guess included	
LHDCDIV	L		
LHDCHV	L		
LHDCSV	L		
LHDCVOR	L		
LIOCFU	L	Work file for cumulative diagnostics	
LIODTND	L	Work file for diabatic tendencies in NMI	
LIOFOU1	L	Work file for first/second scan Fourier data	
LIOFOU15	L	Work file for first/second scan Fourier data in trajectory	
LIOFOU2	L	Work file for second/third scan Fourier data	
LIOGAUX	L	Work file for unfitted full-POS data	
LIOGP5	L	Work file for surface fields in trajectory	
LIOGPP	L	Work file for surface fields	
LIOHOU	L	Work file for explicit NMI	
LIOLPOL	L	Work file for Legendre polynomials	
LIOPIN	L	Work file for implicit NMI help arrays	
LIOTMDT	L	Work file for t-dt gridpoint data	
LIOXFU	L	Work file for instantaneous diagnostics	
LMDYPP	L	Modern dynamical meteorology post-processing	
LMGMEM	L	Use the memory manager	

TABLE 1.21 NAMCT0: CONTROL PARAMETERS, CONSTANT DURING MODEL RUN

Name	Type	Description	Default
LMLTTH	L	CRAY multitasking trace history	
LMLTSK	L	Run in multi-tasking mode	
LMPLOT	L	Plotting requested	
LMULC2	L	Cray macrotasking alternatives	
LNOR	L	Change of orography	
LOBS	L	Term of observations included	
LOBSC1	L	Term of observations included in configuration 1	
LOBSIO	L	I/O scheme for observations	
LOBSREF	L	Comparison to observations for the trajectory (NCONF=131)	
LREGETA	L	Definition of interlayer	
LRPOLE	L	Point at the poles	
LSEPPH	L	Physics to be done in separate step	
LSIDG	L	Semi-implicit-scheme with reduced divergence	
LSIMOB	L	Simulated observations	
LSIMUL	L	Subsurface data rejection and check on orography	
LSLAG	L	Semi-Lagrangian scheme	
LSMSSIG	L	Send signals to supervisor (SMS)	
LSPRT	L	Virtual temp. as spectral variable	
LSTKST	L	Cray stack statistics	
LTRAJ	L	Save trajectory on file	
LTRAJGG	L	Grid-point trajectory not written but taken from SUGRIDF	
LVENIN	L	Non-interpolating vertical semi-Lagrangian	
N2DINI	I	2D initial data control	
N3DINI	I	3D initial data control	
NCNTVAR	I	Choice of control variables	
NCONF	I	Model configuration	
NCYCLE	I	Experiment number	
NDHFDTS	I	Write out steps for limited area DDH diagnostics	
NDHFGTS	I	Write out steps for global DDH diagnostics	
NDHFZTS	I	Write out steps for zonal mean DDH diagnostics	
NDHPTS	I	Times of printed DDH output	
NFFTCR	I	Cray specific interfaces (XMP or CRAY-2)	
NFRCO	I	Frequency of coupled fields (time-steps)	
NFRDHFD	I	Frequency of limited area diagnostics (in steps)	

TABLE 1.21 NAMCT0: CONTROL PARAMETERS, CONSTANT DURING MODEL RUN

Name	Type	Description	Default
NFRDHFZ	I	Write-up of zonal DDH (in steps)	
NFRDHP	I	Frequency of DDH diagnostics printing	
NFRGDI	I	Frequency of grid-point space diagnostics	
NFRHIS	I	Frequency of history write_ups	
NFRPLT	I	Plotting frequency	
NFRPOS	I	Frequency of post-processing events	
NFRSDI	I	Frequency of spectral diagnostics (in steps)	
NGDITS	I	Grid point diagnostics steps	
NMTRA	I	Memory allowed for the descent algorithm	
NQUAD	I	Quadrature (1=Gauss, 2=Lobatto)	
NSDITS	I	Spectral diagnostics steps	
NSPACE	I	Size of space managed by memory manager	
NSPPR	I	Spectrum printed in spnorm	
NSTART	I	First timestep of model	
NSTOP	I	Forecast/model run timespan	
NTASKS	I	Number of tasks used when multitasking	
NTSPL	I	Time-splits (1 if leap-frog, 2 if Matsuno)	
LRFILAF	L	Catalog file LFI	
N2DINI	I	2D initial data control	
N3DINI	I	3D initial data control	
NCNTVAR	I	Choice of control variables	
NCONF	I	Model configuration	
NCYCLE	I	Experiment number	
NDHFDTS	I	Write out steps for limited area DDH diagnostics	
NDHFGTS	I	Write out steps for global DDH diagnostics	
NDHFZTS	I	Write out steps for zonal mean DDH diagnostics	
NDHPTS	I	Times of printed DDH output	
NFFTCR	I	Cray specific interfaces (XMP or CRAY-2)	
NFRCO	I	Frequency of coupled fields (time-steps)	
NFRDHFD	I	Frequency of limited area diagnostics (in steps)	
NFRDHFZ	I	Write-up of zonal DDH (in steps)	
NFRDHP	I	Frequency of DDH diagnostics printing	
NFRGDI	I	Frequency of grid-point space diagnostics	
NFRHIS	I	Frequency of history write_ups	
NFRPLT	I	Plotting frequency	
NFRPOS	I	Frequency of post-processing events	

TABLE 1.21 NAMCT0: CONTROL PARAMETERS, CONSTANT DURING MODEL RUN

Name	Type	Description	Default
NFRSDI	I	Frequency of spectral diagnostics (in steps)	
NGDITS	I	Grid point diagnostics steps	
NMTRA	I	Memory allowed for the descent algorithm	
NQUAD	I	Quadrature (1=Gauss, 2=Lobatto)	
NSDITS	I	Spectral diagnostics steps	
NSPACE	I	Size of space managed by memory manager	
NSPPR	I	Spectrum printed in spnorm	
NSTART	I	First timestep of model	
NSTOP	I	Forecast/model run timespan	
NTASKS	I	Number of tasks used when multitasking	
NTSPL	I	Time-splits (1 if leap-frog, 2 if Matsuno)	

TABLE 1.22 NAMDIM:DIMENSION/TRUNCATION PARAMETERS

Name	Type	Description	Default
NDGL	I	Number of Gaussian latitudes	
NDLON	I	Number of points in a latitude row	
NFLEV	I	Number of vertical levels	
NFTHER	I	Number of thermodynamics variables	
NFPASS	I	Number of passive scalar variables	
NFAUX	I	Number of auxillary variables in t+dt array	
NFD2D	I	Number of 2d fields in the dynamics	
NFC2D	I	Number of 2d fields in the boundaries	
NPMAX	I	Post-processing truncation	
NSMAX	I	Truncation order	
NMSMAX	I	Truncation order in longitude	
NSMIN	I	Lower truncature for configurations 911 and 912	
NCMAX	I	Truncation order	
NXMAX	I	Truncation order	
NTMAX	I	Truncation order for tendencies (on n, m<= NSMAX)	
NPROMA	I	Working dimension for grid-point computations	
NFPP3M	I	Maximum number of extra 3-D fields in post-processing	
NRLEVX	I	Dimesnion of NVAUTF in YOMGEM	

TABLE 1.23 NAMDYN:DYNAMICS AND DIFFUSION

Name	Type	Description	Default
BETADT	R	Coefficient for semi-implicit scheme for divergence/temperature	
BETAZQ	R	Coefficient for semi-implicit treatment of vorticity and humidity	
FLCCRI	R	Critical value of CFL criterion	
FRANDH	R	Threshold for the wavenumber dependency	
FRANDHC	R	Threshold for the wavenumber dependency	
HDIRDIV	R	Diffusion of divergence	
HDIRQ	R	Diffusion of humidity	
HDIRSP	R	Diffusion of Diffusion of surface pressure	
HDIRSV	R	Diffusion of Diffusion of passive scalars	
HDIRT	R	Diffusion of temperature	
HDIRVOR	R	Diffusion of vorticity	
LFREIN	L	Spectral "enhanced diffusion"	
LVADL	L	Vertical derivatives in non-interpolating SL scheme	
LVADRES	L	Residual vertical advection in non-interpolating SL scheme	
NDLNPR	I	Formulation of delta used in non hydrostatic model	
NEXPDH	I	Exponent for the diffusion wavenumber dependency	
NEXPDHC	I	Exponent for the diffusion wavenumber dependency	
NITMP	I	Number of iterations to compute medium point	
NLOSP	I	Surface pressure representation	
NORMV	I	Scaling of vertical eigenmodes	
NQLAG	I	Formulation or discretisation of moisture equation	
NSREFDH	I	Threshold for the truncation dependency	
NTHDHC	I	Threshold truncature for wavenumber dependency	
NTLAG	I	Formulation or discretisation of temperature equation	
NVINTSL	I	Vertical extrapolations	
NVLAG	I	Formulation or discretisation of continuity equation	
REFGEO	R	Value of reference geopotential	
REPS1	R	Time-filter constant applied at t-1	
REPS2	R	Time-filter constant applied at t+1	
REPSP1	R	Timefiltering constant applied to t-1 for all surface fields	
RFREIN	R	Constant for spectral "enhanced diffusion"	
RKROMA	R	Leap-frog or Matsuno timesheme	
SIPR	R	Reference surface pressure	
SIQR	R	Reference humidity	
SITR	R	Reference temperature	

TABLE 1.23 NAMDYN:DYNAMICS AND DIFFUSION

Name	Type	Description	Default
SLEVDH	R	First threshold for the pressure dependency	
SLEVDH2	R	Second threshold for the pressure dependency	
TSTEP	R	Time step in seconds	
VESL	R	Decentering factor for semi-implicit scheme and averages along SL trajectory	
VMAX1	R	Warning threshold V > VMAX1	
VMAX2	R	Abort threshold V > VMAX2	
VMXPLA	R	Maximum velocity for additional horizontal diffusion	
VNORM	R	Constant for new scaling semi-implicit scheme	

:

TABLE 1.24 NAMGRIB:GRIB CODING DESCRIPTORS

Name	Type	Description	Default
NBITSGG	I	Number of bits for packing grid-point data	
NBITSSH	I	Number of bits for packing spectral coefficients	
NENSFNB	I	Ensemble forecast number	
NLOCGRB	I	ECMWF local usage identifier	
NSMAXNP	I	Sub-truncation for complex packing of spectral coefficients	
NTOTENS	I	Total number of forecasts in ensemble	

TABLE 1.25 NAMIOS: I/O CONTROL

Name	Type	Description	Default
CCPCCGL	C*120	Name of file which holds the preconditioner	
CEVCGL	C*120	Name of file to which eigenvectors are written	
CFRCF	C*120	Pathname for restart control fil	
CIOCFU	C*120		
CIODDHRF	C*120	Pathname for DDH restart file	
CIODTND	C*120	Pathname of work file	
CIOFOU1	C*120	Pathname for first/second scan Fourier data	
CIOFOU15	C*120	Pathname for first/second scan Fourier data	
CIOFURF	C*120	Pathname for accumulated fluxes restart file	
CIOGAUX	C*120		

TABLE 1.25 NAMIOS: I/O CONTROL

Name	Type	Description	Default
CIOGPP	C*120		
CIOGPUA	C*120	Pathname for grid-point work file	
CIOGP5	C*120		
CIOGUARF	C*120	Pathname for T grid-point restart file	
CIOHOU	C*120	Pathname for spectral to Hough space matrix file	
CIOLPOL	C*120	Pathname for Legendre polynomial work file	
CIOPIN	C*120	Pathname for file containing partially imlicit help arrays	
CIOSCF	C*120	Pathname for cumulated fluxes file	
CIOSPBUF	C*120		
CIOSPEC	C*120		
CIOSPH	C*120	Pathname for spectral to Hough space matrix file	
CIOSPRF	C*120	Pathname for spectral restart file	
CIOSSU	C*120	Pathname for surface file	
CIOSUA	C*120		
CIOSURF	C*120	Pathname for surface fields restart file	
CIOSXF	C*120		
CIOTMDT	C*120	Pathname for T-dT work file	
CIOTMRF	C*120	Pathname for T-dT restart file	
CIOXFU	C*120		
CIOXURF	C*120	Pathname for accumulated fluxes restart file	
NEXPBCF	I	Number of bits for exponent when packing	
NEXPBGP	I	Number of bits for exponent when packing	
NEXPBGP5	I	Number of bits for exponent when packing	
NEXPBGU	I	Number of bits for exponent when packing	
NEXPBGX	I	Number of bits for exponent when packing	
NEXPBXF	I	Number of bits for exponent when packing	
NIOBFDT	I	Number of buffers for I/O	
NIOBFIT	I	Number of buffers for I/O	
NIOBFIT5	I	Number of buffers for I/O	
NPCKFCF	I	Packing factor	
NPCKFGP	I	Packing factor	
NPCKFGP5	I	Packing factor	
NPCKFGU	I	Packing factor	
NPCKFGX	I	Packing factor	
NPCKFT9	I	Packing factor	
NPCKFXF	I	Packing factor	

TABLE 1.26 NAMMCC: CLIMATE VERSION

Name	Type	Description	Default
LMCCEC	L	The lower boundary are updated from "climate files"	FALSE
LMCCIEC	L	The lower boundary conditions are interpolated in time	TRUE
LMCC04	L	SST coupling	FALSE

TABLE 1.27 NAMNMI: NORMAL MODE INITIALISATION

Name	Type	Description	Default
FILTPRD	R	Period below which diabatic tendencies will not be considered	
FILTR	R	Cut-off array of frequencies defining Rossby modes	
LCOIMP	L	Compute the IMPLICIT NMI help arrays	
LCONMO	L	Compute normal modes	
LCOTYDB	L	Subset of diabatic components index must be computed	
LCOTYGR	L	Subset of Gravity/Rossby index must be computed	
LCOTYTD	L	Subset of tidal components index must be computed	
LEULNMI	L	Eulerian tendencies are used in "Machenhauer" steps	
LEXTIDE	L	Exclude atmospheric tide from initialisation	
LGPROJ	L	Projection of fields onto inertia-gravity modes	
LICONV	L	Perform a (dummy) iteration to test convergence	
LNLNMI	L	Non-linear normal mode initialisation required	
LNMI	L	Linear normal mode initialisation required	
LNMIDB	L	Diabatic initialization	
LNMIFF	L	frequency filtering is used in initialization	
LNMIPR	L	Print frequencies of explicit mode	
LNMIRQ	L	Normal mode initialisation required	
LNPROJ	L	Projection of fields onto normal modes	
LRGENE	L	Normal mode initialisation diagnostics	
LRPIMP	L	Partially implicit initialisation	
LRPROJ	L	Projection of fields onto Rossby modes	
LRSPSL	L	Restore surface pressure after all but last iteration	
LSPNDG	L	NMI spectral diagnostics	
LSTDNMI	L	Standard NMI	
LTIDCEP	L	The CEPMMT scheme is required	
LTIDEME	L	The Arpege EMERAUDE (Geleyn et al.) scheme is required	
LTRANS	L	NMI required on TRANSFORMED sphere	
NACUMTF	I	Time step at which to finish accumulating diabatic tendencies	
NACUMTS	I	Time step from which to start accumulating diabatic tendencies	
NFILTM	I	Maximum zonal wavenumber for inclusion of diabatic tendencies	
NFILTN	I	Maximum total wavenumber for inclusion of diabatic tendencies	
NFILTV	I	Subset of vertical modes for projections of diabatic tendencies	
NITNMI	I	Number of iterations of non-linear NMI	
NLEX	I	Total wave number beyond which NMI is done	

TABLE 1.27 NAMNMI: NORMAL MODE INITIALISATION

Name	Type	Description	Default
NOPTDIA	I	Options for diabatic scheme	
NOPTEME	I	Filtering if the diabatic scheme of EMERAUDE is used	
NOPTMAR	II	Definition of tidal waves	
NTIDL	I	Number of meridional wavenumbers used in tidal file	
NTIDMAX	I	Maximum length of tidal file	
NTIDMIN	I	Minimum length of tidal file for which tidal are really excluded	
NVMOD	I	Number of vertical modes	
NVMODF	I	Hough modes association usage	
NVMODF1	I	Projection amplitudes for vertical modes	
NVMODF2	I	Projection amplitudes at NVMODF2 and beyond are set to zero	
NVMODT	I	Number of vertical modes in tidal file	
RTACUMT	I	Time step during which diabatic tendenciesare accumulated	

TABLE 1.28 NAMPAR0:DISTRIBUTED MEMORY CONFIGURATION

Name	Type	Description	Default
LMESSP	L	Distributed memory message passing version	
LMPDIAG	L	Extensive message passing diagnostic output	
NOUTPUT	I	Controls diagnostic output	
NPRGPEW	I	Number of processors used during grid-point phase in East-West	
NPRGPNS	I	Number of processors used during grid-point phase in North-South	
NPROC	I	Total number of processors requested for this run	
NPROCK	I	Number of processors to be used simultaneously for Kalman filter forecasts	
NPRTRV	I	Number of processors used during transform phase in vertical	
NPRTRW	I	Number of processors used during transform phase in wave space	

TABLE 1.29 NAMPAR1:LAYOUT OF DISTRIBUTION

Name	Type	Description	Default
LAPPLE	L	Grid point decomposition style	
LBIDIR	L	Bi-directional transpositions	
LOCKIO	L	I/O to be done in locked regions	
LSLSYNC	L	SL communication reads/writes synchronised	
LSPLIT	L	Latitude sharing between A-sets	

TABLE 1.29 NAMPAR1:LAYOUT OF DISTRIBUTION

Name	Type	Description	Default
LPPTSF	L	Control of post processing file contents in terms of timesteps	
NAPLAT	I	Number of apple latitudes at the poles	
NCOMBFLEN	I	Size of communication buffer	
NCOSTLAG	I	Lagging factor for cost function gathering	
NFLDIN	I	Number of input fields to be buffered during distribution	
NFLDOUT	I	Number of output fields to be buffered during gathering	
NINTYPE	I	Type of input I/O processing	
NLAGA	I	Lagging factor for transpositions in a-direction	
NLAGB	I	Lagging factor for transpositions in b-direction	
NLAGBDY	I	Lagging factor for forecast error gatherin	
NOUTTYPE	I	Type of output (post) processing	
NPPBUFLEN	I	Size in integer words of a single pp file buffer	
NPPFILES	I	Maximum number of pp file buffers	
NSLPAD	I	Number of pad words at either side of the sl halo	
NSTRIN	I	Number of processors required to perform input processing	
NSTROUT	I	Number of processors required to perform output processing	
NVALAG	I	Lagging factor for spectral array gathering	

TABLE 1.30 NAMRES:RESTART TIME

Name	Type	Description	Default
LSDHM	L	Time stamp control	
NFRRES	I	Restart file interval	
NUMRFS	I	Number of concurrent restart files before deletion	
NRESTS	I	List of restart times	

TABLE 1.31 NAMTRAJP: ECMWF LINEAR PHYSICS

Name	Type	Description	Default
LECOND2	L	Linear large scale condensation scheme	FALSE
LECUBM2	L	Linear adjustment convection scheme	FALSE
LECUMF2	L	Linear mass-flux convection scheme	FALSE
LEDCLD2	L	Linear diagnostic cloud scheme	FALSE
LEGWDG2	L	Linear subgrid-scale orography scheme	FALSE

TABLE 1.31 NAMTRAJP: ECMWF LINEAR PHYSICS

Name	Type	Description	Default
LEKPERT	L	Perturbation of exchange coefficients	FALSE
LEQNGT2	L	Linear negative humidity fixer	FALSE
LERADI2	L	Linear radiation scheme	FALSE
LERADS2	L	Interactive surface radiative properties	FALSE
LESURF2	L	Linear land surface scheme	FALSE
LETRAJP	L	Storage of the trajectory t-dt	FALSE
LEVDIF2	L	Linear vertical diffusion scheme	FALSE
LIOTRPH	L	Trajectory stored on file	FALSE
NEXPBT95	I	Packing exponent for trajectory t-dt	6
NG2D95	I	Number of 2D fields to be stored	15
NG3D95	I	Number of 3D fields (full levels) to be stored	9
NG3P95	I	Number of 3D fields (half levels) to be stored	2
NG3S95	I	Number of 3D fields (soil) to be stored	2
NPCKFT95	I	Packing parameter for trajectory t-dt	1

1.7 POST PROCESSING

This is described in Chapter 2 'FULL-POS post-processing and interpolation' .

1.8 RESTART CAPABILITY

The restart option is designed so that long-running IFS executions (typically configuration 1 with long integrations) can be made to write the model state to files. This data is such that it is possible subsequently to restart from these files and exactly reproduce the same model computations. An additional flexibility is retained by allowing the re-started run to use a different number of processors to the original run. To achieve this, all data is communicated to the master processor who is then responsible for writing it to files. The existence of these restart files is recorded by writing a 'restart control file' which contains additional control information to enable a clean restart to be made. Control over when restart files are written is achieved in two ways:

1) by predefining the timesteps at which the files should be written. This is achieved using the NAMRES namelist array NRESTS.

2) by sending a signal to the running job.

Signal 1 means "write restart files and continue".

Signal 2 means "write restart files and stop"

The files written and their contents are described in the following tables. All file names may be modifed using NAMIOS namelist variables. The file names are all appended with a timestamp in the form of 'ddddhhmm' to reflect the model timestep.

TABLE 1.32 RESTART FILES

IFS Fortran name	Default name	Contents	Condition
CIOSPRF	srf	Upper air 3D and surface 2D spectral fields	
CIOTMRF	tmrf	T-dt 3D grid-point data	
CIOSURF	surf	Surface grid-point data	
CIOGUARF	guarf	Upper air grid-point data at time T	
CIOFURF	furf	Cumulated and instantaneous fluxes grid-point data	LCUMFU = TRUE
CIOXURF	xurf		LXFU = TRUE
CIODDHRF	ddhrf	DDH data	LSDDH = TRUE
CFRCF	rcf	Namelist NAMRCF: CSTEP-character*8 : time step number CTIME-character*8: forecast time as 'DDDDHHMM' NSTEPLPP-integer array: for time range fields, last step number when each GRIB code was post-processed. GMASSI-real: mass at start of integration GMASS0-real: mass at current timestep GMASS9-real: mass at previous timestep	

1.9 MASS CONSERVATION

A global mass-conservation scheme can be invoked at any desired time steps using namelist NAMCT0 parameters. the NMASSCONS array defines steps at which mass conservation is to be applied. NFRMASSCON can be used to set the frequency of this control. LMASPRT (in namelist NAMDYN) provides additional print information on this process. The mass conservation control is preserved across a model restart.

Part VI: TECHNICAL AND COMPUTATIONAL PROCEDURES

CHAPTER 2 FULL-POS post-processing and interpolation

This documentation describes the FULL-POS functionality of ARPEGE/IFS .

Table of contents

2.1 POST-PROCESSABLE FIELDS.

Post-processing can be made on pressure levels, height levels, potential vorticity levels, potential temperature levels or η -levels.

2.1.1 3D dynamical fields (DYN3D).

- 01) Geopotential Φ.
- 02) Temperature T.
- 03 and 04) Horizontal wind components U and V.
- 05) Specific humidity (moisture) q.
- 06) Relative humidity HU.
- 07) Ice content q_{i}.
- 08) Pressure coordinate vertical velocity ω.
- 09) Relative vorticity ζ.
- 10) Divergence D.
- 11) Potential temperature Θ.
- 12) Velocity potential ψ (has to be fitted).
- 13) Stream function χ (has to be fitted).
- 14) Liquid water content q_{l} .
- 15) Moist (irreversible) pseudo-adiabatic potential temperature Θ'_{w}.
- 16) Cloud fraction q_a .
- 17) Wind velocity.
- 18) Equivalent potential temperature Θ_{e}.
- 19) Absolute vorticity $\zeta + f$.
- 20) Stretching deformation STD.
- 21) Shearing deformation SHD.
- 22) Potential vorticity PV.
- 23) Wet potential vorticity PV_{w}.
- 24 to 26) Passive scalars q_{SV} (maximum of three).
- 27) Pressure Π .
- 28 to 30) Possibility to post-process three additional free upper air fields nr 1 to 3 (METEO-FRANCE and ECMWF).
- 31 to 43) Possibility to post-process 13 additional free upper air fields nr 4 to 16 (ECMWF only).

2.1.2 2D dynamical fields (DYN2D).

- 01) Surface pressure Π_{s} .
- 02) Mean sea level pressure Π_{MSL}.
- 03) Interpolated (spectral) model orography.
- 04) Mapping factor M.
- 05) Tropopause folding indicator of the iso-2 PVU surface.
- 06) ICAO jet zonal component of wind.
- 07) ICAO jet meridian component of wind.
- 08) Position of ICAO jet in pressure vertical coordinate.
- 09) Position of ICAO tropopause in pressure vertical coordinate.
- 10) ICAO tropopause temperature.
- 11) Logarithm of surface pressure (at ECMWF only).

Remark about fields 06) to 10): at ECMWF they are replaced by some optional surface fields.

2.1.3 Surface physical fields (PHYSOL) used both at METEO-FRANCE and ECMWF.

- 01) Land/sea mask.
- 02) (Output grid-point orography) $\times g$.
- 03) Surface temperature.
- 04) Deep soil temperature.
- 05) Interpolated surface temperature.
- 06) Surface soil wetness.
- 07) Deep soil wetness.
- 08) Relaxation deep soil wetness.
- 09) Climatological relative surface soil wetness.
- 10) Climatological relative deep soil wetness.
- 11) Snow depth.
- 12) (Surface roughness) $\times g$.
- 13) (Roughness length of bare surface) $\times g$.
- 14) Albedo.
- 15) Emissivity.
- 16) (Standard deviation of orography) $\times g$.
- 17) Percentage of vegetation.
- 18) Percentage of land.
- 19) Anisotropy coefficient of topography.
- 20) Direction of main axis of topography.

2.1.4 Additional surface physical fields (PHYSOL) used only at ECMWF.

- 21) Soil first level temperature.
- 22) Soil first level wetness.
- 23) Soil second level temperature.
- 24) Soil second level wetness.
- 25) Soil third level temperature.
- 26) Soil third level wetness.
- 27) Soil fourth level temperature.
- 28) Soil fourth level wetness.
- 29) Temperature of snow layer.
- 30) Anisotropy of surface orography.
- 31) Angle of surface orography.
- 32) Slope of surface orography.
- 33) Logarithm of surface roughness.
- 34) Skin temperature.
- 35) Apparent surface humidity.
- 36) Skin wetness.
- 37 to 79) Diagnostic fields (not detailed, some of them are present in the CFU and XFU list).
- 80 to 109) Additional optional fields.

2.1.5 Surface cumulated fluxes (CFU).

- 01) Large scale precipitation.
- 02) Convective precipitation.
- 03) Large scale snow fall.
- 04) Convective snow fall.

- 05) U -stress.
- 06) V -stress.
- 07) Surface sensible heat flux.
- 08) Surface latent heat flux.
- 09) Tendency of surface pressure.
- 10) Total cloud cover.
- 11) Boundary layer dissipation.
- 12) Surface solar radiation.
- 13) Surface thermal radiation.
- 14) Top solar radiation.
- 15) Top thermal radiation.
- 16) Convective cloud cover.
- 17) High cloud cover.
- 18) Medium cloud cover.
- 19) Low cloud cover.
- 20) U -gravity-wave stress.
- 21) V -gravity-wave stress.
- 22) Water evaporation.
- 23) Snow sublimation.
- 24) Latent heat evaporation.
- 25) Latent heat sublimation.
- 26) Cloudiness.
- 27) Soil moisture.
- 28) Snow mass.
- 29) Total precipitable water.
- 30) Total ozone.
- 31) Top mesospheric enthalpy.
- 32) Solid specific moisture.
- 33) Liquid specific moisture.
- 34) Contribution of convection to U .
- 35) Contribution of convection to V .
- 36) Contribution of convection to q .
- 37) Contribution of convection to $c_p T$.
- 38) Contribution of turbulence to q .
- 39) Contribution of turbulence to $c_p T$.
- 40) Clear sky shortwave radiative flux.
- 41) Clear sky longwave radiative flux.
- 42) Surface parallel solar flux.
- 43) Top parallel solar flux.
- 44) Surface down solar flux.
- 45) Surface down thermic flux.
- 46) Melt snow.
- 47) Heat flux in soil.
- 48) Water flux in soil.
- 49) Surface soil runoff.
- 50) Deep soil runoff.
- 51) Interception soil layer runoff.

- 52) Evapotranspiration flux.
- 53) Transpiration flux.

2.1.6 Surface instantaneous fluxes (XFU).

- 01) Total cloud cover.
- 02) U-component of wind at 10 meters (pbl).
- 03) V-component of wind at 10 meters (pbl).
- 04) Temperature at 2 meters (pbl).
- 05) Specific humidity at 2 meters (pbl).
- 06) Relative humidity at 2 meters (pbl).
- 07) Convective cloud cover.
- 08) High cloud cover.
- 09) Medium cloud cover.
- 10) Low cloud cover.
- 11) Maximum temperature at 2 meters.
- 12) Minimum temperature at 2 meters.
- 13) Cloudiness.
- 14) Contribution of convection to U.
- 15) Contribution of convection to V.
- 16) Contribution of convection to q.
- 17) Contribution of convection to $c_p T$.
- 18) Contribution of turbulence to U.
- 19) Contribution of turbulence to V.
- 20) Contribution of turbulence to q.
- 21) Contribution of turbulence to $c_p T$.
- 22) Contribution of gravity wave drag to U.
- 23) Contribution of gravity wave drag to V.
- 24) Large scale precipitation.
- 25) Convective precipitation.
- 26) Large scale snow fall.
- 27) Convective snow fall.
- 28) Surface solar radiation.
- 29) Surface thermal radiation.
- 30) Top solar radiation.
- 31) Top thermal radiation.
- 32) U at bottom level.
- 33) V at bottom level.
- 34) Temperature at bottom level.
- 35) Specific humidity at bottom level.
- 36) Geopotential at bottom level.
- 37) Surface temperature.
- 38) Deep soil temperature.
- 39) Surface water content.
- 40) Deep soil water content.
- 41) Snow mass.

2.2 OTHER NOTATIONS.

- L : number of layers of the model.
- $(\mathrm{d}T/\mathrm{d}z)^{\mathrm{st}}$: standard atmosphere vertical gradient of the temperature in the troposphere (0.0065 K m^{-1}).
- R_{a} : dry air constant.
- g : gravity acceleration.

2.3 HORIZONTAL INTERPOLATIONS.

Horizontal interpolations can be bilinear interpolations or 12 points cubic interpolations. No namelist variable is available to switch from 12 points to bilinear interpolations. The call to 12 points interpolations is hard coded.

2.3.1 Bilinear horizontal interpolations.

Horizontal interpolation grid and weights for bilinear interpolations.

A 16 points horizontal grid is defined as is shown in Fig. 2.1 . The interpolation point O is between $B1$, $C1$, $B2$ and $C2$. Λ and Θ are the longitudes and latitudes on the computational sphere (departure geometry). The following weights are defined as follows:

- zonal weight number 1:

$$ZDLO1 = \frac{\Lambda_O - \Lambda_{B_1}}{\Lambda_{C_1} - \Lambda_{B_1}}$$

- zonal weight number 2:

$$ZDLO2 = \frac{\Lambda_O - \Lambda_{B_2}}{\Lambda_{C_2} - \Lambda_{B_2}}$$

- meridian weight:

$$ZDLAT = \frac{\Theta_O - \Theta_{B_1}}{\Theta_{B_2} - \Theta_{B_1}}$$

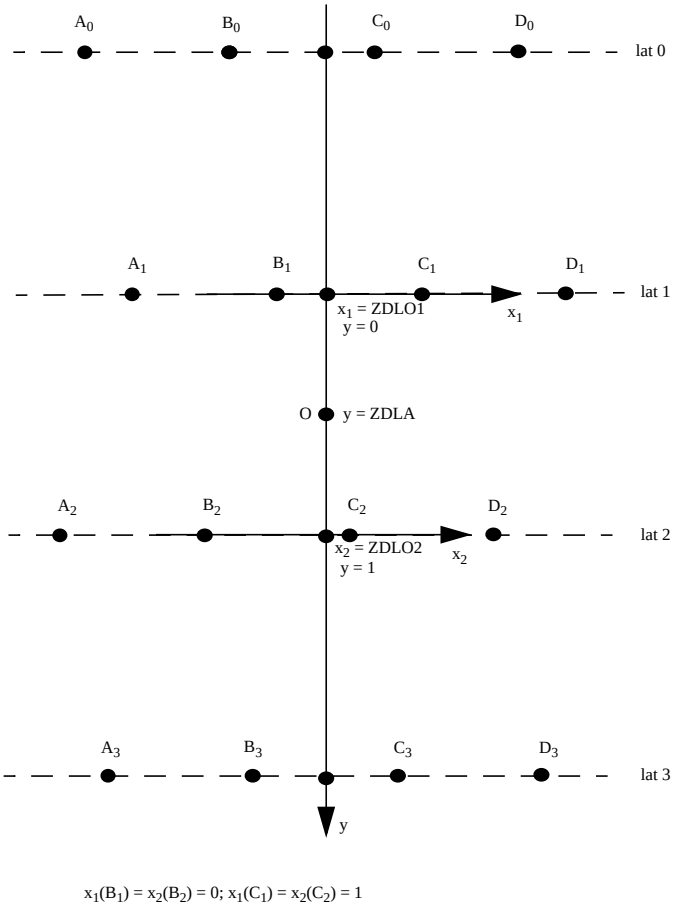


Figure 2.1 Interpolation horizontal grid for bilinear interpolations.

Bilinear interpolation.

For a quantity X, are computed successively:

- a linear interpolation on the longitude number 1:

$$X_1 = X_{B_1} + ZDLO1(X_C - X_{B_1}).$$

- a linear interpolation on the longitude number 2:

$$X_2 = X_{B_2} + ZDLO2(X_{C_2} - X_{B_2}).$$

- a meridian linear interpolation:

$$X_{\text{interpolated}} = X_1 + ZDLAT(X_2 - X_1).$$

In the FULL-POS code the weights are pre-computed in routines **SUHOW2** and **SUHOWLSM**, so the separation of zonal and meridian interpolations is not visible in the interpolation routines. Currently there is no routine coded to do FULL-POS bilinear interpolations.

2.3.2 12 points horizontal interpolations.

Horizontal interpolation grid and weights for 12 points cubic interpolations.

A 16 points horizontal grid is defined as shown in Fig. 2.2 . The interpolation point O is between $B1$, $C1$, $B2$ and $C2$. The following weights are defined as follows:

- zonal weight number 0:

$$ZDLO0 = \frac{\Lambda_O - \Lambda_{B_0}}{\Lambda_{C_0} - \Lambda_{B_0}}$$

- zonal weight number 1:

$$ZDLO1 = \frac{\Lambda_O - \Lambda_{B_1}}{\Lambda_{C_1} - \Lambda_{B_1}}$$

- zonal weight number 2:

$$ZDLO2 = \frac{\Lambda_O - \Lambda_{B_2}}{\Lambda_{C_2} - \Lambda_{B_2}}$$

- zonal weight number 3:

$$ZDLO3 = \frac{\Lambda_O - \Lambda_{B_3}}{\Lambda_{C_3} - \Lambda_{B_3}}$$

- meridian weights:

$$ZCLA2 = \frac{(\Theta_O - \Theta_{B_0})(\Theta_O - \Theta_{B_2})(\Theta_O - \Theta_{B_3})}{(\Theta_{B_1} - \Theta_{B_0})(\Theta_{B_1} - \Theta_{B_2})(\Theta_{B_1} - \Theta_{B_3})}$$

$$ZCLA3 = \frac{(\Theta_O - \Theta_{B_0})(\Theta_O - \Theta_{B_1})(\Theta_O - \Theta_{B_3})}{(\Theta_{B_2} - \Theta_{B_0})(\Theta_{B_2} - \Theta_{B_1})(\Theta_{B_2} - \Theta_{B_3})}$$

$$ZCLA4 = \frac{(\Theta_O - \Theta_{B_0})(\Theta_O - \Theta_{B_1})(\Theta_O - \Theta_{B_2})}{(\Theta_{B_3} - \Theta_{B_0})(\Theta_{B_3} - \Theta_{B_1})(\Theta_{B_3} - \Theta_{B_2})}$$

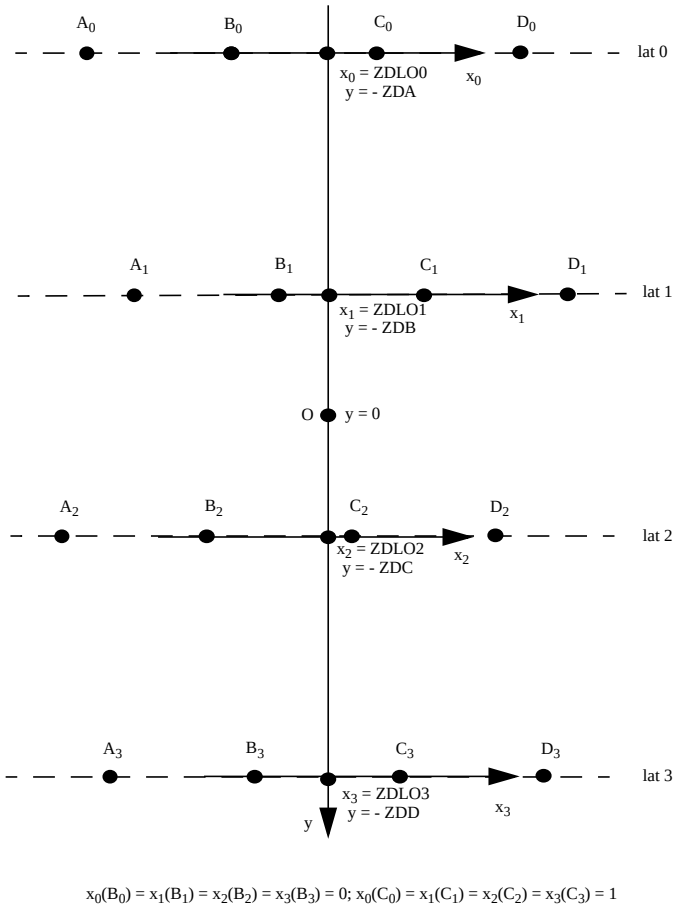


Figure 2.2 Interpolation horizontal grid for 12 point interpolation.

Horizontal 12 points interpolation.

Let us define:

- $f_2(\alpha) = (\alpha+1)(\alpha-2)(\alpha-1)/2$
- $f_3(\alpha) = -(\alpha+1)(\alpha-2)\alpha/2$
- $f_4(\alpha) = \alpha(\alpha-1)(\alpha+1)/6$

For a quantity *X*, are computed successively:

- a linear interpolation on the longitude number 0:

$$X_0 = X_{B_0} + ZDLO0(X_{C_0} - X_{B_0}).$$

- a cubic 4 points interpolation on the longitude number 1:

$$X_1 = X_{A_1} + f_2(ZDLO1)(X_{B_1} - X_{A_1}) + f_3(ZDLO1)(X_{C_1} - X_{A_1}) + f_4(ZDLO1)(X_{D_1} - X_{A_1}).$$

- a cubic 4 points interpolation on the longitude number 2:

$$X_2 = X_{A_2} + f_2(ZDLO2)(X_{B_2} - X_{A_2}) + f_3(ZDLO2)(X_{C_2} - X_{A_2}) + f_4(ZDLO2)(X_{D_2} - X_{A_2}).$$

- a linear interpolation on the longitude number 3:

$$X_3 = X_{B_3} + ZDLO3(X_{C_3} - X_{B_3}).$$

- a meridian cubic 4 points interpolation:

$$X_{\text{interpolated}} = X_0 + ZCLA2(X_1 - X_0) + ZCLA3(X_2 - X_0) + ZCLA4(X_3 - X_0).$$

In the FULL-POS code the weights are pre-computed in routines **SUHOW2** and **SUHOWLSM**, so the separation of zonal and meridian interpolations is not visible in the interpolation routines.

2.3.3 Location of computations.

Once the coordinates of the interpolation points known:

- The "north-western" model grid point coordinates are computed in the routine **SUHOW1**. This is the model (departure geometry) grid point which is immediately at the north-west of the interpolation point.
- The weights not modified by the land-sea mask are computed in routine **SUHOW1**.
- The weights modified by the land-sea mask are computed in routine **SUHOWLSM**. This is equivalent to use weights not modified by the land-sea mask and to multiply the field to be interpolated by the land-sea mask (0 if sea, 1 if land).
- The horizontal 12 points interpolations are done by routines **FPAERO** (variables linked to aeronautic jet), **FPHOR12** (3D derivatives) and **FPINT12** (other variables). Routine **FPMIMAX** is used to give to the interpolated quantity the value of the nearest model grid point (used for maximum and minimum temperature at 2 meters). They need intermediate quantities computed in routine **FPSCAW**.
-

Additional modifications can be done in the interpolation routines after interpolations (principally in routine **FPINT12**), for example add a monotonicity condition. More details about these additional modifications and the post-processed fields concerned by these modifications are described in the part describing each routine.

2.3.4 Plane geometry (ALADIN).

All previous formulae for weight computation can be used for an irregular latitude spacing and a different number of points on each longitude. The ALADIN grid has a horizontal regular spacing, so the previous formulae can be simplified. **SUEHOW1**, **SUEHOW2** and **SUEHOWLSM** are called instead of **SUHOW1**, **SUHOW2** and **SUHOWLSM**. **SUEHOW1**, **SUEHOW2** and **SUEHOWLSM** are currently not yet coded and have to be coded in the future cycle AL09 based on the future cycle 18T1 of ARPEGE/IFS.

2.4 VERTICAL INTERPOLATIONS AND EXTRAPOLATIONS.

2.4.1 General considerations.

For 3D variables to be vertically interpolated, vertical interpolations are generally linear interpolations between the layers where are defined model variables. The treatment of the extrapolations above the upper layer, the extrapola-

tions below the lower layer or the surface depend on the variable considered. In particular cases some variables can be diagnosed using the vertically interpolated value of some other variables.

The '**OLD**' versions of routines when available are used at ECMWF (switch **LOLDPP**=.T.). METEO-FRANCE uses the option **LOLDPP**=.F. .

2.4.2 More details for 3D dynamical variables.

Wind components, wind velocity.

Way of interpolating if **LOLDPP**=.T. (routine **PPUV_OLD**):

- Linear interpolation between the layer 2 and the lower layer.
- The coordinate used for linear interpolation is the logarithm of the pressure.
- Quadratic interpolation between the layer 1 and the layer 2 using the values of the layers 1, 2 and 3.
- Quadratic interpolation between the top and the layer 1 using the values of the top, layers 1 and 2; the value of the top is obtained by a linear extrapolation from the values of the layers 1 and 2.
- The coordinate used for quadratic interpolation is the logarithm of the pressure.
- Extrapolation below the middle of the lower layer and below the surface assumes that the quantity is constant.

Way of interpolating if **LOLDPP**=.F. (routine **PPUV**):

- The same as for **LOLDPP**=.T. but quadratic interpolations are replaced by linear interpolations.

Temperature.

Applies to temperature if the vertical coordinate of post-processing is not the potential vorticity, otherwise see routine **PP2DINT**.

Way of interpolating if **LOLDPP**=.F. (routine **PPT**):

- Quadratic interpolation between the middles of the upper and lower layers.
- Quadratic interpolation between the top and the middle of the upper layer: the top value of the temperature is assumed to be equal to the value of the middle of the upper layer; due to the fact that the interpolation is a quadratic one, that does not mean that the temperature is constant in this atmosphere depth.
- The coordinate used for quadratic interpolation is the logarithm of pressure. For more details about the quadratic interpolation used, which is a quadratic analytic expression of the logarithm of pressure, and the reason of using a quadratic interpolation, see (Undén, 1995).
- A surface temperature \TSURF is computed as follows:

$$T_{\text{surf}} = T_L + \left(\frac{\mathrm{d}T}{\mathrm{d}z}\right)^{\text{st}} \frac{R_a}{g}\left(\frac{\Pi_s}{\Pi_L} - 1\right) T_L \tag{2.1}$$

- Extrapolation below the middle of the lower layer and the surface is a linear interpolation between T_L and T_{surf} .
- Extrapolation under the surface is made according a more complicated algorithm:

$$T_{\text{extrapolated}} = T_{\text{surf}}\left(1 + y + \frac{y^2}{2} + \frac{y^3}{6}\right) \tag{2.2}$$

where:

$$y = \Gamma \frac{R_a}{g} \log\left(\frac{\Pi_s}{\Pi_L}\right) \tag{2.3}$$

If $\Phi_s / g < 2000$ m, $\Gamma = (dT/dz)^{st}$; if $\Phi_s / g \geq 2000$ m, the expression for Γ is more complicated:

$$\Gamma = \frac{g}{\Phi_s} \max(T_0' - T_{surf}, 0) \tag{2.4}$$

if $\Phi_s / g > 2500$ m, :

$$T_0' = \min\left(T_{surf} + \left(\frac{dT}{dz}\right)^{st} \frac{\Phi_s}{g}, 298 \text{ K}\right) \tag{2.5}$$

- $\Phi_s / g \leq 2500$ m and $\Phi_s / g \geq 2000$ m, : T_0' is computed by a linear interpolation (coordinate of interpolation is Φ_s) between the two values $\min(T_{surf} + (dT/dz)^{st}(\Phi_s / g), 298 \text{ K})$ and $T_{surf} + (dT/dz)^{st}(\Phi_s / g)$

Way of interpolating if **LOLDPP**=.T. (routine **PPT_OLD**):

- Linear interpolation (between the upper and the lower layer).
- The coordinate used for linear interpolation is the pressure.
- Extrapolation above the middle of the upper layer assumes that the quantity is constant.
- Extrapolation below the middle of the lower layer and the surface is a linear interpolation between *TL* and \TSURF like in **PPT**.
- Extrapolation under the surface is made according the same algorithm as in **PPT** (code of part 1.4 is different in **PPT** and in **PPT_OLD** but actually does the same calculations).

Geopotential.

Applies to geopotential if the vertical coordinate of post-processing is not the potential vorticity, otherwise see routine **PP2DINT**.

Way of interpolating if **LOLDPP**=.T. (routine **PPGEOP_OLD**):

- The variable interpolated is a geopotential departure from a reference defined by a standard atmosphere without any orography. After the interpolation an increment is added, sum of the surface orography and the 'standard' geopotential depth between the pressure level of interpolation and the actual surface. This method avoids to introduce interpolations for the standard component of the geopotential which can be computed analytically (in routine **PPSTA**).
- Linear interpolation between the layer 2 and the surface.
- The coordinate used for linear interpolation is the logarithm of the pressure.
- Quadratic interpolation between the layer 1 and the layer 2 using the values of the layers 1, 2 and 3.
- Quadratic interpolation between the top and the layer 1 using the values of the top, layers 1 and 2.
- The coordinate used for quadratic interpolation is the logarithm of the pressure.
- Extrapolation below surface uses the surface temperatur T_{surf} of Eq. (2.1).

$$\Phi_{extrapolated} = \Phi_s - R_a T_{surf} \log\left(\frac{\Pi_s}{\Pi_L}\right)\left(1 + \frac{y}{2} + \frac{y^2}{6}\right) \tag{2.6}$$

- where y is defined by formula (2.3) with $\Gamma = ((dT)/(dz))^{st}$ in all cases.

- For more details about this algorithm, see (Andersson and Courtier, 1992) which is still valid for the cycles 17 and 18.

Way of interpolating if **LOLDPP**=.F. (routine **PPGEOP**):

- The variable interpolated is a geopotential departure from a reference defined by a standard atmosphere without any orography. After the interpolation an increment is added, sum of the surface orography and the "standard" geopotential depth between the pressure level of interpolation and the actual surface. This method avoids to introduce interpolations for the standard component of the geopotential which can be computed analytically (in routine **PPSTA**).
- Quadratic interpolation between the middles of the upper and lower layers.
- Quadratic interpolation between the top and the middle of the upper layer.
- The coordinate used for quadratic interpolation is the logarithm of pressure. The quadratic interpolation is not exactly the same as for **LOLDPP**=.T., it is a quadratic analytic expression of the logarithm of pressure of the same type as the one used to post-process the temperature for **LOLDPP**=.F. . For more details about the quadratic interpolation used, and the reason of using a quadratic interpolation, see (Undén, 1995).
- Linear interpolation between the lower layer and the surface, as for **LOLDPP**=.T. .
- Extrapolation below surface uses the same algorithm as for **LOLDPP**=.T. .

2.4.2 (a) Variables interpolated using routine PP2DINT.

List of variables:

- Geopotential Φ if vertical coordinate is potential vorticity.
- Temperature T if vertical coordinate is potential vorticity.
- Relative vorticity ζ.
- Divergence D.
- Potential temperature Θ if vertical coordinate is not potential temperature.
- Velocity potential ψ.
- Stream function χ.
- Equivalent potential temperature Θ_e.
- Absolute vorticity $\zeta + f$.
- Stretching deformation STD.
- Shearing deformation SHD.
- Potential vorticity PV.
- Wet potential vorticity PV_w (not yet coded).

Way of interpolating:

- Linear interpolation (between the upper and the lower layer for quantities defined on the middle of layers, between the layer 1 and the surface for quantities defined on interlayers).
- The coordinate used for linear interpolation is the pressure.
- Extrapolation above the middle of the upper layer assumes that the quantity is constant.
- Extrapolation below the middle of the lower layer and below the surface assumes that the quantity is constant.

Moisture, liquid water, solid water, cloud fraction, passive scalars: variables using routine PPQ.

Way of interpolating (routine **PPQ**):

- Linear interpolation (between the upper and the lower layer).
- The coordinate used for linear interpolation is the pressure.
- Extrapolation above the middle of the upper layer assumes that the quantity is constant.

- Extrapolation below the middle of the lower layer and below the surface assumes that the quantity is constant.

Relative humidity (routine PPRH).

Way of interpolating (routine **PPRH**):

- Linear interpolation (between the upper and the lower layer).
- The coordinate used for linear interpolation is the pressure.
- Extrapolation above the middle of the upper layer assumes that the quantity is constant.
- Extrapolation below the middle of the lower layer and below the surface assumes that the quantity is constant.

Pressure coordinate vertical velocity ω (routine PPVVEL).

Way of interpolating (routine **PPVVEL**):

- Linear interpolation (between the upper and the lower layer).
- The coordinate used for linear interpolation is the pressure.
- Extrapolation above the middle of the upper layer is a linear interpolation between a zero value at the top and the value of the upper layer.
- Extrapolation between the middle of the lower layer and the surface assumes that the quantity is constant.
- Extrapolation below the surface assumes that the quantity is zero.

Moist (irreversible) pseudo-adiabatic potential temperature Θ'_w (routine PPTHPW.

Routine **PPTHPW** is a diagnostic one. It takes as input the vertically post-processed pressure, temperature, moisture, liquid water and ice and computes Θ'_w at the post-processing levels using a diagnostic (and rather complicated) algorithm.

2.4.3 2D dynamical variables which need extrapolations.

Mean sea level pressure Π_{MSL} (routine PPPMER).

If $|\Phi_s|$ is lower than 0.001 J kg^{-1} the mean sea level pressure is set to the surface pressure. In the other cases one uses the following algorithm:

- One computes the surface temperature T_{surf} of Eq. (2.1) and the 'mean sea level' temperature

$$T_0 = T_{surf} + \left(\frac{dT}{dz}\right)^{st} \frac{\Phi_s}{g} .$$

- To avoid extrapolation of too low pressures over high and warm surfaces the following modifications are done:
  - if $T_0 > 290.5$ K and $T_{surf} \leq 290.5$ K, Γ is defined by:

$$\Gamma = (290.5 - T_{surf})\frac{g}{\Phi_s} \qquad (2.7)$$

  - if $T_0 > 290.5$ K and $T_{surf} > 290.5$ K, Γ is set to 0, T_{surf} is modified and set to $0.5 \times (255\ \text{K} + \text{old value of } T_{surf})$.

- To avoid extrapolation of too high pressures over cold surfaces the following modifications are done when $T_{\text{surf}} < 255$ K: Γ is set to $(\mathrm{d}T/\mathrm{d}z)^{\text{st}}$ and T_{surf} is modified and set to $0.5 \times (255 \text{ K} + \text{old value of } T_{\text{surf}})$.
- In the other cases Γ is set to $(\mathrm{d}T/\mathrm{d}z)^{\text{st}}$.
- Mean sea level pressure is computed as follows:

$$\Pi_{\text{MSL}} = \Pi_{\text{s}} \exp\left\{ \frac{\Phi_{\text{s}}}{R_{\text{a}} T_{\text{surf}}} \left(1 - \frac{x}{2} + \frac{x^2}{3} \right) \right\} \tag{2.8}$$

where:

$$x = \frac{\Gamma \Phi_{\text{s}}}{g T_{\text{surf}}} \tag{2.9}$$

2.5 FILTERING IN SPECTRAL SPACE.

2.5.1 General considerations.

Three filterings are successively applied:

- 1) A "THX" filter on derivatives in ARPEGE, or a bell-shaped filter on derivatives for ALADIN.
- 2) A bell-shaped filter on all fields (derivatives and not derivatives).
- 3) A "THX" filter on all fields.

Filtering is done in routine **pp_obs/SPOS** (**pp_obs/ESPOS** for ALADIN).

2.5.2 'THX' (ARPEGE) or bell-shaped (ALADIN) filtering on derivatives.

Cases where this filtering is used:

This filtering applies in the spectral space to absolute vorticity, relative vorticity, divergence, vertical velocity, stretching and shearing deformations, and potential vorticity (and extend to all variables if the vertical coordinate of post-processing is the potential vorticity). This filter is active if:

- **CFPFMT** is not 'MODEL' in namelist **NAMFPC**.
- **LFPFIL**=.TRUE. in namelist **NAMFPF** for ARPEGE, or **LFPBED**=.TRUE. in namelist **NAMFPF** for ALADIN.
- Variable **NFMAX** is smaller than the 'equivalent unstretched sphere' truncation N_c (in practical N_c is between $1.1 \times c \times N_{\text{s}}$ and $1.2 \times c \times N_{\text{s}}$, where c is the stretching factor.

If you wish to keep these fields unfiltered, then just set **LFPFIL**=.FALSE. in ARPEGE/IFS, or **LFPBED**=.FALSE. in ALADIN (namelist **NAMFPF**). On the other hand, you can keep the filter active but you can tune the filtering function.

Function of filtering:

In ARPEGE/IFS, this function looks like a smoothed step function; for a given total wavenumber *n* in the unstretched spectral space (i.e. the spectral space of the 'equivalent unstretched sphere' of truncation N_c), the formula is:

$$f_{\mathrm{THX}}(n)) = \frac{1-\tanh\{k(n-n_0)\}}{2} \tag{2.10}$$

(The use of the function hyperbolic tangent is the reason of the nickname 'THX' for this filter). It means that this function equals roughly 1 if n is less than n_0, and 0 if it is bigger than n_0.

Tunable parameters in the previous function:

k and n_0 are tunable parameters:

- n_0 is in the variable **NFMAX**. If **CFPFMT**='GAUSS' or 'MODEL' in namelist **NAMFPC**, the default value is $\mathbf{NFPMAX} \times \mathbf{FPSTRET}$; else, it is the truncation of the Gaussian grid which would have been defined by **NLAT** latitudes and **NLON** longitudes with default so that $3 \times \mathbf{NFMAX} + 1 = \min(\mathbf{NLAT}, \mathbf{NLON})$ (see **YOMFPD** and namelist **NAMFPD**).
- k is defined as follows:

$$k = \frac{\ln(1-\varepsilon)/\varepsilon}{L} \tag{2.11}$$

where ε and L are percentages. This means that: on L times the width of the spectrum, the function will decrease quickly, and that at the boundaries of this window, the values of the function will be respectively $(1-\varepsilon)$ and ε. Quantities $L \times n_s$ and ε are respectively in **NFPWID** and **RFPEPS**.

Operations done in SPOS for this filtering:

One assumes that CFPFMT is not 'MODEL'.

- **LFPFIL**=.T. and $N_s < N_c$: the previous function $f_{\mathrm{THX}}(n)$ is computed in the 'equivalent unstretched sphere' of truncation N_c , so reading dilatation and contraction matrixes respectively denoted by D and C (computed by the configuration 911 of ARPEGE/IFS, see the corresponding documentation) is necessary. The operator applied to spectral fields in the computational sphere is a matrical operator $C \times f_{\mathrm{THX}}(n) \times D$ pre-computed in the routine **pp_obs**/**FPFILTER** (called by **setup**/**SU3FPOS**) and stored in the array **RFPMAT**. In **SPOS** the initially unfiltered fields are in **SPDFP** and the filtered fields are put in **SPBFP**. Filtering is done only if **NFMAX** < \Nc , elsewhere there is a simple copy of **SPDFP** in **SPBFP** without filtering.
- **LFPFIL**=.T. and \Ns = \Nc : identity \Ns = \Nc is satisfied if the model resolution has no stretching. fTHX(n) is stored in the array **RFPFIL**. This function is directly applied in **SPOS** to **SPDFP** and the filtered fields are put in **SPBFP**. Filtering is done only if $\mathbf{NFMAX} < N_s$, elsewhere there is a simple copy of **SPDFP** in **SPBFP** without filtering.
- **LFPFIL**=.F.: no filtering, simple copy of **SPDFP** in **SPBFP**.

ALADIN:

In ALADIN, this filter is a 'bell-shaped' function and is done in the computational geometry. For a given pair of wavenumbers (n, m), the formula is:

$$f_{\mathrm{bsAL}}(n, m) = \exp\left[-\frac{k}{2}\left\{\left(\frac{n}{N}\right)^2 + \left(\frac{m}{M}\right)^2\right\}\right] \tag{2.12}$$

where N is the model truncation (**NSMAX**) and k a tunable variable. k is in variable **RFPBED** of namelist **NAMFPF**.

2.5.3 'Bell-shaped' filtering on non-derivative fields.

Way of filtering:

It is also possible to filter the non-derivative post-processed fields through "bell-shaped" filters. Separate 'bell-shaped' filters are available for geopotential, temperature, mean sea level pressure, relative humidity and all other non-derivatives. By default, these filters are active for geopotential, temperature, mean sea level pressure and relative humidity. In ALADIN, the formula is the same as above (formula (2.12)). In ARPEGE/IFS, for a given wavenumber n in the stretched spectral space, the formula is:

$$f_{\mathrm{bsARP}}(n) = \exp\left\{-\frac{k}{2}\left(\frac{n}{N_{\mathrm{s}}}\right)^2\right\} \tag{2.13}$$

where N_{s} is the model truncation (**NSMAX**) and k a tunable variable. In **SPOS**, the bell-shaped filtering is done by multiplying the array **SPAFP** by $f_{\mathrm{bsARP}}(n)$, the result is still in **SPAFP**.

Variables controlling bell-shaped filtering:

- Switches **LFPBEG** for geopotential, **LFPBET** for temperature, **LFPBEH** for relative humidity, **LFPBEP** for mean sea level pressure, **LFPBEL** for other non-derivatives, **LFPBED** for derivatives (**LFPBED** is used only in ALADIN): .TRUE. if filtering, .FALSE. if no filtering.
- Variables **RFPBEG** for geopotential, **RFPBET** for temperature, **RFPBEH** for relative humidity, **RFPBEP** for mean sea level pressure, **RFPBEL** for other non-derivatives, **RFPBED** for derivatives (**RFPBED** is used only in ALADIN): to control the intensity of the filtering (variable k of formula (2.13)).

2.5.4 Additional 'THX' filtering on all fields.

Way of filtering:

Filtering is still made according the formula (2.10): the threshold n_0 is now the variable **NPMAX** and filtering is always made in the computational space. There is filtering only for n between **NPMAX** and N_{s}. If $\mathbf{NPMAX} \geq N_{\mathrm{s}}$ there is no filtering.

2.6 POST-PROCESSING FILES.

- Departure file (ICMSH[code]INIT) is read on logical unit number 81.
- If FULL-POS configurations are not of the type 927 (**LFPSPEC**=.F.) post-processing fields are written on logical unit number 54 (file PF[code]DDDDDD+0000), where DDDDDD is the content of the character variable **CFPDOM** of **YOMFPC**.
- If FULL-POS configurations are of the type 927 (**LFPSPEC**=.T.) post-processing fields are written on logical unit number 54 (file PF[code]DDDDDD+0000), and intermediate data corresponding to model fields interpolated in the final geometry at the end of the first part of the FULL-POS run are written on logical unit 91. The second part of the FULL-POS run first reads unit 91, does spectral fit or spectral transformations of some fields and writes them on unit 91; then unit 91 is read as an ARPEGE file like a departure file.
- Departure geometry climatology is read on logical unit 10 (file Const.Clim).
- Arrival geometry climatology is read on logical unit 54 (file Const.Clim.DDDDDD).

2.7 ORGANIGRAM

When not precised, directory is **pp_obs** (except for **LFI...** routines which are in the auxiliary library).

Routines ending by **SM** (ex. **SPECFITASM**, **SPECFITGSM**, **SCAN2MSM**, **SPEREESM**, **REESPESM**, **UVSPESM**, **SPECRTSM**) are written as they are in cycle 18. The suffix **SM** disappears in cycle 17.

2.7.1 Setup routines and arborescence above STEPO.

General architecture under CNT0:

Only features concerning FULL-POS are mentioned.

control/CNT0 →

- **setup/SU0YOMA** →
  - **setup/SUCT0**
  - **setup/SUAFN** →
    - **setup/SUAFN1**
    - **setup/SUAFN2**
  - **setup/SUFPC**
  - **setup/SUDIM** → **setup/SUFPDIM** →
    - **setup/SUFPDYN** → **setup/SUFPDOM**
    - **setup/SUFPCONF**
    - **transform/UPDVPOS** → **setup/SUVPOS**
    - **setup/SUFPFIT**
    - **pp_obs/CPVPOSPR**
  - **setup/SUFPD** → **utility/EGGX** →
    - **utility/EGGRVS**
    - ALADIN routine **EGGMLT**
  - **setup/SUCFU** → **setup/SUCFUFP**
  - **setup/SUXFU** → **setup/SUXFUFP**
  - **setup/SUGPPRP** → **pp_obs/FPINIPHY**
  - **setup/SUAFPOS** →
    - **setup/SURFPDS**
    - **setup/SUWFPDS**
    - **setup/SUAFPDS**
  - **setup/SUALLO** → **setup/SUALFPOS**
- **setup/SU0YOMB** →
  - **setup/SUMP** →
    - **setup/SUALMP2**
    - some other routines which do not contain any FULL-POS feature.
  - **setup/SUFPG** →
    - **setup/SUFPG1** →
      - **setup/DFLTVV1**
      - **setup/SUGAW** (arborescence is not described in detail).
      - **setup/SUNMEN**
    - **setup/SUFPG2** →
      - **utility/EGGX** → **utility/EGGRVS** and ALADIN routine **EGGMLT**
      - **transform/TRACARE**

- **setup/SUGENORD**
- **utility/EGGDIR**
- **utility/EGGRVS**
- **transform/TRARECA**

- **setup/SUSC2** →
  - **setup/SUSIZBUF**
  - ARPEGE: **setup/SUFPCSET** → **setup/SUFPRSET** → processor communication routines **MPE_SEND** and **MPE_RECV**.
  - Some multitasking routines.
  - Some transposition routines (do not touch FULL-POS).
  - Some **MIO...** routines (do not touch FULL-POS).
- **setup/SUBFPOS**
  - **setup/SUFPF**
  - **setup/SUFPOPH** → **setup/SUFRAME** → **FACADE** (or **SUEFRAME** if plane geometry, or **SUFPGRIB** if GRIB files).
  - **setup/SUFPSC2**
  - **setup/SUFPIOS** → **MIOOPEN**
  - **setup/SUFPSC2B**

- **control/CNT1** → (see below arborescence under **CNT1**).

General architecture under CNT1:

Only features concerning FULL-POS are mentioned. For arborescence under **transform/REESPESM**, **transform/SPEREESM**, **transform/UVSPESM** see documentation about spectral transforms. **control/CNT1** →

- **setup/SU1YOM** →
  - **setup/SUINIF** →
    - **pp_obs/SPECFITASM** →
      - **utility/CHIEN** (or the ALADIN routine **CCHIEN** if ALADIN).
      - **LFI...** and **FA...** routines.
      - **transform/REESPESM** (ARPEGE) or **transform/EREESPE** (ALADIN)
      - **transform/SPEREESM** (ARPEGE) or **transform/ESPEREE** (ALADIN)
      - **transform/UVSPESM**
    - **pp_obs/SPECFITADM** →
      - **utility/CHIEN** or **utility/CCHIEN**
      - **LFI...** and **FA...** routines.
      - processor communication routines **parallel/DISGRID**, **parallel/DIWRGRID**, **parallel/DIWRSPE**, **parallel/FPDISPVECTOR**, which call **MPE_SEND** and **MPE_RECV**.
      - **transform/REESPESM** (ARPEGE) or **transform/EREESPEDM** (ALADIN)
      - **transform/SPEREESM** (ARPEGE) or **transform/ESPEREEDM** (ALADIN)
      - **transform/UVSPESM**
    - **pp_obs/SPECFITGSM** (GRIB version of **SPECFITASM**, arborescence not detailed)
    - **pp_obs/SPECFITGDM** (GRIB version of **SPECFITADM**, arborescence not detailed)

- **setup**/**SUSPEC**, **setup**/**SUGRIDF**, **setup**/**SUGRIDU**, **adiab**/**SPECRTSM**, **adiab**/**SPECRTDM**, **setup**/**SUGRCFU**, **setup**/**SUGRXFU** (arborescence not detailed)
- other routines without any relationship with FULL-POS.

- **control**/**CNT2** → **control**/**CNT3** → (see below arborescence under **CNT3**).

General architecture under CNT3:

Only features concerning FULL-POS are mentioned. **control**/**CNT3** →

- **setup**/**SU3FPOS** →
  - ARPEGE: **setup**/**SUHOW1**
  - **setup**/**SURFPBUF** →
    - **pp_obs**/**CPCLIMO** →
      - ARPEGE: **setup**/**SUHOW2**
      - **pp_obs**/**FPSCAW**
      - **utility**/**SC2RDG** → **utility**/**EMPTB** and "MIO" routines.
      - **parallel**/**FPADD3P** (DM)
      - **parallel**/**FPWRIT** (DM) → processor communication routine **MPE_SEND**.
      - **parallel**/**FPREAD** (DM) → processor communication routine **MPE_RECV**.
      - **parallel**/**FPEXTPOL** (DM)
      - **adiab**/**EXTMERB** (no DM)
      - **adiab**/**INIPEX** (no DM)
      - **adiab**/**EXTPOLB** (no DM)
      - **pp_obs**/**FPINT12**
      - "MIO" routines.
    - **c9xx**/**RDCLIMO** → **utility**/**CHIEN**, "LFI" end "FA" routines.
    - **c9xx**/**RDECCLIMO** (GRIB version of **RDCLIMO**) → GRIB routines.
    - **utility**/**SC2WRG** → **utility**/**FILLB**
    - processor communication routines **parallel**/**FPSENDTOPROC1**, **parallel**/**FPDISPTOPROC1**, which call **MPE_SEND** and **MPE_RECV** for cycle 17 (do not work properly).
    - processor communication routines **parallel**/**OSNDGPFFP**, **parallel**/**ORCVGPFFP**, **parallel**/**ISNDGPFFP**, **parallel**/**IRCVGPFFP**, which call **MPE_SEND** and **MPE_RECV**; processor communication routine **MPE_BARRIER** for cycle 18.
  - **setup**/**SUWFPBUF** →
    - **pp_obs**/**CPCLIMI** →
      - **pp_obs**/**FPSCAW**
      - **utility**/**SC2RDG** → **utility**/**EMPTB** and "MIO" routines.
      - **parallel**/**FPADD3P** (DM)
      - **parallel**/**FPWRIT** (DM) → processor communication routine **MPE_SEND**.
      - **parallel**/**FPREAD** (DM) → processor communication routine **MPE_RECV**.
      - **parallel**/**FPEXTPOL** (DM)
      - **adiab**/**EXTMERB** (no DM)
      - **adiab**/**INIPEX** (no DM)
      - **adiab**/**EXTPOLB** (no DM)

        - "MIO" routines.
    - **pp_obs**/**CPCLIMO** →
        - ARPEGE: **setup**/**SUHOW2**
        - **pp_obs**/**FPSCAW**
        - **utility**/**SC2RDG** → **utility**/**EMPTB** and "MIO" routines.
        - **parallel**/**FPADD3P** (DM)
        - **parallel**/**FPWRIT** (DM) → processor communication routine **MPE_SEND**.
        - **parallel**/**FPREAD** (DM) → processor communication routine **MPE_RECV**.
        - **parallel**/**FPEXTPOL** (DM)
        - **adiab**/**EXTMERB** (no DM)
        - **adiab**/**INIPEX** (no DM)
        - **adiab**/**EXTPOLB** (no DM)
        - **pp_obs**/**FPINT12**
        - "MIO" routines.
    - **utility**/**EXTGPF** → (**utility**/**SC2RDG** → **utility**/**EMPTB**) and "MIO" routines.
    - **c9xx**/**RDCLIMO** → **utility**/**CHIEN**, "LFI" end "FA" routines.
    - **c9xx**/**RDECCLIMO** (GRIB version of **RDCLIMO**) → GRIB routines.
    - processor communication routine **parallel**/**FPDISPTOPROC1**, which calls **MPE_SEND** and **MPE_RECV** for cycle 17 (do not work properly).
    - processor communication routines **parallel**/**ISNDGPFFP**, **parallel**/**IRCVGPFFP**, which call **MPE_SEND** and **MPE_RECV**; processor communication routine **MPE_BARRIER** for cycle 18.
    - ARPEGE: **setup**/**SUHOW2**
    - ARPEGE: **setup**/**SUHOWLSM**
    - "MIO" routines.
    - **utility**/**SC2WRG** → **utility**/**FILLB**
  - **pp_obs**/**FPFILTER**
- **control**/**CNT4** → (see below arborescence under **CNT4**).
- **setup**/**SU4FPOS** →
  - **pp_obs**/**PPREQ**
  - **setup**/**SUFPPHY** → **setup**/**SUFPDOM**
  - **setup**/**SUFPDYN** → **setup**/**SUFPDOM**
- **control**/**CPREP4** → (see below arborescence under **CPREP4**).

Remark: **SU4FPOS** and **CPREP4** are also called by **var**/**ARCHFP** when using FULL-POS mode 927 to truncate the trajectory from high resolution to low resolution.

General architecture under CNT4:

Only features concerning FULL-POS are mentioned. **control**/**CNT4** →

- **setup**/**SU4FPOS** →
  - **pp_obs**/**PPREQ**
  - **setup**/**SUFPPHY** → **setup**/**SUFPDOM**
  - **setup**/**SUFPDYN** → **setup**/**SUFPDOM**
- **pp_obs**/**GRIDFPOS** → (see below arborescence under **GRIDFPOS**).

- **pp_obs/DYNFPOS** → (see below arborescence under **DYNFPOS**).
- **control/CPREP4** → (see below arborescence under **CPREP4**).

General architecture under GRIDFPOS:

Only features concerning FULL-POS are mentioned. **pp_obs/GRIDFPOS** →

- **utility/CPGRIDF** →
  - **utility/SC2RDG** → **utility/EMPTB**
  - **utility/SC2WRG** → **utility/FILLB**
  - some "MIO" routines.
- **utility/PKGRIDA** (or **PKGRIDG** if use of GRIB files, not described in detail) →
  - some "FA" and "LFI" routines.
  - processor communication routines **parallel/FPSENDTOPROC2**, **parallel/FPDISPTOPROC2**, which call **MPE_SEND** and **MPE_RECV** (do not work properly).
  - **utility/SC2RDG** → **utility/EMPTB**
  - **utility/SC2WRG** → **utility/FILLB**
  - **utility/EXTGPF** (no DM) → (**utility/SC2RDG** → **utility/EMPTB**) and "MIO" routines.
  - **utility/INCGPF** (no DM) →
    - **utility/SC2RDG** → **utility/EMPTB**
    - **utility/SC2WRG** → **utility/FILLB**
    - some "MIO" routines.
- **pp_obs/FPMODCFU** → **pp_obs/FPMODPREC** →
  - **utility/SC2RDG** → **utility/EMPTB**
  - **utility/SC2WRG** → **utility/FILLB**
  - **utility/EXTGPF** (no DM) → (**utility/SC2RDG** → **utility/EMPTB**) and "MIO" routines.
  - **utility/INCGPF** (no DM) →
    - **utility/SC2RDG** → **utility/EMPTB**
    - **utility/SC2WRG** → **utility/FILLB**
    - some "MIO" routines.
  - **parallel/FPADD3P** (DM)
  - **parallel/FPREAD** (DM) → processor communication routine **MPE_RECV**.
  - **parallel/FPWRIT** (DM) → processor communication routine **MPE_SEND**.
  - **parallel/FPEXTPOL** (DM)
  - **adiab/EXTMERB** (no DM)
  - **adiab/INIPEX** (no DM)
  - **adiab/EXTPOLB** (no DM)
  - **pp_obs/FPSCAW2**
  - **pp_obs/FPOLIS**
  - some "MIO" routines.
- **pp_obs/FPMODXFU** → **pp_obs/FPMODPREC** →
  - **utility/SC2RDG** → **utility/EMPTB**
  - **utility/SC2WRG** → **utility/FILLB**
  - **utility/EXTGPF** (no DM) → (**utility/SC2RDG** → **utility/EMPTB**) and "MIO" routines.
  - **utility/INCGPF** (no DM) →
    - **utility/SC2RDG** → **utility/EMPTB**

        - **utility/SC2WRG** → **utility/FILLB**
        - some "MIO" routines.
    - **parallel/FPADD3P** (DM)
    - **parallel/FPREAD** (DM) → processor communication routine **MPE_RECV**.
    - **parallel/FPWRIT** (DM) → processor communication routine **MPE_SEND**.
    - **parallel/FPEXTPOL** (DM)
    - **adiab/EXTMERB** (no DM)
    - **adiab/INIPEX** (no DM)
    - **adiab/EXTPOLB** (no DM)
    - **pp_obs/FPSCAW2**
    - **pp_obs/FPOLIS**
    - some "MIO" routines.
- **utility/MAXGPFV** →
    - processor communication routines **parallel/FPSENDTOPROC2**, **parallel/FPDISPVECTOR**, which call **MPE_SEND** and **MPE_RECV** for cycle 17 (do not work properly).
    - processor communication routines **parallel/OSNDGPF**, **parallel/ORCVGPF** which call **MPE_SEND** and **MPE_RECV**; processor communication routines **MPE_SEND**, **MPE_RECV** and **MPE_BARRIER** for cycle 18.
    - **utility/SC2RDG** → **utility/EMPTB**
    - **utility/EXTGPF** (no DM) → (**utility/SC2RDG** → **utility/EMPTB**) and "MIO" routines.
- **control/SCAN2H** → **control/SCAN2MSM** (SM) or **control/SCAN2MDM** (DM) → (see below arborescence under **CPREP4**).
- **dia/WRHFP** (**WRMLFPL** or **WRMLFP** if use of GRIB files, not described in detail) →
    - some "FA" and "LFI" routines.
    - **utility/EXTFPF** → (**utility/SC2RDG** → **utility/EMPTB**) and "MIO" routines.
    - processor communication routines **parallel/FPSENDTOPROC1**, which calls **MPE_SEND** and **MPE_RECV** (does not work properly).
    - **pp_obs/FPBIPER** → library routines **ESPLIN**, **ESMOOTH** and **ESHFT**.
    - **dia/FPNORM**

General architecture under DYNFPOS:

Only features concerning FULL-POS are mentioned.

pp_obs/DYNFPOS →

- **setup/SUFPCONF**
- **setup/SUVFPOS** →
    - **transform/UPDVPOS** → **setup/SUVPOS**
    - **setup/SUFPFIT**
    - **pp_obs/CPVPOSPR**
    - **setup/CPFPDS** → **setup/FPFINDS** → **setup/SUFPDS**
- **setup/SUVFPOSL**
- **control/STEPO** → (see below arborescence under **STEPO**).

General architecture under CPREP4:

Only features concerning FULL-POS are mentioned. **CPREP4** is divided in two parts calling the same routines.

control/CPREP4 →

- **control/STEPO** → (see below arborescence under **STEPO**).
- **pp_obs/GRIDFPOS** → (see above arborescence under **GRIDFPOS**).
- **pp_obs/DYNFPOS** → (see above arborescence under **DYNFPOS**).
- **utility/ENDIOS** → several "MIO" routines and library routine **HPSHRINK**
- **utility/MATCLOSE** → "LFI" routine **LFIFER**.
- **utility/FREEMEM** → several **utility/DEAL...** routines (**DEALLO**, **DEALFPOS**, **DEALDDH**, **DEALCOS**, **DEALCTV**, **DEALGES**, **DEALCAN**, **DEALSPA**, **DEALSC2**, **DEALSCR**, **DEALMOD**, **DEALNMI**) and **utility/DEELLO** for ALADIN.
- **control/CNT0** → (see above arborescence under **CNT0**).

General architecture under STEPO:

Only features concerning FULL-POS are mentioned.

control/STEPO →

- Gestion of IO: **utility/IOPACK** (see below arborescence under **IOPACK**).
- Inverse Legendre transforms: **transform/LTINVH**
- **utility/CPGRIDF** →
  - **utility/SC2RDG** → **utility/EMPTB**
  - **utility/SC2WRG** → **utility/FILLB**
  - some "MIO" routines.
- Inverse Fourier transforms + grid point computations + direct Fourier transforms: **control/SCAN2H** → **control/SCAN2MSM** (SM) or **control/SCAN2MDM** (DM) → : (see below arborescence under **SCAN2MSM/SCAN2MDM**).
- Direct Legendre transforms: **transform/LTDIRH**
- Spectral computations: **control/SPCH** (see below).

General architecture under IOPACK:

Part 6 of **IOPACK** concerns FULL-POS.

utility/IOPACK →

- **dia/WRPLFP** for GRIB files, not described in detail.
- **dia/WRPVLFP** for GRIB files, not described in detail.
- **dia/WRTHLFP** for GRIB files, not described in detail.
- **dia/WRMLFP** for GRIB files, not described in detail.
- **dia/ESPFP** (for ALADIN)
- **dia/WRSFP** →
  - "LFI" and "FA" routines.
  - processor communication routine **parallel/DIWRSPE**, **parallel/FPSENDTOPROC2**, which call **MPE_SEND** and **MPE_RECV** (do not properly work).
  - **dia/SPNORMBL** for ARPEGE (arborescence not described in detail)
  - **dia/SPNORMAVE** for ARPEGE (arborescence not described in detail)
  - **dia/ESPNORMBL** for ALADIN (arborescence not described in detail)
  - **utility/SC2RDG** → **utility/EMPTB**
  - **utility/EXTGPF** → (**utility/SC2RDG** → **utility/EMPTB**) and "MIO" routines.
- **dia/WRSFPMIO** →
  - "LFI" and "FA" routines.
  - **dia/SPNORMBL** (arborescence not described in detail)

- **dia/SPNORMAVE** (arborescence not described in detail)
- **transform/SPEREESM** (arborescence not described in detail, see documentation about spectral transforms).
- **transform/REESPESM** (arborescence not described in detail, see documentation about spectral transforms).
- **utility/EXTGPF** → (**utility/SC2RDG** → **utility/EMPTB**) and "MIO" routines.
- **utility/FILLB**
- **utility/WRSPBUF** → "MIO" routines.
- "MIO" routines.

- **dia/WRHFP** →
  - some "FA" and "LFI" routines.
  - processor communication routine **parallel/FPSENDTOPROC1**, which calls **MPE_SEND** and **MPE_RECV**.
  - **utility/EXTFPF** → (**utility/SC2RDG** → **utility/EMPTB**) and "MIO" routines.
  - **pp_obs/FPBIPER** → library routines **ESPLIN**, **ESMOOTH** and **ESHFT**.
  - **dia/FPNORM**

2.7.2 Grid-point computations.

General architecture of SCAN2MSM (shared memory) and SCAN2MDM (distributed memory):

control/SCAN2H → **control/SCAN2MSM** (SM) or **control/SCAN2MDM** (DM) → :

- Inverse Fourier transforms: **transform/FTINVH**.
- Some memory transfers and pointer computations before grid-point computations.
- Comparison with observations (non-lagged part, then information communication between processors, then lagged part).
- Model grid-point computations (non-lagged part, then information communication between processors, then lagged part).
- Post-processing (grid-point part, see below for more details).
- Grid-point computations for analysis.
- Direct Fourier transforms: **transform/FTDIRH**.

General architecture concerning FULL-POS in SCAN2MSM and SCAN2MDM:

FULL-POS under **control/SCAN2MSM** or **control/SCAN2MDM** includes:

- Vertical interpolations. **VPOS** →
  - **utility/SC2RDG** → **utility/EMPTB**
  - **CPVPOSR**
  - **POS** → (see below).
  - **utility/SC2WRG** → **utility/FILLB**
  - **GPOS** → **adiab/GPPRE** and **CTSTAR**.
- Non lagged part of horizontal interpolations. **HPOS** →
  - Function **utility/MGETBUF**
  - **utility/SC2RDG** → **utility/EMPTB**
  - **adiab/EXTMERB** (no DM)
  - **adiab/INIPEX** (no DM)
  - **adiab/EXTPOLB** (no DM)
- For distributed memory, some routines doing halo constitution and communications between processors before performing interpolations under **HPOSLAG**.

- **parallel/FPADD3P**
- **parallel/FPWRIT** → processor communication routine **MPE_SEND**.
- **parallel/FPREAD** → processor communication routine **MPE_RECV**.
- **parallel/FPEXTPOL**

- Lagged part of horizontal interpolations and some vertical interpolations made after horizontal interpolations. **HPOSLAG** →
  - Function **utility/MFINDBUF**
  - **FPOSHOR** → (see below)
  - Function **utility/MDONEBUF**
- Diagnostic of the post-processable variables which are computed from the post-processed model variables (height or η-levels post-processing and change of horizontal geometry). **ENDVPOS** →
  - **utility/SC2RDG** → **utility/EMPTB**
  - **CPVPOSPR**
  - **ENDPOS** →
    - Several **adiab/GP...** routines: **GPRCP**, **GPPREH**, **GPXYB**, **GPPREF**, **GPGEO**, **GPRH**, **GPTET**, **GPEPT**.
    - **PPTHPW**
  - **utility/SC2WRG** → **utility/FILLB**

Organigram of POS:

- Several **adiab/GP...** routines: **GPRCP**, **GPPREH**, **GPXYB**, **GPPREF**, **GPGEO**, **GPPVO**, **GPEPT**.
- **CTSTAR**

*** A series of routines computing intermediate quantities and weights for vertical interpolations.

- **PPINITZ**
- **PPZHLEV**
- **PPPS** → **PPINTZ** and **EXPBESU**
- **PPLTP**
- **PPLTETA**
- **PPLETA** → **adiab/GPPREH**, **adiab/GPXYB** and **adiab/GPPREF**
- **PPINIT**
- **PPFLEV**

*** A series of interpolation routines.

- **PPUV** →
- **PPUV_OLD** → **PPINTP** and **PPITPQ**
- **PPINTP**
- **(PPITPQ)** (currently commented).
- **PPT** →
  - **PPT_OLD** → **PPINTP**
  - **PPSTA**
  - **PPINTP**
- **PPQ** → **PPINTP**
- **c9xx/APACHE** → (see below)
- **PPGEOP** →
  - **PPGEOP_OLD** → **PPSTA**, **adiab/GPGEO**, **PPITPQ** and **PPINTP**.
  - **PPSTA**

        - **adiab/GPGEO**
    - **PP2DINT**
    - **PPRH** → **adiab/GPRH** and **PPINTP**
    - **adiab/GPRH**
    - **PPVVEL** → **adiab/GPCTY** and **PPINTP**
    - **adiab/GPCTY**
    - **PPPMER**
    - **PPLTP**
    - **PPTHPW**
    - **POAERO** →
        - **TJQUD**
        - **TJCUBI**
        - **TJQUAA**
    - Storage of post-processed fields.

Organigram of FPOSHOR:

- **utility/SC2RDG** → **utility/EMPTB**
- **FPSCAW**
- **FPSURF** → **FPINT12**
- **FPINTDYN** → **FPINT12**, **FPHOR12** and **FPAERO**
- **FPINTPHY** → **FPINT12** and **FPMIMAX**
- **FPGEO**
- **FPVERT** →
    - **CPVPOSPR**
    - **FPPS** →
        - **adiab/GPRCP**
        - **adiab/GPPREH**, **adiab/GPXYB** and **adiab/GPPREF**
        - **adiab/GPGEO**
        - **PPINITZ**
        - **PPZHLEV**
        - **PPPS** → **PPINTZ** and **EXPBESU**
        - **adiab/GPPRE**
    - **c9xx/APACHE** → (see below)
    - **utility/SC2WRG** → **utility/FILLB**
- **utility/SC2WRG** → **utility/FILLB**

Organigram of c9xx/APACHE:

- Several **adiab/GP...** routines: **GPPREH**, **GPXYB**, **GPPREF**, **GPRCP** and **GPPRE**
- **PPINIT**
- **c9xx/VEINE** → **adiab/GPGEO**, **CTSTAR**, **PPPMER**, **adiab/GPPREH**, **adiab/GPXYB** and **adiab/GPPREF**
- **c9xx/AVAL** →
    - **CTSTAR**
    - **PPFLEV**
    - **PPT** →
        - **PPT_OLD** → **PPINTP**
        - **PPSTA**
        - **PPINTP**

- **PPQ** → **PPINTP**
- **PPRH** → **adiab**/**GPRH** and **PPINTP**
- **PPUV** →
  - **PPUV_OLD** → **PPINTP** and **PPITPQ**
  - **PPINTP**
  - **(PPITPQ)** (currently commented).
- **PPGEOP** →
  - **PPGEOP_OLD** → **PPSTA**, **adiab**/**GPGEO**, **PPITPQ** and **PPINTP**.
  - **PPSTA**
  - **adiab**/**GPGEO**
- Several **adiab**/**GP...** routines: **GPPREH**, **GPXYB**, **GPPREF** and **GPRCP**
- **c9xx/BOB** → **PPINTP**

2.7.3 Spectral transforms.

See documentation about spectral transforms of model fields. There are additional routines to treat FULL-POS fields (**transform**/**SC2FPFSC**, **transform**/**UPDSPFP**, **pp_obs**/**PRFIP**, **parallel**/**SUFPTRDIR**, **parallel**/**SUFP-TRINV**, are called instead of **transform**/**SC2FSC**, **transform**/**UPDSP**, **transform**/**PRFI1**) **parallel**/**SUTRDIR**, **parallel**/**SUTRINV**.

2.7.4 Spectral computations.

control/**STEPO** →

- **control**/**SPCH** → **control**/**SPCM** → **pp_obs**/**SPOS** → **utility**/**MXMAOP**.

2.7.5 Action and brief description of each routine.

- Expression "full level" is synonym of "middle of layer".
- Expression "half level" is synonym of "interlayer".
- For meaning of [L5] see section "Sequences of calls of post-processing".
- Expression "DM-local" for a quantity means "local to the couple of processors (*proca*,*procb*)": each processor has its own value for the quantity. Expression "DM-local computations" means that the computations are made independently in each processor on "DM-local" quantities, leading to results internal to each processor, which can be different from a processor to another one.
- Expression "DM-global" for a quantity means that it has a unic value available in all the processors. Expression "DM-global computations" means that the computations are either made in one processor, then the results are dispatched in all the processors, or the same computations are made in all the processors, leading to the same results in all the processors.
- In a routine description the mention "For distributed memory computations are DM-local" means that all calculations made by this routine are DM-local; the mention "For distributed memory computations are DM-global" means that all calculations made by this routine are DM-global; when no information is provided it means that a part of calculations are DM-local and the other part is DM-global.
- Expression "main" processor currently refers to the processor number 1: (*proca*,*procb*)=(1,1).

Grid-point and spectral routines of directory "adiab":

- **EXTMERB**: computes fields for extra-longitudes 0, **NLOENG**(*jgl*+1) and **NLOENG**(*jgl*+2). This routine is used only for shared memory.
- **EXTPOLB**: computes fields for extra-polar latitudes. This routine is used only for shared memory.

- **GP...** routines computing some dynamical grid-point quantities; for distributed memory computations are DM-local.
  - **GPCTY**: computes quantities linked to vertical velocities at half levels.
  - **GPEPT**: computes the equivalent potential temperature at full levels. Is currently in directory **pp_obs** but has to be moved later in directory **adiab**.
  - **GPGEO**: computes geopotential at half and full levels.
  - **GPPRE**: computes hydrostatic pressures at half and full levels.
  - **GPPREF**: computes hydrostatic pressures at full levels.
  - **GPPREH**: computes hydrostatic pressures at half levels.
  - **GPPVO**: computes potential vorticity and potential temperature at full levels.
  - **GPRCP**: computes *cp*, *R* and \frac *Rcp* at full levels.
  - **GPRH**: computes saturation vapour pressure and relative humidity from temperature and specific humidity at full levels.
  - **GPTET**: computes potential temperature at full levels.
  - **GPXYB**: computes auxiliary quantities related to the hybrid cooordinate, for example pressure depths, coefficients to compute vertical integrals, etc.
- **INIPEX**: computes intermediate quantities for call to **EXTPOLB** (extra-polar latitudes indices where information has to be copied). This routine is used only for shared memory.
- **SPECRTSM**: spectral space conversions between real temperature and virtual temperature for shared memory.
- **SPECRTDM**: spectral space conversions between real temperature and virtual temperature for distributed memory.

Routines of directory "c9xx".

- **APACHE**: interface routine for some vertical interpolations, called if post-processing on height or η -levels and change of horizontal geometry. For distributed memory computations are DM-local.
- **AVAL**: inner interface routine for some vertical interpolations. For distributed memory computations are DM-local.
- **BOB**: vertical interpolation of the *B*-coefficients of the vertical hybrid coordinate, to the levels of post-processing. For distributed memory computations are DM-local.
- **RDCLIMO** (**RDECCLIMO** at ECMWF if GRIB files used): opens output climatology files and reads the needed fields. In case of a file ARPEGE, poles values are not read. For distributed memory computations are DM-global.
- **VEINE**: vertical interpolations linked to difference of orographies between two systems. For distributed memory computations are DM-local.

Control routines of directory "control":

- **CNT0**: controls integration job at level 0.
- **CNT1**: controls integration job at level 1.
- **CNT2**: controls integration job at level 2.
- **CNT3**: controls integration job at level 3.
- **CNT4**: controls integration job at level 4.
- **CPREP4**: control routine called only if post-processing needs a change of horizontal geometry and if spectral fit or computations are made in the arrival geometry: pseudo configurations 927 (for example file ARPEGE $\rightarrow$ file ARPEGE), E927 (for example file ALADIN $\rightarrow$ file ALADIN). Calls twice **CNT0** after other routines, the first call of **CNT0** to manage post-processing computations in the arrival geometry and the second call to **CNT0** if required to return to the departure (model) geometry.

- **SCAN2H**: control routine for grid-point computations.
- **SCAN2MSM**: multitasking interface for grid-point computations for shared memory calculations.
- **SCAN2MDM**: interface for grid-point computations for distributed memory calculations.
- **SPCH**: control routine for spectral computations.
- **SPCM**: multitasking interface for spectral computations.
- **STEPO**: control routine for one time integration step.

Routines of directory "dia".

- **ESPFP**: ALADIN routine writing vertical post-processed data on a FA file, equivalent of **WRSFP**.
- **ESPNORMBL**: ALADIN version of routine (**SPNORMBL**+**SPNORMAVE**).
- **FPNORM**: writes out norms on the horizontally post-processed fields (including mean value and extrema). For distributed memory this routine is not parallelised and can do computations for all data on one processor only, so data have to be previously collected on one processor.
- **SPNORMAVE**: computes the average of norms in spectral space for each level and for all levels, uses data computed in **SPNORMBL**. For distributed memory computations are DM-global.
- **SPNORMBL**: computes norms in spectral space, multitasking interface. For distributed memory this routine is not parallelised and can do computations for all data on one processor only, so data have to be previously collected on one processor.
- **WRHFP**: writes out the horizontally post-processed fields, sorts out the output points for each domain, then writes records on files. Only grid-point data are written. If distributed memory, each processor reads its own data in buffers **GDFPBUF** and **GPFPBUF**, then all data are communicated to the "main" processor. Operation such as norm computation, polar value computation, biperiodicisation (for ALADIN domain), writing on ARPEGE or ALADIN file, are made in the "main" processor for all data.
- **WRHFPG**: GRIB version of **WRHFP**. No longer used in cycle 18.
- **WRMLFP**: GRIB routine used instead of **WRHFPG** in some cases (especially in FULL-POS configurations other than 927-type ones). Writes data on model layers.
- **WRPLFP**: GRIB routine used instead of **WRHFPG** in some cases (especially in FULL-POS configurations other than 927-type ones). Writes data on pressure layers.
- **WRPVLFP**: GRIB routine. Writes data on potential vorticity layers.
- **WRSFP**: writes out the vertically post-processed dynamical fields to ARPEGE/ALADIN file. Spectral fields are written out as spectral coefficients (if spectral output is wanted), grid-point fields are written out as grid-point fields. The horizontal grid is the model one. If distributed memory, each processor reads its own data in buffers **GAUXBUF**, then all data are communicated to the "main" processor. Operation such as norm computation, polar value computation, biperiodicisation (for ALADIN domain), writing on ARPEGE or ALADIN file, are made in the "main" processor for all data.
- **WRSFPG**: GRIB version of **WRSFP**. No longer called in cycle 18.
- **WRSFPMIO**: writes out the vertically post-processed dynamical fields at the target resolution on a MIO file for the upper air fields, and on standard FULL-POS files for the surface fields. Spectral fields are written out as spectral coefficients. This routine is called only when FULL-POS is used in a variational analysis. This routine is not called in a configuration 1, and is not completely parallelised in the cycles 17 and 18.
- **WRTHLFP**: GRIB routine. Writes data on potential temperature layers.

Distributed memory environment routines (directory "parallel"):

- **FPADD3P**: memory transfer routine which adds extra-longitudes points: one assumes that there are one western extra-longitude and 2 eastern extra-longitudes. Equivalent of the old routine

SLADD3P (no longer used in cycles 17 and 18) previously used in the semi-Lagrangian scheme or the observation interpolator but works with a width band of **NFPWIDE** instead of **NSLWIDE**.

- **FPDISPTOPROC1**: communication between processors; dispatches data of several fields stored in a DM-global array from the "main" processor and for each processor stores data in a DM-local array. For each field the number of grid-points to dispatch is not necessary equal to the number of model grid points.
- **FPDISPTOPROC2**: communication between processors; particular case of **FPDISPTOPROC1** when the number of grid-points to dispatch is equal to the number of model grid points.
- **FPDISPVECTOR**: communication between processors; communicates a scalar or a vector to all processors.
- **FPEXTPOL**: fills the halo of width band **NFPWIDE** for non polar extra-polar latitudes. Equivalent to the routine **SLEXTPOL** used in the semi-Lagrangian scheme or the observation interpolator but works with a width band of **NFPWIDE** instead of **NSLWIDE**. Coded only without points at the poles.
- **FPREAD**: communication between processors; receives data from the other processors to constitute the halo of width band **NFPWIDE**. Equivalent of the old routine **SLREAD** (no longer used in cycles 17 and 18) previously used in the semi-Lagrangian scheme or the observation interpolator but works with a width band of **NFPWIDE** instead of **NSLWIDE**.
- **FPSENDTOPROC1**: communication between processors; sends data of several fields stored in a DM-local array to the "main" processor and collects data in the "main" processor in a DM-global array. For each field the number of grid-points to send is not necessary equal to the number of model grid points.
- **FPSENDTOPROC2**: communication between processors; particular case of **FPSENDTOPROC1** when the number of grid-points to send is equal to the number of model grid points.
- **FPWRIT**: communication between processors; sends data to the other processors to constitute the halo of width band **NFPWIDE**. Equivalent to the old routine **SLWRIT** (no longer used in cycles 17 and 18) previously used in the semi-Lagrangian scheme or the observation interpolator but works with a width band of **NFPWIDE** instead of **NSLWIDE**.
- **IRCVGPF**: routine to receive a distributed model grid point field from another processor. DM-local fields are dimensioned with **NGPTOT** (=**NBDYSZ** for DM), DM-global fields are dimensioned with **NGPTOTG**.
- **IRCVGPFFP**: routine to receive a distributed FULL-POS grid point field from another processor. DM-local fields are dimensioned with **NFPRGPL**, DM-global fields are dimensioned with **NFPRGPG**.
- **ISNDGPF**: routine to send a distributed model grid point field to other processors. DM-local fields are dimensioned with **NGPTOT** (=**NBDYSZ** for DM), DM-global fields are dimensioned with **NGPTOTG**.
- **ISNDGPFFP**: routine to send a distributed FULL-POS grid point field to other processors. DM-local fields are dimensioned with **NFPRGPL**, DM-global fields are dimensioned with **NFPRGPG**.
- **MPE_BARRIER**: auxiliary library routine for communication between processors: waits for the end of communication between all processors before starting another action.
- **MPE_RECV**: auxiliary library routine for communication between processors: reception of data from other processors.
- **MPE_SEND**: auxiliary library routine for communication between processors: sends data to other processors.
- **SUFPTRDIR**: initialise data structures for transposition of grid point data from column structure to latitudinal, for FULL-POS post-processed fields which are spectrally fitted.

- **SUFPTRINV**: initialise data structures for transposition of grid point data from latitudinal structure to column, for FULL-POS post-processed fields which are spectrally fitted.

Remark: routines **FPDISPTOPROC1**, **FPDISPTOPROC2**, **FPDISPVECTOR**, **FPSENDTOPROC1** and **FPSENDTOPROC2** are not validated (still bugged) and are close to the **ORCV...**, **OSND...**, **IRCV...**, **ISND...** communication routines used at ECMWF, it is planned to use only the last ones in a future cycle.

Routines of directory "pp_obs".

- **CPCLIMI**: fills interpolation buffers with some surface fields (land-sea mask) previous to compute weights with land-sea mask.
- **CPCLIMO**: computes output climatology from input one (orography and land-sea mask): extracts fields from model buffer/work file or array, then performs horizontal interpolations.
- **CPVPOSPR**: computes field pointers for vertical post-processing. Computes the pointers of the post-processing fields so that the fields are stored one after the others in the arrays GT0 (fitted fields) and GAUX (unfitted fields). The fields are stored in the following order: 1) underived 3D fields; 2) underived 2D fields; 3) derived 3D fields; 4) derived 2D fields.
- **CTSTAR**: computes the standard surface temperature and the surface temperature to be used for extrapolations of temperature and geopotential. For distributed memory computations are DM-local.
- **DYNFPOS**: interface routine managing vertical then horizontal interpolations for post-processing on dynamical 3D and 2D fields, calls several sequences of **STEPO**.
- **ENDPOS**: finish post-processing calculations for output height levels or η -levels. For distributed memory computations are DM-local.
- **ENDVPOS**: interface routine managing all post-processing computations which can be done only after horizontal interpolations. Called if dynamical variables other than model variables are post-processed on height or η -levels. For distributed memory computations are DM-local.
- **EXPBESU**: extrapolates pressure below surface. For distributed memory computations are DM-local.
- **FPAERO**: 12 points horizontal interpolations or taking nearest available value if data are missing. Used for aeronautic jet. For distributed memory computations are DM-local.
- **FPBIPER**: FULL-POS interface for double periodicisation (ALADIN). Calls the following library externals: **ESHFT** (ensuring correct results when conservation domain is shifted), **ESPLIN** (spline extension), **ESMOOTH** (smoothing across to get isotropy). For distributed memory this routine is not parallelised and can do computations for all data on one processor only, so data have to be previously collected on one processor.
- **FPFILTER**: set-up routine which computes a matrical operator which is a product of a transformation from computational sphere to "equivalent" not dilated sphere, a "THX"-filtering and the inverse transformation from "equivalent" not dilated sphere to computational sphere. The operator is stored in the array **RFPMAT**. Sphere to sphere transformations are made by reading a LFI file containing dilatation/contraction matrixes which can be initialised by a configuration 911 (see the ad-hoc documentation). **FPFILTER** is called only in the following cases:
  - filtering in spectral space activated, **LFPFIL**=.TRUE. (if ARPEGE) or **LFPBED**=.TRUE. (if ALADIN) in **NAMFPF**.
  - the geometry where the spectral fit is done is a stretched one: for example the initial geometry if not 927-type FULL-POS configurations, and **CFPFMT** is not 'MODEL'.
  - post-processing for any variable if potential vorticity vertical coordinate.
  - post-processing of horizontal derivatives (vorticity, divergence, vertical velocity, deformations) for other vertical coordinates.

For distributed memory computations are DM-global.

- **FPGEO**: transforms the post-processed fields into the output geometry. For distributed memory computations are DM-local.
- **FPHOR12**: 12-points horizontal interpolations for derivatives: interpolations are done subdomain by subdomain. For distributed memory computations are DM-local.
- **FPINIPHY**: controls that the pointers of physical fields to be post-processed are defined.
- **FPINTDYN**: interface for 12-points horizontal interpolations, for fields coming from the dynamics. For distributed memory computations are DM-local.
- **FPINTPHY**: interface for 12-points horizontal interpolations, for fields coming from the physics. Land-sea mask is taken into account for the surface variables. For distributed memory computations are DM-local.
- **FPINT12**: 12 points horizontal interpolations, with different options: after the 12 points horizontal interpolations, additional corrections can be done according to the values of the dummy arguments **LDAMLY**, **KORR2** and **KORR**.
  - **LDAMLY**=.TRUE.: add to the interpolated field the value of another (already computed) field specified at the interpolation points. This option is used if the field is an anomaly; in this case a correcting field is added to recover the absolute value of the field.
  - (**KORR2**,**KORR**)=(1,1): interpolated value is bounded over zero if land, is reset to zero if sea. Option used for skin reservoir contents,snow depth.
  - (**KORR2**,**KORR**)=(1,2): interpolated value is bounded between **PMIN** and **PMAX**. Option used for soil wetness.
  - (**KORR2**,**KORR**)=(2,1): interpolated value is replaced by a pre-defined value over seas; then all interpolated values are multiplied by a rescaling factor **PSCAL**. Option used for deep soil temperature.
  - (**KORR2**,**KORR**)=(2,2): interpolated value is filtered in order to remain above zero; then it is reset to zero over sea, and also over land if the surface temperature at this point is below a threshold; then all interpolated values are multiplied by a rescaling factor **PSCAL**. Option used for snow cover.
  - (**KORR2**,**KORR**)=(2,3): interpolated value is filtered in order to remain between **PMIN** and **PMAX**; then it is reset to the maximum value **PMAX** over sea; then all interpolated values are multiplied by a rescaling factor **PSCAL**. Option used for wetness.
  - (**KORR2**,**KORR**)=(2,4): interpolated value is filtered in order to remain above **PMIN**. Option used for positive fields.
  - (**KORR2**,**KORR**)=(2,5): interpolated value is reset to 0 if lower than 0.5, and 1 if above 0.5. Option used for fields which can take only the values 0 or 1, like the land-sea mask.
  - (**KORR2**,**KORR**)=(2,6): interpolated value is filtered in order to remain between **PMIN** and **PMAX**. Option used for normed fields.
  - (**KORR2**,**KORR**)=(2,7): add a monotonic constraint to avoid overshoots and undershoots; interpolated value is reset to something between the values of the four adjacent grid-points. This is the same monotonic constraint which is used in the semi-Lagrangian interpolation routine **LAIDQM**. Used for precipitations.

For distributed memory computations are DM-local.

- **FPMIMAX**: 12 points interpolation, search for the nearest point. Used to interpolate the maximum of a field (such as temperature). For distributed memory computations are DM-local.
- **FPMODCFU**: preliminary modification of cumulated fluxes for post-processing: prepares input fields before horizontal interpolations.

- **FPMODPREC**: to modify precipitation fields before horizontal post-processing: to avoid enlargement of the precipitation areas, input fields are transformed so that there are negative values at the points with no precipitations. Output fields will be filtered after horizontal interpolations. This negative value is computed as the opposite of the maximum value of the nearest points: 4 points if the grid is regular; 5 or 6 points is the grid is reduced. Poles rows are ignored in the post-processing. For distributed memory computations are DM-local. The way to modify values is coded like horizontal interpolations, but in this particular case the interpolation points are model grid-points, so weights computations are useless, one only needs to know the grid-point coordinates of the surrounding points.
- **FPMODXFU**: preliminary modification of instantaneous fluxes for post-processing: prepares input fields before horizontal interpolations.
- **FPOLIS**: routine modifying some zero values by negative values according to the positive values of the surrounding points; is coded like an interpolation routine; works only for positive quantities, like rainfall amount. For distributed memory computations are DM-local.
- **FPOSHOR**: inner interface routine managing lagged part of horizontal interpolations. For distributed memory computations are DM-local.
- **FPPS**: interface to **PPPS**, to compute pressures corresponding to heights levels or η -levels of post-processing. For distributed memory computations are DM-local.
- **FPSCAW**: for horizontal interpolations of FULL-POS, finds the indices in interpolation buffers where information of the surrounding model grid points can be found. For distributed memory computations are DM-local.
- **FPSCAW2**: particular case of **FPSCAW**, where the interpolation points are the grid-points themselves, and where interpolations are replaced by value change according to the values of the surrounding points; finds the indices in interpolation buffers where information of the surrounding model grid points can be found. For distributed memory computations are DM-local.
- **FPSURF**: 12 points horizontal interpolations, then altitude correction if necessary. Applied to auxiliary surface fields. For distributed memory computations are DM-local.
- **FPVERT**: interface to **APACHE**, to perform vertical post-processing from model levels to height or η -levels, just after horizontal interpolations to take account of the difference of orography between the departure geometry and the arrival geometry. For distributed memory computations are DM-local.
- **GPOS**: saves in grid-point arrays the spectral fields that will be needed for post-processing on height levels above an outside orography. For distributed memory computations are DM-local.
- **GRIDFPOS**: interface routine managing post-processing of grid point fields. Performs horizontal post-processing on physical fields and fluxes and writes post-processed fields on a file.
- **HPOS**: interface routine managing non lagged part of horizontal interpolations. Works like the sequence **adiab**/**CPG** $\rightarrow$ **adiab**/**LACDYN** $\rightarrow$ (subroutines) for semi-Lagrangian scheme. Fills buffers for quantities to be interpolated (arrays **PFPBUF1** and **PFPBUF2**). **PFPBUF2** contains only surface physical fields, horizontally interpolated fields can be used before interpolations of the other fields stored in buffer **PFPBUF1**. Treats: 1) fitted fields coming from vertical interpolations, 2) unfitted fields coming from vertical interpolations, 3) physical fields and auxiliary surface fields, 4) cumulated fluxes, 5) instantaneous fields. A memory transfer by calling routine **SC2RDG** is generally necessary to get quantities for series 2 to 5. For monotasking or multitasking modes, computations for extra-longitudes are made by calling **EXTMERB** and computations for extra-latitudes are made by calling **EXTPOLB** (option with points at the pole is not coded). For distributed memory computations are DM-local.

- **HPOSLAG**: interface routine managing lagged part of horizontal interpolations and some vertical interpolations made after horizontal interpolations. Works like the sequence **adiab**/**CPGLAG** → **adiab**/**LAPINE** → (subroutines) for semi-Lagrangian scheme, but code is simpler because there is no trajectory computation: points where quantities have to be interpolated are already known, so coordinates and weights for interpolations can be precomputed in a set-up routine. First surface fields are interpolated (if post-processing on height or η-levels), then dynamical fields and physical fields. If post-processing is made on height or η-levels there are additional vertical interpolations to take account of the difference of orography between the departure geometry and the arrival geometry. For distributed memory computations are DM-local.
- **POAERO**: post-processing of jet and tropopause heights. For distributed memory computations are DM-local.
- **POS**: inner interface for vertical post-processing. For distributed memory computations are DM-local.
- **PPFLEV**: finds full or half levels under specified pressures. Called if post-processing on pressure, potential vorticity or potential temperature levels ([L5]='B', 'V' or 'T'). For distributed memory computations are DM-local.
- **PPGEOP**: vertical interpolations on pressure levels for geopotential. For distributed memory computations are DM-local.
- **PPGEOP_OLD**: old version of routine **PPGEOP** used at ECMWF. For distributed memory computations are DM-local.
- **PPINIT**: computes intermediate quantities connected with full levels and half levels pressures, prior to vertical interpolations on pressure or height levels. Called in all cases except the case where post-processing is done on model η-levels ([L5]='M' or 'S'). For distributed memory computations are DM-local.
- **PPINITZ**: computes intermediate quantities connected with full levels and half levels geopotential, prior to vertical interpolations on height levels. Called if post-processing on height levels ([L5]='H'). For distributed memory computations are DM-local.
- **PPINTP**: vertical linear interpolation on pressure levels. For distributed memory computations are DM-local.
- **PPINTZ**: vertical linear interpolation on height levels. For distributed memory computations are DM-local.
- **PPITPQ**: vertical quadratic 3 points interpolation. For distributed memory computations are DM-local.
- **PPLETA**: computes the pressures of post-processing on η-levels. Called if post-processing on η-levels ([L5]='S'). For distributed memory computations are DM-local.
- **PPLTETA**: computes the pressures of post-processing on θ-levels. Called if post-processing on potential temperature levels ([L5]='T'). For distributed memory computations are DM-local.
- **PPLTP**: computes the pressures of post-processing on potential vorticity levels. Called if post-processing on potential vorticity levels ([L5]='V'). For distributed memory computations are DM-local.
- **PPPMER**: computes mean sea level pressure. For distributed memory computations are DM-local.
- **PPPS**: computes surface pressure at a given height, called if post-processing on height levels ([L5]='H'). For distributed memory computations are DM-local.
- **PPQ**: vertical interpolations on pressure levels for humidity, liquid water, ice, cloudiness, passive scalars. For distributed memory computations are DM-local.
- **PPREQ**: search for a suitable post-processing namelist file and reads it.

- **PPRH**: vertical interpolations on pressure levels or height levels for relative humidity. For distributed memory computations are DM-local.
- **PPSTA**: integrates standard atmosphere for geopotential. For distributed memory computations are DM-local.
- **PPT**: vertical interpolations on pressure levels for temperature. For distributed memory computations are DM-local.
- **PPT_OLD**: old version of routine **PPT** used at ECMWF. For distributed memory computations are DM-local.
- **PPTHPW**: computation of the moist irreversible adiabatic potential temperature, vertical interpolation on pressure or height levels. For distributed memory computations are DM-local.
- **PPUV**: vertical interpolations on pressure levels for wind components. For distributed memory computations are DM-local.
- **PPUV_OLD**: old version of routine **PPUV** used at ECMWF. For distributed memory computations are DM-local.
- **PPVVEL**: vertical interpolations on pressure levels or height levels for variables linked to vertical velocity. For distributed memory computations are DM-local.
- **PPZHLEV**: finds full or half levels under specified geopotential heights. Called if post-processing on height levels ([L5]='H'). For distributed memory computations are DM-local.
- **PP2DINT**: vertical linear interpolation (on pressure levels or height levels): used for potential vorticity, potential temperature, equivalent potential temperature, divergence, vorticity, stretching deformation, shearing deformation, stream function, velocity potential, absolute vorticity, wet potential vorticity. For distributed memory computations are DM-local.
- **SPECFITASM**: spectral transformation of model fields (except wind components) and surface orography read on a grid-point field, computation of spectral stream function and velocity potential on model η -levels, writing of all these fields on a file. This routine is called in the second part of a 927-type FULL-POS configuration. Used for shared memory.
- **SPECFITADM**: distributed memory version of **SPECFITASM**.
- **SPECFITGSM**: GRIB version of **SPECFITASM**.
- **SPECFITGDM**: distributed memory version of **SPECFITGSM**.
- **SPOS**: filters some spectral quantities in model spectral space.
- **TJCUBI**: post-processing of jet and tropopause: finds coefficients of a cubic expression. For distributed memory computations are DM-local.
- **TJQUAA**: post-processing of jet and tropopause: performs the least square fit by a quadratic function. For distributed memory computations are DM-local.
- **TJQUD**: post-processing of jet and tropopause: finds coefficients of a quadratic expression. For distributed memory computations are DM-local.
- **VPOS**: interface routine managing all vertical interpolations which can be done before horizontal interpolations. **VPOS** does one call to **POS** for "basic" variables, and special separate calls to **POS** for variables linked with the wind field, such as divergence, (relative) vorticity, absolute vorticity, stream function and velocity potential. For distributed memory computations are DM-local.

Set-up routines of directory "setup":

- **CPFPDS**: initialises the list of fields to be post-processed. For distributed memory computations are DM-global.
- **DFLTVV1**: initialises default values of functions *A* and *B* of hybrid vertical coordinate of the arrival geometry of FULL-POS. For distributed memory computations are DM-global.

- **FPFINDS**: computes full post-processing descriptors of dynamics. For distributed memory computations are DM-global.
- **SUAFN**: initialises ARPEGE field descriptors. Reads namelist **NAMAFN**, initialises **YOMAFN**. For distributed memory computations are DM-global.
- **SUAFN1**: initialises ARPEGE field descriptors: part of calculations done before the namelist reading. For distributed memory computations are DM-global.
- **SUAFN2**: initialises ARPEGE field descriptors: part of calculations done after the namelist reading. For distributed memory computations are DM-global.
- **SUAFPDS**: initialise pointers of the basic fields which have been horizontally then vertically post-processed, initialises **YOMAFPDS**. For distributed memory computations are DM-global.
- **SUAFPOS**: interface routine, initialise relative pointers and descriptors for post-processing help arrays. For distributed memory computations are DM-global.
- **SUALFPOS**: memory allocation for FULL-POS arrays.
- **SUALLO**: memory allocation: general interface routine.
- **SUALMP2**: allocation of arrays used for distributed memory; includes quantities used in the distributed memory version of FULL-POS.
- **SUBFPOS**: interface routine, initialise some commons/modules:
  - **YOMFPF** which describes the spectral filter to be applied.
  - **YOMFPOP** which contains post-processing file-handling parameters.
  - **YOMFPSC2** which contains the control arrays for horizontal post-processing rows.
  - **YOMFPIOS** which contains the parameters for IO scheme on post-processing buffers.
  - **YOMFPSC2B** which contains the control arrays for horizontal post-processing sub-rows.
  - initialise start address of each row within every input/output post-processing buffer.
- **SUCFU**: initialises the control of cumulated fluxes.
- **SUCFUFP**: initialises cumulated fluxes switches for FULL-POS. For distributed memory computations are DM-global.
- **SUCT0**: routine to initialize level 0 control common.
- **SUDIM**: initialises dimensioning quantities, reads namelist **NAMDIM**, initialises **YOMDIM**.
- **SUFPC**: initialises FULL-POS post-processing scientific and technical options, reads namelist **NAMFPC**, initialises **YOMFPC**. For distributed memory computations are DM-global.
- **SUFPCONF**: initialise FULL-POS configuration letters for **STEPO**. Initialise for each scan the configuration letters for vertical post-processing, horizontal post-processing and for **IOPACK**. For distributed memory computations are DM-global.
- **SUFPCSET**: initialises parallel environment for FULL-POS necessary for horizontal interpolations. Works with a lateral band of **NFPWIDE** latitudes in the halo. For distributed memory computations are DM-local. Some quantities are computed in **SUFPRSET** called by **SUFPCSET**.
- **SUFPD**: initialise FULL-POS horizontal subdomains. Computes total number of output points (including the poles if the output grid is Gaussian). Reads namelist **NAMFPD**, initialises **YOMFPD**. For distributed memory computations are DM-global.
- **SUFPDIM**: initialise dimensioning quantities for FULL-POS. Computes number of derived/underived fields, number of fitted/unfitted fields, number of scalar/vector fields as if for a given post-processing time step, all the fields were requested at all levels for all subdomains. For distributed memory computations are DM-global.
- **SUFPDOM**: initialise list of FULL-POS post-processing subdomains. To read the list of subdomains as a character string and convert it into an integer array. In the string, the subdomain

names are separated with a ":". If the string is blank, all the subdomains are required. For distributed memory computations are DM-global.

- **SUFPDS**: set up FULL-POS descriptors for dynamics. For distributed memory computations are DM-global.
- **SUFPDYN**: initialise dynamical part of **YOMFP4** and **PTRFP4**, which contains the requests on post-processing at a given time-step. In this routine, we consider that, for a given level type, all the fields are post-processed at all levels and for all subdomains. Reads namelists containing local variables (**NAMFPDY2**, **NAMFPDYP**, **NAMFPDYH**, **NAMFPDYV**, **NAMFPDYT** and **NAMFPDYS**).
- **SUFPF**: initialises the profile of the spectral post-processing filter. Reads namelist **NAMFPF** and initialises **YOMFPF**.
- **SUFPFIT**: initialise spectral fit option for post-processing. Initialises for each scan the logical switches to decide if spectral fields should be fitted or not. For distributed memory computations are DM-global.
- **SUFPG1**: initialise output geometry: sets default values, then reads namelist **NAMFPG** and checks it. For distributed memory computations are DM-global.
- **SUFPG2**: initialise output geometry: computes DM-global quantities containing the coordinates of the output points in the model geometry, the output mapping factor, and the relative rotation matrix. When an output point is at a geographical pole, the equation of the compass is degenerating, so a direct (and exact) computation is needed. For distributed memory computations are DM-global.
- **SUFPG**: initialise output geometry: calls **SUFPG1** and **SUFPG2**, makes some additional tests then computes the DM-local versions of the DM-global quantities computed in **SUFPG2**. Initialises **YOMFPG**.
- **SUFPIOS**: set-up for using work files on post-processing buffers. To set-up **YOMFPIOS** which contains parameters for using the MIO package on work files. Opens work files.
- **SUFPOPH**: initialise post-processing file-handling parameters (**YOMFPOP**). For distributed memory computations are DM-global.
- **SUFPPHY**: initialise physical part of **YOMFP4** and **PTRFP4**, which contain the requests on post-processing at a given time-step. In this routine, we consider that, for a given level type, all the fields are post-processed at all levels and for all subdomains. Reads namelist containing local variables (**NAMFPPHY**).
- **SUFPRSET**: initialises parallel environment for FULL-POS necessary for horizontal interpolations: communication tables and test. Works with a lateral band of **NFPWIDE** latitudes in the halo. This routine is called only if distributed memory, and computations are DM-local.
- **SUFPSC2**: reads namelist **NAMFPSC2** and sets-up **YOMFPSC2** which contains the control parameters and arrays to scan the post-processing buffers per row.
- **SUFPSC2B**: computes start address of each row within every input/output post-processing buffer. Initialise control arrays on sub-rows (**YOMFPSC2B**). Also initialises **YOMRFPB**, **YOMWFPB**, **YOMDFPB**, **YOMPFPB**, **YOMAFPB**, **YOMFPT0** and **PTRFPB2**.
- **SUFRAME** (or **SUEFRAME** if plane geometry, or **SUFPGRIB** if GRIB files): sets-up the frame of ARPEGE file, in the current case for the FULL-POS files. For distributed memory computations are DM-global.
- **SUGAW**: initialises the Gaussian/Lobatto latitudes and weights for the geometry of FULL-POS grid. Called if the FULL-POS grid is a global Gaussian/Lobatto one. For distributed memory computations are DM-global.
- **SUGENORD**: computes the rotation matrix to find the geographical north, from grid-points of a computational transformed sphere; this matrix is necessary to convert vectors which are initially

given in the repere of the model (transformed) geometry (for example after the horizontal interpolations) towards the geographical repere. For distributed memory computations are DM-global.

- **SUGPPRP**: initialises relative pointers for physics arrays. For distributed memory computations are DM-global.
- **SUGRCFU**: reads the cumulated fluxes on ARPEGE files. For distributed memory computations are DM-global.
- **SUGRXFU**: reads the instantaneous fluxes on ARPEGE files. For distributed memory computations are DM-global.
- **SUGRIDF**: interface for reading the surface grid point fields of the departure files. For distributed memory computations are DM-global.
- **SUGRIDU**: interface for reading the upper air grid point fields of the departure files. For distributed memory computations are DM-global.
- **SUHOW1** (**SUEHOW1** for ALADIN): initialises the coordinates of the nearest model grid point of the FULL-POS interpolation points: the general rule is to take the grid-point which is immediately at the north-west (relative to the transformed geometry) of the interpolation point, except in the case where the north-western point is the pole, in this case one takes the south-western model grid-point. Additional coordinates are computed. For distributed memory computations are DM-local.
- **SUHOW2** (**SUEHOW2** for ALADIN): computes weights without land-sea mask and surface temperature for bilinear or 12 points horizontal interpolations used in FULL-POS. For distributed memory computations are DM-local.
- **SUHOWLSM** (**SUEHOWLSM** for ALADIN): computes weights for bilinear or 12 points horizontal interpolations used in FULL-POS. Possibility to use the land/sea/ice mask or the surface temperature. For distributed memory computations are DM-local.
- **SUINIF**: interface routine for reading the departure files.
- **SUMP**: initialises quantities and arrays controling distributed memory.
- **SUNMEN**: initialises array **NFPMEN** which contains for each Gaussian/Lobatto latitude of the FULL-POS grid the highest zonal wave number represented. Called if the FULL-POS grid is a global Gaussian/Lobatto one.
- **SURFPBUF**: initialise buffer **RFPBUF** of **YOMRFPB**, which contains the output climatology and geometry.
  - If variable **NFPCLI** is 1,2 or 3 the output geometry climatology is read on a file by routine **RDCLIMO** or **RDECCLIMO** according to use of ARPEGE or GRIB files.
  - If variable **NFPCLI** is 0 the output geometry is interpolated from the input geometry by routine **CPCLIMO**, at least for orography.
  - Output geometry is stored in the array **ZFIELD** for orography, mapping factor, local repere (north direction).
  - These data are stored on a MIO file or in the buffer array **RFPBUF** according to the value of **LIOFPR**.
  - Remark: for distributed memory, all computations are DM-local (each processor treats its interpolation points) but diagnostic prints and reading climatology on a file is made on the "main" processor for all the grid-points, so processor communication is necessary after reading climatologies to dispatch data in all the processors and to collect data in the "main" processor to do diagnostics on orography.
- **SURFPDS**: initialises the descriptors of the auxiliary fields needed for horizontal post-processing, initialises **YOMRFPDS**. For distributed memory computations are DM-global.

- **SUSC2**: initialises some control variables required in routine **SCAN2MSM** and **SCAN2MDM**: interpolation buffers quantities, some multitasking quantities but also some distributed memory quantities.
- **SUSIZBUF**: initialises the number of buffers required to store some quantities from the non-lagged grid-point computations (for example computations done in **HPOS** for FULL-POS) towards the lagged grid-point computations (for example computations done in **HPOSLAG** for FULL-POS). Also used for model, semi-Lagrangian interpolations buffers, observation interpolation buffers. Some quantities are in **YOMSC2**. For distributed memory computations are DM-global.
- **SUSPEC**: interface for reading the spectral fields of the departure files.
- **SUVFPOS**: initialises and sorts the list of the dynamical fields to post-process for a given level type of post-processing. Fields are stored as follows: 1) fitted underivated 3D fields; 2) fitted derivated 3D fields; 3) fitted underivated 2D fields; 4) fitted derivated 2D fields; 5) unfitted 3D fields; 6) unfitted 2D fields.
- **SUVFPOSL**: initialises **YOMFP4L** which contains the working variables and arrays to write out the dynamical post-processed fields.
- **SUVPOS**: initialises **YOMVPOS** which contains control variables for vertical post-processing.
- **SUWFPBUF**: initialises buffer **WFPBUF** of **YOMWFPB**, which contains the working fields for horizontal interpolations: size of the box, indexes and weights. Reads or computes input/output land-sea mask, computes weights and index, then fills buffer.
  - Fills interpolation buffers containing the model grid-point land-sea mask and surface temperature (currently surface temperature is not used in weight computations, so the array is filled with -999) in routine **CPCLIMI**.
  - Fills the output geometry land-sea mask and surface temperature (currently surface temperature is not used in weight computations, so the array is filled with -999): such quantities can be read on a file if **NFPCLI**=1,2 or 3 by routines **RDCLIMO** or **RDECCLIMO** or interpolated if **NFPCLI**=0 by routine **CPCLIMO**.
  - Computes weights for horizontal interpolations: a set of weights without land-sea mask and a set to weights with land-sea mask.
  - Stores the weights on an MIO file or in the buffer **WFPBUF** according to the value of **LIOFPW**. Some additional quantities can be stored: model Gaussian/Lobatto latitude index immediately at the north of the interpolation point (lowerly bounded by 1), longitudes indices of the points denoted by $B0$, $B1$, $B2$ and $B3$, number of the FULL-POS subdomain to which belongs each interpolation point.
  - Remark: for distributed memory, all computations are DM-local (each processor treats its interpolation points) but reading climatology on a file is made on the "main" processor for all the grid-points, so processor communication is necessary after reading climatologies to dispatch data in all the processors.
- **SUWFPDS**: initialises the FULL-POS working fields descriptors needed for horizontal post-processing, initialises **YOMWFPDS**. For distributed memory computations are DM-global.
- **SUXFU**: initialises the control of instantaneous fluxes.
- **SUXFUFP**: initialises instantaneous fluxes switches for FULL-POS. For distributed memory computations are DM-global.
- **SU0YOMA**: 0-level interface routine for set-up: first part.
- **SU0YOMB**: 0-level interface routine for set-up: second part.
- **SU1YOM**: 1-level interface routine for set-up.
- **SU3FPOS**: 3-level interface routine for set-up of FULL-POS features. Initialises buffers **RFPBUF** and **WFPBUF**.

- **SU4FPOS**: 4-level interface routine for set-up of FULL-POS features. Initialises blocks **YOM4FPOS**, **YOMFP4** and **PTRFP4**, which contain the requests for post-processing at a given time-step. The grid-point fields are stored in the following order: physics, then cumulated fluxes CFU, then instantaneous fluxes XFU.

Routines of directory "transform".

See documentation about spectral transforms, only specific FULL-POS routines are here described.

- **FTDIRH**: interface routine for direct Fourier transforms.
- **FTINVH**: interface routine for inverse Fourier transforms.
- **LTDIRH**: interface routine for direct Legendre transforms.
- **LTINVH**: interface routine for inverse Legendre transforms.
- **PRFIP**: FULL-POS spectral fit version of **PRFI1**. In inverse Legendre transforms, when transforming FULL-POS fields, **pp_obs**/**PRFIP** is called instead of **transform**/**PRFI1** for fields preparation. **PRFIP** has to be moved later in directory **transform**.
- **REESPESM**: spectral transformation from grid point space to spectral space. For more details see documentation about spectral transforms.
- **SC2FPFSC**: FULL-POS spectral fit version of **SC2FSC**. In inverse Fourier transforms, when transforming FULL-POS fields, **transform**/**SC2FPFSC** is called instead of **transform**/**SC2FSC**.
- **SPEREESM**: spectral transformation from spectral space to grid-point space. For more details see documentation about spectral transforms.
- **TRACARE**: computes geographical coordinates, once given the computational sphere coordinates.
- **TRARECA**: computes the computational sphere coordinates, once given the geographical coordinates.
- **UPDSPFP**: FULL-POS spectral fit version of **UPDSP**. After direct Legendre transforms, when transforming FULL-POS fields, **transform**/**UPDSPFP** is called instead of **transform**/**UPDSP** (final memory transfer in spectral array containing all coefficients).
- **UPDVPOS**: updates **YOMVPOS** which contains control variables for vertical post-processing.
- **UVSPESM**: spectral transformation from grid point space to spectral space: takes the grid-point wind components, transform them into the spectral divergence/vorticity or velocity potential/stream function. For more details see documentation about spectral transforms.

Routines of directory "utility".

- **CHIEN** (or **CCHIEN** if ALADIN): It controls consistency between defined geometry and ARPEGE file. In the case of inconsistency it calls **ABOR1**. For distributed memory computations are DM-global.
- **CPGRIDF**: copies a grid point buffer/work file on another buffer/work file. For distributed memory computations are DM-local.
- **DEALFPOS**, **DEALDDH**, **DEALCOS**, **DEALCTV**, **DEALGES**, **DEALCAN**, **DEALSPA**, **DEALSC2**, **DEALSCR**, **DEALMOD**, **DEALNMI**, **DEALLO**: deallocate space for global variables. Respectively for:
  - Full-POS variables.
  - DDH variables.
  - Cost function variables.
  - Control variables (arrays depending on the size of the control variable).
  - Arrays needed for first guess constraint and for CANARI OI.
  - Analysis grid and diagnostic fields.
  - Spectral arrays.

- Model grid-point buffers and other arrays allocated in **SUSC2**.
- Screening of observations.
- Model fields for screening.
- Other arrays.

Remark: **DEALCAN** is provisionally in directory **canari** and has to be moved later in directory **utility**.

- **EGGDIR**: see **SUGENORD** but used for ALADIN: the departure geometry is the ALADIN one. For distributed memory computations are DM-global.
- **EGGRVS**: geography of grid-points, inversion from grid to geographical sphere ARPEGE-ALADIN. For distributed memory computations are DM-global.
- **EGGX**: geography of grid-points ARPEGE-ALADIN. For distributed memory computations are DM-global.
- **EMPTB**: memory transfer routine; calls **EXPAND21**, **EXPAND31** or **EXPAND41** if unpacking is required. For distributed memory computations are DM-local.
- **ENDIOS**: closes "MIO" files and deletes them.
- **EXTFPF**: extracts 2D fields from grid-point post-processing file. Writes out the post-processed fields of the post-processing grid-point buffer. Performs a transpose from ordering by the **NFPROMAL** based sub-rows of packets used in the grid-point calculations to an ordering by fields. The result is in the form of an array **PREAL** containing the extracted fields sorted out from the first subdomain to the last subdomain. For distributed memory this routine performs DM-local computations on the current processor data only.
- **EXTGPF**: extracts 2D fields from grid-point file. Writes out the model level fields of the model grid-point buffer. Performs a transpose from ordering by the **NPROMA** based packets used in the grid-point calculations to an ordering by fields. The result is in the form of an array **PREAL** containing the extracted fields stored from the North pole to the South pole (including the poles) stored in a compact way taking into account the reduced grid. The result is suitable for being fed directly to the output routines **FAIENC** or grib packing routines **GRIBEX**. This routine can be used only for shared memory.
- **FILLB**: memory transfer routine; calls **PACK21**, **PACK31** or **PACK41** if packing is required. Does the inverse operation of routine **EMPTB**. For distributed memory computations are DM-local.
- **FREEMEM**: interface routine to free memory used by pointer allocations.
- **INCGPF**: includes 2D fields into array/work file. Writes in model fields into the model grid-point buffer. Performs a transpose from ordering by fields to an ordering by the **NPROMA** based packets used in the grid-point calculations. Warning: it is considered that **PREAL** contains the poles values in case ARPEGE/IFS. This routine can be used only for shared memory.
- **IOPACK**: interface for writing data on ARPEGE, ALADIN or GRIB files.
- **MATCLOSE**: routine to close rotation or contraction/dilatation matrixes.
- **MAXGPFV**: computes maximum value of a grid-point field. Extracts a field from model work file or array, then computes its maximum value. If the field is not found, nothing is done. In distributed memory, each processor reads its own data in the buffer **GPPBUF**, then data are collected in the "main" processor, then the maximum of the field is computed in the "main" processor and communicated to all the other processors.
- Function **MGETBUF**: selects and reserves a free buffer *ibuf* in a non-lagged routine to store information about a package of latitude *iproc*. This routine is used only for shared memory.
- Function **MFINDBUF**: finds the buffer *ibuf* in a lagged routine. This routine is used only for shared memory.

- Function **MDONEBUF**: releases the buffer *ibuf* in a lagged routine when information about the package of latitude *iproc* becomes useless. This routine is used only for shared memory.
- **MXMAOP**: product matrix times matrix, also used in spectral computations and spectral transforms.
- **PKGRIDA**: packs model grid point variables by writing to then reading from an ARPEGE file. At the beginning and the end of the routine data are in a buffer or an MIO file. If distributed memory destoring from buffer or MIO file is DM-local, then a communication between processors collects data in the "main" processor, then packing by writing/reading on an ARPEGE file is made in the "main" processor, then packed data are dispatched in all the processors, then each processor stores its own data on a buffer or an MIO file.
- **PKGRIDG**: the same as **PKGRIDA** but for GRIB files.
- **SC2RDG**: memory transfer routine; reads data in a buffer or an IO file and copies it in a grid-point array. For distributed memory computations are DM-local.
- **SC2WRG**: memory transfer routine; reads data in a grid-point array and copies it in a buffer or an MIO file. Does the inverse operation of routine **SC2RDG**. For distributed memory computations are DM-local.
- **WRSPBUF**: writes additional spectral data on MIO files.

2.8 SEQUENCES OF CALLS OF POST-PROCESSING.

2.8.1 Vertical coordinates of post-processing.

Post-processing can be made on pressure levels, height levels, potential vorticity levels, potential temperature levels or η -levels. For post-processing on pressure levels a vertical coordinate linked to pressure (pressure or logarithm of pressure, according to the post-processed variable) is used to compute weights for vertical interpolations. For post-processing on height levels a vertical coordinate linked to geopotential is used to compute weights for vertical interpolations. For post-processing on potential vorticity levels, potential temperature levels or η -levels an intermediate state through a pressure-type coordinate is needed to compute vertical weights.

2.8.2 Sequences of calls: general considerations.

Post-processing for each coordinate is made by different sequences under routine **control**/**STEPO**. Vertical interpolations and horizontal interpolations are made by different sequences under **control**/**STEPO**. A sequence is defined by nine letters (or zeros) [L1][L2][L3][L4][L5][L6][L7][L8][L9] (variable **CLCONF** in routine **control**/**CNT4** and **CDCONF** in routine **control**/**STEPO**).

- L1 controls the IO.
- L2 controls the inverse Legendre transforms.
- L3 controls the inverse Fourier transforms.
- L4 controls the grid-point computations for dynamics and physics.
- L5 controls the grid-point computations for post-processing.
- L6 controls the grid-point computations for assimilation.
- L7 controls the direct Fourier transforms.
- L8 controls the direct Legendre transforms.
- L9 controls the spectral computations.

For example a model integration time-step is defined by the sequence [L1]AAA00AAA. Additional sequences can be performed by calls to **control**/**SCAN2MSM** (SM) or **control**/**SCAN2MDM** (DM) under routines other than

control/**STEPO**. The next subsections will detail sequences for different type of output post-processed data. Note that writing fields post-processed by a sequence [L2][L3][L4][L5][L6][L7][L8][L9] is managed by the [L1] of the following sequence.

Abbreviations:

- DYN3D: 3D dynamical fields.
- DYN2D: 2D dynamical fields.
- DYN2DGP: 2D dynamical fields always written in grid-point.
- MSLP: mean sea level pressure.
- PHIS: surface orography.
- PS: surface pressure.
- PHYSOL: surface physical fields.
- CFU: cumulated fluxes.
- XFU: instantaneous fluxes.

Remark:

CFU and XFU are treated like PHYSOL and not detailed on the following tables.

2.8.3 Global spectral post-processed fields in the same horizontal geometry as the model geometry.

Variable **CFPFMT** is equal to 'MODEL' in **NAMFPC**

.

TABLE 2.1 VERTICAL COORDINATE EXTRAPOLATIONS FOR INTERPOLATION TO 2D FIELDS

Type of interpolated variable	Pressure (with spectaral fit)	Height (without spectral fit)	*PV* (with spectral fit)	θ (with spectral fit)	η (with spectral fit)
DYN3D	AAA0B0PPP E00000000	AAA0H0000 U00000000	AAA0V0PPP Y00000000	AAA0T0PPP M00000000	AAA0S0PPP Z00000000
DYN2D	AAA0H0PPP (PS) UAA0S0PPP (oher fields) Z00000000				
PHYSOL, CFU or XFU [*]					A00000000
DYN3D + DYN2D + PHYSOL	AAA0H0PPP	[*] (PHYSOL) AAA0H0PPP (DYN3D)	[*] (PHYSOL) AAA0H0PPP (DYN3D + PS)	[*] (PHYSOL) AAA0H0PPP (PS)	[*] (PHYSOL) AAA0S0PPP (PS)
(DYN3D+DYN2D)	EAA0H0PPP (PS) U00000000	UAA0S0PPP (other DYN2D) Z00000000	UAA0V0PPP (DYN3D) YAA0S0PPP Z00000000	UAA0T0PPP (DYN3D) MAA0S0PPP Z00000000	

Remarks:

- [*] corresponds to a sequence **GRIDFPOS** $\rightarrow$ (**SCAN2H** $\rightarrow$ (**SCAN2MSM** or **SCAN2MDM**) $\rightarrow$ (**HPOS**+**HPOSLAG**) with [L5]='P') and **WRHFP** (with [L1]='I') by-passing **STEPO**. Equivalent to call **STEPO** first with sequence 0000P0000 then with sequence I00000000. Since there is no change of horizontal geometry here, horizontal interpolations are equivalent to identity. This sequence treats surface physical fields, CFU or XFU fields.

- All sequences called by **STEPO** call **VPOS**, but not **HPOS**, **HPOSLAG** and **ENDVPOS**. **STEPO** is generally called by **DYNFPOS** (under **CNT4**), except the last call to **STEPO** (where only [L1] is non zero) which is called directly by **CNT4**.
- **CPREP4** is never called.
- Headline of organigramme under **CNT4** is generally:
  - **SU4FPOS**
  - **GRIDFPOS** $\rightarrow$
    - **SCAN2H** $\rightarrow$ (**SCAN2MSM** or **SCAN2MDM**) $\rightarrow$ (**HPOS**+**HPOSLAG**) for physics or CFU or XFU if post-processing of such fields is required.
    - **WRHFP** to write post-processed physics or CFU or XFU fields.
  - **DYNFPOS** (always called): **DYNFPOS** calls one or several **STEPO** sequences according to the 3D or 2D dynamical fields to post-process (call to vertical interpolations only). Direct spectral transforms are called for fields to be fitted. Writing on files is made by routine **WRSFP**.
  - **STEPO** (last call for the case where only [L1] is non zero); writing on files is made by routine **WRSFP**.
- For η as the vertical coordinate, the sequence is the same either the η values are a subset of model values or not.
- All 3D variables are directly interpolated by vertical interpolations under **VPOS**, there is no call to **ENDVPOS**.
- Spectral fit is compulsory to post-process velocity potential and stream function.
- For height coordinate, the switch controlling spectral fit is not available in namelist and is set to .FALSE. when only 3D dynamical fields are post-processed. So there is no spectral fit and 3D dynamical fields are provided in grid-point values.
- When spectral fit is activated, post-processed fields are spectral ones for 3D dynamical fields, surface pressure, mean sea level pressure and orography, grid-point ones for the other 2D dynamical fields, the surface physical fields, the CFU and XFU.
- For η coordinate, post-processing of surface pressure is considered as a η-coordinate post-processing on surface $\eta = 1$ ([L5]='S'). For other vertical coordinates, post-processing of surface pressure is considered as a height coordinate post-processing on surface Z=0 ([L5]='H').
- Unavailable variables:
  - Post-processing of velocity potential and stream function is not available for height coordinate.
  - Post-processing of pressure (resp. potential vorticity, potential temperature) is not made even if required for pressure (resp. potential vorticity, potential temperature) coordinate.
  - Post-processing of stretching deformation and shearing deformation is not available if FULL-POS is made in a semi-Lagrangian job.

2.8.4 Global spectral post-processed fields in a horizontal geometry different from the model geometry.

Variable **CFPFMT** is equal to 'GAUSS' and variable **LFPSPEC** is .TRUE. in **NAMFPC** (pseudo-configuration 927). This configuration requires a spectral fit in the arrival geometry.

Routine **CPREP4** is always called. Organigramme has recursivity and is of the following type: **CNT0** (model geometry) $\rightarrow$ set-up, control routines, including **CNT4**. Headline of organigramme under this call to **CNT4** does not depend on the vertical coordinate of post-processing and on the list of post-processed variables, and is generally:

- **SU4FPOS**

- **CPREP4** →
  - **GRIDFPOS** →
    - **SCAN2H** → (**SCAN2MSM** or **SCAN2MDM**) → (**HPOS**+**HPOSLAG**) for physics or CFU or XFU if post-processing of such fields is required.
    - **WRHFP** to write post-processed physics or CFU or XFU fields on unit 91.
  - **DYNFPOS** (always called):
    - first call to **STEPO**: inverse Legendre transform; then (**SCAN2H** → (**SCAN2MSM** or **SCAN2MDM**) →): inverse Fourier transform and horizontal interpolations (**HPOS** + **HPOSLAG**) on model η-levels for grid-point model variables (wind components, temperature, humidity, surface pressure, surface orography).
    - second call to **STEPO**: writing grid-point interpolated fields on model layers on unit 91 (routine **WRHFP**).
  - **CNT0** (output geometry).

CNT0 (output geometry) → set-up, control routines, including:

- a call to **SPECFITASM** under **CNT1**: spectral transformation of model fields (except wind components) and surface orography in unit 91, computation of spectral stream function and velocity potential on model η-levels, writing of all these fields on unit 91.
- a call to **CNT4**.

Headline of organigramme under this call to **CNT4** is generally:

- **SU4FPOS**
- **CPREP4** →
  - **GRIDFPOS** →
    - **SCAN2H** → (**SCAN2MSM** or **SCAN2MDM**) → (**HPOS**+**HPOSLAG**) for physics or CFU or XFU if post-processing of such fields is required (horizontal interpolations are equivalent to identity in this case).
  - **WRHFP** to write post-processed physics or CFU or XFU fields on unit 54.
  - **DYNFPOS** (always called): **DYNFPOS** calls one or several **STEPO** sequences according to the 3D or 2D dynamical fields to post-process (call to vertical interpolations, horizontal interpolations are neutral here). Direct spectral transforms are called for fields to be fitted.
  - **STEPO** (last call for the case where only [L1] is non zero, routine **WRSFP**).
  - **CNT0** (return to model geometry to continue integration if required).

Post-processed fields are written on unit 54.

Under the first call to **CNT0** one finds the following sequences of **STEPO**: AAA0M0000 (vertical interpolations on departure geometry, simple copy on model η-levels), then 0000A0000 (horizontal interpolations on grid-points of the arrival geometry), then Z00000000. It is planned in the future to merge the two sequences AAA0M0000 and 0000A0000 into one sequence AAA0A0000 but that needs some code development: such a modification will allow to save memory.

Under the second call to **CNT0** one finds the same sequences of **STEPO** as in the case without change of geometry (variable **CFPFMT** equal to 'MODEL' in **NAMFPC**, see above).

Remarks, restrictions of use:

APACHE is called if post-processing on height or η-levels.

- **ENDVPOS** is theoretically called if post-processing on height or η-levels to post-process dynamical variables which are not model variables (does not work in cycles 17 and 18). In practical

configurations calling **ENDVPOS** are not completely coded, so **ENDVPOS** is never called for configurations which work.

- It is necessary to post-process at least physical surface fields, in the contrary fields are not written in the first call to **CNT0** but read in the second call to **CNT0** (abort when reading inexistant fields).
- Names of articles cannot be changed in namelist **NAMAFN**.
- For η vertical coordinate, the sequence is the same either the η values are a subset of model values or not.
- Spectral fit is compulsory to post-process velocity potential and stream function.
- For height coordinate, the switch controlling spectral fit is not available in namelist and is set to .FALSE. when only 3D dynamical fields are post-processed. So there is no spectral fit and 3D dynamical fields are provided in grid-point values.
- When spectral fit is activated, post-processed fields are spectral ones for 3D dynamical fields, surface pressure, mean sea level pressure and orography, grid-point ones for the other 2D dynamical fields, the surface physical fields, the CFU and XFU.
- Restrictions of use for η -coordinate:
  - If no change of vertical coordinate, the number of levels to be post-processed has to be the model number of levels.
  - If change of set of vertical levels, the number of levels to be post-processed has to be the number of levels specified in namelist **NAMFPG**.
  - 3D variables which can be post-processed: temperature, stream function (if spectral fit), velocity potential (if spectral fit), horizontal wind components, specific humidity, wind velocity, absolute vorticity, pressure.
  - Variables which cannot be post-processed: geopotential (operand range error abort), relative humidity (operand range error abort), vertical velocity (no abort but post-processing not made), divergence and vorticity (abort in **FTDIRH**), potential temperature (operand range error abort), liquid and solid water (no abort but post-processing not made), moist irreversible adiabatic potential temperature (abort with negative value in a logarithm), cloud fraction (no abort but post-processing not made), equivalent potential temperature (operand range error abort), stretching deformation (no abort but post-processing not made), shearing deformation (no abort but post-processing not made), potential vorticity (no abort but post-processing not made), wet potential vorticity (no abort but post-processing not made).
- Restrictions of use for height coordinate:
  - 3D variables which can be post-processed: temperature, horizontal wind components, specific humidity, wind velocity, absolute vorticity, pressure.
  - Stream function and velocity potential cannot be post-processed due to the fact that there is no spectral fit.
  - Other variables which cannot be post-processed: geopotential (operand range error abort), relative humidity (operand range error abort), vertical velocity (no abort but post-processing not made), divergence and vorticity (abort in **FTDIRH**), potential temperature (operand range error abort), liquid and solid water (no abort but post-processing not made), moist irreversible adiabatic potential temperature (abort with negative value in a logarithm), cloud fraction (no abort but post-processing not made), equivalent potential temperature (operand range error abort), stretching deformation (no abort but post-processing not made), shearing deformation (no abort but post-processing not made), potential vorticity (no abort but post-processing not made), wet potential vorticity (no abort but post-processing not made).
- Other restrictions of use:

- Post-processing of pressure (resp. potential vorticity, potential temperature) is not made even if required for pressure (resp. potential vorticity, potential temperature) coordinate.
- Post-processing of stretching deformation and shearing deformation is not available if FULL-POS is made in a semi-Lagrangian job.

- Residual bugs in cycles 17 and 18:
  - Potential vorticity or potential temperature coordinate: works when no 2D dynamical variable is interpolated but aborts in **GPEPT** called by **POS** when 3D and 2D dynamical variables are post-processed (logarithm of a negative value).
  - Post-processing only 2D dynamical variables without any other variable generates an abort in POS in the second part of the configuration 927 with the diagnostic 'ETA LEVELS SELECTION IMPOSSIBLE'.

A solution to run not working configurations is to perform two calls to FULL-POS, first a configuration 927 which creates an historic file in the final geometry (new η coordinate), then a FULL-POS run without change of horizontal geometry and with global spectral outputs (**CFPFMT**='MODEL').

2.8.5 Global grid-point post-processed fields in the same horizontal geometry or in a different horizontal geometry as the model geometry.

Variable **CFPFMT** is equal to 'GAUSS' and variable **LFPSPEC** is .FALSE. in **NAMFPC**. This configuration does not require any spectral fit in the arrival geometry. Spectral fit can be done in the departure geometry. The case where there is no change of horizontal geometry uses the same computations and is not distinct from the case with change of horizontal geometry: in this particular case horizontal interpolations are made but equivalent to identity.

2.8.6 Spectral fit.

TABLE 2.2 VERTICAL COORDINATE EXRAPOLATIONS FOR INTERPOLATION TO 2D FILEDS

Type of interplated variable	Pressure (with spectral fit)	Height (without spectral fit)	*PV* (with spectral fit)	θ (with spectral fit)	η (with spectral fit)
DYN3D	AAA0B0PPP 0PP0A0000 E00000000	AAA0M0000 0PP0A0000 0000L0000 U00000000	AAA0V0PPP 0PP0A0000 Y00000000	AAA0T0PPP 0PP0A0000 M00000000	AAA0M0000 0000E0000 0000Y0000 Z00000000
DYN2D	AAA0B0PPP 0PP0A0000 (DYN2DGP + MSLP + PHIS) EAA0M0000 0000Z0000 0000L0000 0000L0000 (PS) U00000000				
PHYSOL, CFU or XFU	[*] A00000000				

TABLE 2.2 VERTICAL COORDINATE EXRAPOLATIONS FOR INTERPOLATION TO 2D FILEDS

Type of interplated variable	Pressure (with spectral fit)	Height (without spectral fit)	*PV* (with spectral fit)	θ (with spectral fit)	η (with spectral fit)
DYN3D+ DYN2D	[*] (PHYSOL) AAA0B0PPP 0PP0A0000 (DYN2DGP +MSLP + PHIS+ DYN3D) EAA0M0000 0000Z0000 0000L0000 (PS) U00000000	[*] (PHYSOL) AAA0B0PPP 0PP0A0000 (DYN2DGP +MSLP + PHIS) EAA0M0000 0000Z0000 0000L0000 (PS + DYN3D) U00000000	[*] (PHYSOL) AAA0B0PPP 0PP0A0000 (DYN2DGP +MSLP + PHIS) EAA0M0000 0000Z0000 0000L0000 (PS) UAA0V0PPP	[*] (PHYSOL) AAA0B0PPP 0PP0A0000 (DYN2DGP +MSLP + PHIS) EAA0M0000 0000Z0000 0000L0000 (PS) UAA0T0PPP 0PP0A0000 (DYN3D) Y00000000	[*] (PHYSOL) AAA0B0PPP 0PP0A0000 (DYN2DGP +MSLP + PHIS) EAA0M0000 0000E0000 0000Y0000 (PS + DYN3D) Z00000000 0PP0A0000 (DYN3D) M00000000

Remarks:

- [*] corresponds to a sequence **GRIDFPOS** → (**SCAN2H** → (**SCAN2MSM** or **SCAN2MDM**) → (**HPOS**+**HPOSLAG**) with [L5]='P') and **WRHFP** (with [L1]='I') by-passing **STEPO**. Equivalent to call **STEPO** first with sequence 0000P0000 then with sequence I00000000. This sequence treats surface physical fields, CFU or XFU fields.
- Sequences called by **STEPO** with [L5]='A', 'Z' or 'E' call (**HPOS**+**HPOSLAG**).
- All sequences called by **STEPO** with [L5]='B', 'M', 'V' or 'T' call **VPOS**. **STEPO** is generally called by **DYNFPOS** (under **CNT4**), except the last call to **STEPO** (where only [L1] is non zero) which is called directly by **CNT4**.
- Sequences called by **STEPO** with [L5]='L' or 'Y' call **ENDVPOS**, but exist only if variables other than model variables are post-processed.
- **CPREP4** is never called.
- Headline of organigramme under **CNT4** is generally:
  - **SU4FPOS**
  - **GRIDFPOS** →
  - **SCAN2H** → (**SCAN2MSM** or **SCAN2MDM**) → (**HPOS**+**HPOSLAG**) for physics or CFU or XFU if post-processing of such fields is required.
  - **WRHFP** to write post-processed physics or CFU or XFU fields.
  - **DYNFPOS** (always called): **DYNFPOS** calls one or several **STEPO** sequences according to the 3D or 2D dynamical fields to post-process: call to horizontal interpolations only if [L5]='A', 'Z' or 'E', vertical interpolations before horizontal interpolations only if [L5]='B', 'M', 'V' or 'T', vertical interpolations after horizontal interpolations only if [L5]='Z' or 'E', **ENDVPOS** only if [L5]='L' or 'Y'. Direct spectral transforms are called for fields to be fitted.
  - **STEPO** (last call for the case where only [L1] is non zero, routine **WRHFP**).
- For η vertical coordinate, the sequence is the same either the η values are a subset of model values or not.
- **ENDVPOS** is called only if post-processing of 3D variables other than model variables is wanted, and for height or η coordinates only ([L5]='L' or 'Y').
- **APACHE** is called only if post-processing of model variables and height or η vertical coordinate (case [L5]='M').

- Spectral fit is compulsory to post-process velocity potential and stream function.
- For height coordinate, the switch controlling spectral fit is not available in namelist and is set to .FALSE. when only 3D dynamical fields are post-processed. So there is no spectral fit.
- For η coordinate, there is no spectral fit for 3D dynamical variables, switch **LFITS** is not used.
- When spectral fit is activated, post-processed fields are fitted for 3D dynamical fields (except if η vertical coordinate), mean sea level pressure and orography.
- For η coordinate, post-processing of surface pressure is considered as a η-coordinate post-processing on surface $\eta = 1$ (sequence [L1]AA0M0000 followed by 0000E0000 + 0000Y0000). For other vertical coordinates, post-processing of surface pressure is considered as a height coordinate post-processing on surface Z=0 (sequence [L1]AA0M0000 followed by 0000Z0000 + 0000L0000).
- Restrictions of use for η-coordinate or height coordinate:
  - 3D variables which can be post-processed:
    - Temperature *T*.
    - Horizontal wind components *U* and *V*, wind velocity.
    - Specific humidity (moisture) q.
    - Relative humidity *HU*.
    - Potential temperature θ.
    - Moist (irreversible) pseudo-adiabatic potential temperature θ_w.
    - Equivalent potential temperature θ_e.
    - Passive scalars (maximum of three).
    - Pressure p_{hyd}.
    - Possibility to post-process three additional free upper air fields.
  - 3D variables which cannot be post-processed (abort due to the fact that there is no spectral fit):
    - Velocity potential ψ.
    - Stream function χ.
  - Other 3D variables which cannot be post-processed (post-processing not made even if required):
    - Geopotential μ.
    - Ice content q_i.
    - ω (pressure coordinate vertical velocity).
    - Relative vorticity ζ.
    - Divergence *D*.
    - Liquid water content q_l.
    - Cloudiness q_a.
    - Absolute vorticity $\zeta + f$.
    - Stretching deformation *STD*.
    - Shearing deformation *SHD*.
    - Potential vorticity *PV*.
    - Wet potential vorticity PV_w.
- Other restrictions of use:
  - Post-processing of pressure (resp. potential vorticity, potential temperature) is not made even if required for pressure (resp. potential vorticity, potential temperature) coordinate.
  - Post-processing of stretching deformation and shearing deformation is not available if FULL-POS is made in a semi-Lagrangian job.

2.8.7 No spectral fit.

Sequence [L1][L2][L3][L4][L5]0PPP followed by 0PP0[L5][L6][L7][L8][L9] is replaced by sequence [L1][L2][L3][L4][L5]0000 followed by 0000[L5][L6][L7][L8][L9]. Routine **SPOS** is no longer called.

2.8.8 Grid-point post-processed fields in a ALADIN limited area domain.

Variable **CFPFMT** is equal to 'LELAM' and variable **LFPSPEC** is .FALSE. in **NAMFPC**.
Same sequences and same organigramme as in the case **CFPFMT**='GAUSS' and **LFPSPEC**=.FALSE. .

2.8.9 Grid-point post-processed fields in a latitude-longitude limited area domain.

Variable **CFPFMT** is equal to 'LALON' and variable **LFPSPEC** is .FALSE. in **NAMFPC**.
Same sequences and same organigramme as in the case **CFPFMT**='GAUSS' and **LFPSPEC**=.FALSE. .

2.8.10 Mixed ARPEGE/IFS-ALADIN FULL-POS configurations: spectral post-processed fields in a limited area domain.

Variable **CFPFMT** is equal to 'LELAM' and variable **LFPSPEC** is .TRUE. in **NAMFPC** (pseudo-configuration E927). ALADIN libraries are necessary in this case (take AL07 ALADIN library and CY16T1_AL07 ARPEGE/IFS library). The first call to **CNT0** makes an ARPEGE/IFS run, the second one makes an ALADIN run.

Same sequences and same organigramme as in the case **CFPFMT**='GAUSS' and **LFPSPEC**=.TRUE. with change of horizontal geometry (pseudo-configuration 927).

2.8.11 Pure ALADIN FULL-POS configurations.

ALADIN libraries are necessary in this case (take AL07 ALADIN library and CY16T1_AL07 ARPEGE/IFS library).

- **CFPFMT**='MODEL', **LFPSPEC**=.FALSE.: spectral outputs on an ALADIN domain which is identical to the model ALADIN domain. Same sequences as in the case (**CFPFMT**='MODEL', **LFPSPEC**=.FALSE.) of ARPEGE/IFS. One call to **CNT0**, no call to **CPREP4**.
- **CFPFMT**='LELAM', **LFPSPEC**=.FALSE.: grid-point outputs on an ALADIN domain which is included in the model ALADIN domain. Cannot work if the required post-processing ALADIN domain is not included in the model ALADIN domain. Same sequences as in the case (**CFPFMT**='LELAM', **LFPSPEC**=.FALSE.) of ARPEGE/IFS. One call to **CNT0**, no call to **CPREP4**.
- **CFPFMT**='LALON', **LFPSPEC**=.FALSE.: grid-point outputs on some LALON domains which are included in the model ALADIN domain. Cannot work if the required post-processing LALON domains are not included in the model ALADIN domain. Same sequences as in the case (**CFPFMT**='LALON', **LFPSPEC**=.FALSE.) of ARPEGE/IFS. One call to **CNT0**, no call to **CPREP4**.
- **CFPFMT**='GAUSS', **LFPSPEC**=.FALSE. has no sense for ALADIN.
- **CFPFMT**='LELAM', **LFPSPEC**=.TRUE.: pseudo-configuration EE927, spectral outputs on an ALADIN domain which is included in the model ALADIN domain. Cannot work if the required post-processing ALADIN domain is not included in the model ALADIN domain. Same sequences as in the cases (**CFPFMT**='LELAM', **LFPSPEC**=.TRUE.) of the pseudo-configurations 927 end E927. Two calls to **CNT0** with **LELAM**=.T., **CPREP4** is called.

2.9 SOME SHARED MEMORY FEATURES:

2.9.1 Calculations packets:

Grid-point computations:

In the grid-point space there is a subdivision of the grid-points into **NSLBR**+**NDGSUR** packets of length **NPROMA** (the useful number of values in each packet is lower or equal than **NPROMA**). These packets contain the extra-longitudes. Some packets can contain polar and extra-polar latitudes data. A **NPROMA**-packet always contains a set of complete latitudes, so **NPROMA** is always greater or equal than **NDLON**+3. For the FULL-POS there is a second subdivision into sub-packets of length **NFPROMAL** containing **NFPROF** useful points for computations at the points of the arrival horizontal geometry (routines **SURFPBUF**, **SUWFPBUF**, **HPOSLAG** and **ENDVPOS**). More details will be given later for the data transmission for horizontal interpolations.

One 2D field has **NGPTOT** points. **NGPTOT** does not take account of the extra-longitudes, the poles and the extra-polar latitudes. When counting the additional points due to extra-longitudes, poles and the extra-polar latitudes one obtains a total amount of **NBDYSZ** points. For shared memory the variables **NGPTOTG** and **NGPTOTMX** are equal to **NGPTOT**. All these variables take account of the reduced Gaussian/Lobatto grid. It is assumed and hardcoded that there are one western extra-longitude and two eastern extra-longitudes. Longitude *jlon*=1 is always the "Greenwich" meridian of the computational sphere.

Fourier transforms:

Fourier transforms are done latitude by latitude for each **NPROMA**-packet. Data reorganisation is necessary in the Fourier space between the zonal wave structure necessary for Legendre transforms and the latitudinal structure necessary for Fourier transforms.

Legendre transforms:

Legendre transforms are done zonal wave number by zonal wave number.

Filtering in spectral space:

Filtering in spectral space is done zonal wave number by zonal wave number. A call to **SPOS** currently treats only one zonal wave number.

2.9.2 Transmission of data necessary for FULL-POS horizontal interpolations from HPOS to HPOSLAG: interpolation buffers.

First one associates to each interpolation point, which is generally not a model grid-point, an "associated" model grid-point which currently satisfies to the following rule:

- the "associated" model grid-point is on the longitude immediately west to the interpolation point.
- the "associated" model grid-point is on the latitude immediately south to the interpolation.

Interpolations use data of points which are not necessary on the same latitude as the interpolation point. Thus interpolation routines need to have access to a limited number of latitudes north and south of the "associated" model grid-point latitude. Interpolation points are collected so that a call to **HPOSLAG** for horizontal interpolations treats a set of points (defined by the two arrays **NFPBSR** and **NFPROF**), the "associated" model grid-points set of which is a model package of latitudes (of dimension lower or equal than **NPROMA**), so one can name this set of grid-points the "associated" model grid-point package of latitudes. The number of packages of latitudes north and south of the "associated" model grid-point package of latitudes is precomputed in the subroutine **SUSC2** so as the grid-points necessary for interpolations are always in this set of packages of latitudes. For example, if one needs *N* packages of latitudes north and *N* packages of latitudes south (*N* is stored in variable **NFPWIDE**) of the "associated"

model grid-point package of latitudes (which is denoted by *nlat*), computations of interpolation quantities (made in subroutine **HPOS**) must be performed for the packages of latitudes *nlat-N* to *nlat+N* before making interpolations of points, the "associated" model grid-point of which is localised on the package of latitudes number *nlat*. This is the reason of splitting the grid-point computations into two parts, one non-lagged part (**HPOS**) and one lagged part (**HPOSLAG**). Quantities to be interpolated are computed in the non-lagged part and interpolations are performed in the lagged part. Of course all grid-point data are not kept in memory between the non-lagged part and the lagged part, in order to save memory (and use the input-output scheme). Data for packages of latitudes *nlat-N* to *nlat+N* is stored in interpolation buffer arrays called **PFPBUF1** and **PFPBUF2** in **HPOS**, **HPOSLAG** and **SCAN2MSM** and local arrays **ZFPBUF1** and **ZFPBUF2** in the routine **SCAN2H**. The following considerations made for **PFPBUF1** are valid for **PFPBUF2**.

PFPBUF1 contains the buffers for some variables to be interpolated. The first dimension of **PFPBUF1** is **NPROMA**. (**NPROMA** is above the maximum number of longitudes per latitude + 3 and is defined in the namelist **NAMDYN** and is used to dimension arrays containing only data of the package of latitudes *nlat*). The second dimension of **PFPBUF1** is the number of 2D horizontal fields to be horizontally interpolated. The second element of **PFPBUF1** is used to identify the the FULL-POS quantity to be interpolated (local variable **IFLD** computed in **HPOS**). The third dimension of **PFPBUF1** is **NFPB1** (dummy variable **KFPB1** in **HPOS**), to make available a set of latitudes (*nlat-N* to *nlat+N*) in the lagged part of the grid-point computations. The third element of array **PFPBUF1** for storage of the current latitude *nlat* is denoted by the local integer variable **IB1** in **HPOS** and **HPOSLAG**. **NFPB1** is computed in the set-up routine **SUSIZBUF** called by **SUSC2** in a rather complicated way. Its value is close (but not always equal) to 2N+**NTASKS**+**NDGSUR** (**NTASKS** is the number of tasks, **NDGSUR** is the number of extra-latitudes). Attribution of buffers is made under **HPOS** by calling the function **MGETBUF**: this function searches for a free buffer to store the data currently treated by **HPOS**; if there is no free buffer routine **MGETBUF** generates an **ABOR1** and that means that **NFPB1** has an unsufficient value.

Fig. 2.3 shows how data is arranged in interpolation buffers in the case of asking for post-processing of two variables *X* and *Y* on four layers numbered 1, 2, 3, 4, *N*=2, in the non-lagged and lagged parts of the grid-point computations. **IFLDX** and **IFLDY** are the corresponding second dimension in array **PB1** corresponding to layer number 1. *F* is a mapping: packages of latitudes are not necessary in order in the interpolation buffer arrays in the lagged part of grid-point computations. In the example shown in figure 11.1:

- $F(nlat\text{-}2) = nlat+2$
- $F(nlat\text{-}1) = nlat\text{-}2$
- $F(nlat\ ) = nlat\text{-}1$
- $F(nlat+1) = nlat$
- $F(nlat+2) = nlat+1$

i.e. buffer 1 contains *nlat*+2 data, ..., buffer 5 contains *nlat*+1 data. In interpolation routines one needs to know the number of the buffer where data corresponding to a latitude denoted by **JGL** can be found. This information is computed in **HPOSLAG** by calling function **MFINDBUF**, before interpolations, and stored in the array **IBUFP1**. In the example shown in figure 11.1, that gives:

- if latitude **JGL** is in the package of latitudes *nlat*-2, **IBUFP1**(**JGL**)=2.
- if latitude **JGL** is in the package of latitudes *nlat*-1, **IBUFP1**(**JGL**)=3.
- if latitude **JGL** is in the package of latitudes *nlat* , **IBUFP1**(**JGL**)=4.
- if latitude **JGL** is in the package of latitudes *nlat*+1, **IBUFP1**(**JGL**)=5.
- if latitude **JGL** is in the package of latitudes *nlat*+2, **IBUFP1**(**JGL**)=1.

In this example the local variable **IB1** is equal to 4 in **HPOS** and **HPOSLAG**. When interpolations are done **HPOSLAG** calls a function **MDONEBUF** which searches for the useless buffers: if data for the package of latitudes *nlat*-2 becomes useless for the following interpolations **MDONEBUF** sends a sort of message that **IBUFP1**=2 (the

buffer number 2) will be free to store data of another package of latitudes when calling again **HPOS**.

Set-up routines **CPCLIMI** and **CPCLIMO** use the same logics but in a simpler way: all interpolation buffers are completely filled for all latitudes before the interpolations, so functions **MGETBUF**, **MFINDBUF** and **MDONEBUF** are not used. Similar features are used in **FPMODPREC** to modify the precipitations before interpolations: an interpolation buffer and some routines coded like interpolation routines (**FPSCAW2**, **FPOLIS**) are used.

One uses arrays with horizontal dimension **NPROMA**. One **NPROMA**-length array can contain several latitudes. In interpolation routines one needs to know the position **JROF** in a **NPROMA**-dimensionned array of data concerning the point of coordinates (longitude number= **JLON**, latitude number= **JGL**). The array **NSTABUF** computed in subroutine **SUSC2** gives this information: **JROF**=**JLON**+**NSTABUF**(**JGL**).

Contrary to what occurs in the semi-Lagrangian scheme, the number of interpolation points (variable **NFPRGPL**) is not equal to the number of model grid-points (variable **NGPTOT**). So there is no mapping between the interpolations points and the model grid-points contrary to the semi-Lagrangian scheme where there is a mapping between the interpolation point (origin or medium point of the semi-Lagrangian trajectory) and the model grid-point (corresponding arrival point of the semi-Lagrangian trajectory). So there is need to associate to the **NPROMA** packets a certain amount of FULL-POS interpolation points which are subdivised into **NFPBSR** subpackets of length lower or equal than **NFPROMAL** (the actual length is in **NFPROF**). Computations are made in the set-up routines **SUFPSC2** and **SUFPSC2B**. In **SUFPSC2** one computes:

- for each **NPROMA**-packet number *jslbr*: **NFPLENR**(*jslbr*) is the number of post-processing points associated to the **NPROMA**-packet number *jslbr*. Some statistics about **NFPLENR** are printed (mean, minimum, maximum, standard deviation, number of zero values of **NFPLENR**).
- then one defines **NFPROMAL** which will control the additional subdivision.
- then one computes the number **NFPBSR** of subpackets necessary to store all the **NFPLENR** values in a **NFPROMAL**-dimensioned array.
- **NFPSTAP** is used to count the total number of sub-packets for all the post-processing points. **NFPSTAP**(*jslbr*) is the total number of subpackets for all the **NPROMA**-packets numbered from **NDGSAG** to *jslbr*-1.

In **SUFPSC2B**:

- for each sub-packet *jbsr* from 1 to **NFPBSR**(*jslbr*) one computes the number of FULL-POS interpolated data to be stored (array **NFPROF**); the current rule is: **NFPROF**(**NFPSTAP**(*jslbr*)+1) to **NFPROF**(**NFPSTAP**(*jslbr*)+**NFPBSR**(*jslbr*)-1) are equal to **NFPROMAL**, **NFPROF**(**NFPSTAP**(*jslbr*)+**NFPBSR**(*jslbr*)) is equal to the remaining number of points to store.
- for each post-processing point number *jfp*: if *jfp2* is the position in a dataset of length **NFPROF**, *jslbr* the number of the associated **NPROMA**-packet and *jbsr* the number of the sub-packet between 1 to **NFPBSR**(*jslbr*), the array **NFPSORT** provides the information **NFPSORT**(*jfp2*, **NFPSTAP**(*jslbr*)+*jbsr*) = *jfp*
- some other quantities are computed to store or read interpolated FULL-POS data in buffers or MIO files using the routines **SC2RDG** and **SC2WRG**.

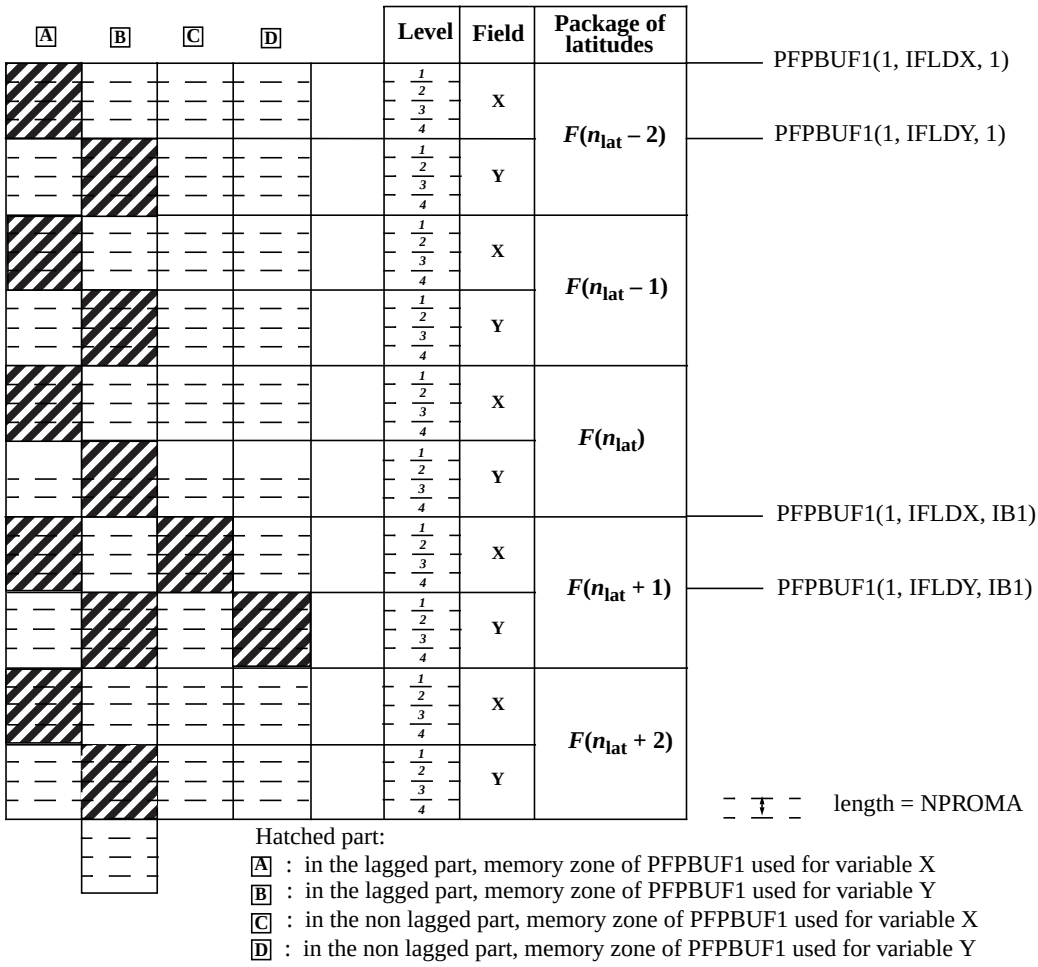


Figure 2.3 Memory management of FULL-POS arrays in non-lagged and lagged parts of horizontal interpolations for shared memory environment.

2.10 SOME DISTRIBUTED MEMORY FEATURES:

2.10.1 Calculations packets:

Grid-point computations:

One processor treats **NGPTOT** points (a part of the Gaussian/Lobatto grid points). The total amount of grid-points for all the processors is **NGPTOTG**. The maximum value of **NGPTOT** is **NGPTOTMX**. In the grid-point space there is a subdivision of the current processor grid-points into **NGPBLKS**=int[(**NGPTOT**+**NPROMA**-1)/**NPROMA**] packets of length **NPROMA** (the useful number of values in each packet is lower or equal than **NPROMA**). These packets do not contain neither extra-longitudes nor polar or extra-polar latitudes data. A **NPROMA**-packet does not always contain a set of complete latitudes. This subdivision into **NPROMA**-packet only concern not lagged computations. Lagged computations (interpolations) are made at one time for all the **NFPRGPL** FULL-POS points treated by the current processor. For the FULL-POS lagged computations there is a second subdivision

into sub-packets of length **NFPROMAL** containing **NFPROF** useful points for computations at the points of the arrival horizontal geometry (routines **SURFPBUF**, **SUWFPBUF**, **HPOSLAG** and **ENDVPOS**). More details will be given later for the data transmission for horizontal interpolations. One 2D field has **NGPTOTG** points divided into **NPROCA***NPROCB sets of **NGPTOT** points treated by each processor. **NGPTOT** does not take account of the extra-longitudes, the poles and the extra-polar latitudes. Variable **NBDYSZ** is equal to **NGPTOT**. All these variables take account of the reduced Gaussian/Lobatto grid. It is assumed and hardcoded that there are one western extra-longitude and two eastern extra-longitudes. The DM-global longitude *jlon*=1 is always the "Greenwich" meridian of the computational sphere. All the vertical levels and the variables corresponding to a same grid-point are treated by the same processor. There are necessary transpositions (reorganisation of data) between grid point computations and Fourier transforms because Fourier transforms need complete latitudes.

Fourier transforms:

Fourier transforms are done latitude by latitude for each **NPROMA**-packet. A processor treats a subset of latitudes, one latitude at the time. A processor can treat only a subset of the vertical levels (if **NPROCB**>1). Data reorganisation and transpositions are necessary in the Fourier space between the zonal wave structure necessary for Legendre transforms and the latitudinal structure necessary for Fourier transforms.

Legendre transforms:

Legendre transforms are done zonal wave number by zonal wave number. A processor treats a subset of zonal wave numbers, one zonal wave number at the time.

Filtering in spectral space:

Filtering in spectral space is done zonal wave number by zonal wave number. A call to **SPOS** currently treats only one zonal wave number. A processor treats a subset of zonal wave numbers, one zonal wave number at the time.

2.10.2 Transmission of data necessary for FULL-POS horizontal interpolations from HPOS to HPOSLAG: interpolation buffers.

First one associates to each interpolation point, which is generally not a model grid-point, an "associated" model grid-point which currently satisfies to the following rule:

- the "associated" model grid-point is on the longitude immediately west to the interpolation point.
- the "associated" model grid-point is on the latitude immediately north to the interpolation point if the interpolation point is not between the Gaussian/Lobatto North pole and the first Gaussian/Lobatto latitude; in the contrary one takes the first Gaussian/Lobatto latitude.

This associated model grid-point is always between the latitudes 1 and **NDGLG** (it is never a pole or an extra-polar latitude point). The processor which treats the FULL-POS interpolation point is the processor which treats this associated model grid-point.

Interpolations use data of points which are not necessary on the same latitude and longitude as the interpolation point. Thus interpolation routines need to have access to a limited number of surrounding latitudes and longitudes which are not necessary treated by the current processor. Interpolation points are collected so that a call to **HPOSLAG** for horizontal interpolations treats a set of points (defined by the two arrays **NFPBSR** and **NFPROF**), the "associated" model grid-points set of which is a **NGPTOT** set which is treated by one processor. The number of surrounding latitudes and longitudes rows necessary for interpolations but which do not belong to the current processor is precomputed in the subroutine **SUSC2** (variable **NFPWIDE**). This is a sort of "halo" belonging to some other processors. Due to the "halo" there is still need to split calculations into not lagged ones (**HPOS**) and lagged ones (**HPOSLAG**). Quantities to be interpolated are computed in the non-lagged part and interpolations are performed in the lagged part. In **HPOS**, only data of the current processor (without any extra-longitudinal data nor

polar and extra-polar data) are computed. For all the **NPROMA**-packages treated by the current processor these data are stored in the arrays **PFPBUF1** and **PFPBUF2** in **HPOS** (also called **PFPBUF1** and **PFPBUF2** in **SCAN2MDM**). Then some communication routines are called in **SCAN2MDM** to constitute the halo. **FPADD3P** does a memory transfer in a bigger interpolation buffer where some place is let for extra-longitudes (assumption is made that interpolations use only the data of 2 longitudes west and 2 longitudes east) and the halo (buffers **ZFPBUF1** and **ZFPBUF2** in **SCAN2MDM**). **FPREAD** and **FPWRIT** do processor communication to constitute the halo (receives and sends data from some other processors). **FPEXTPOL** add data of extra-polar latitudes in the halo when necessary. The first dimension of **ZFPBUF1** and **ZFPBUF2** is **NAFPB1** which is the total number of points one needs for the interpolations (**NAFPB1** is greater than **NGPTOT**). The second dimension of **ZFPBUF1** and **ZFPBUF2** is the same as for **PFPBUF1** and **PFPBUF2**: the number of 2D fields to be interpolated. When all the **NAFPB1** dataset is constituted, the lagged part **HPOSLAG** is called which do horizontal interpolations for all the **NFPRGPL** to be interpolated.

Contrary to the shared memory case, the **NPROMA**-packets subdivision no longer exist in the lagged part of the computations (**HPOSLAG**, **ENDVPOS**) for distributed memory, the only subdivision existing for lagged computations is the **NFPROMAL**-packets one. Functions **MGETBUF**, **MFINDBUF** and **MDONEBUF** are useless. It is planned in the future to reintroduce an intermediate **NPROMA** subdivision for lagged computations, in order to save memory, especially if there are a few numbers of processors.

Set-up routines **CPCLIMI** and **CPCLIMO** use the same logics. Similar features are used in **FPMODPREC** to modify the precipitations before interpolations: an interpolation buffer and some routines coded like interpolation routines (**FPSCAW2**, **FPOLIS**) are used.

Concerning the computations of **SUFPSC2** and **SUFPSC2B**, the principal change compared to shared memory case is that there is no intermediate **NPROMA**-packets subdivision for interpolations, so there is only one packet *jslbr*=1 of **NFPRGPL** FULL-POS interpolations points which is divided into **NFPBSR** subpackets of length lower or equal than **NFPROMAL**. That make simpler the computations in **SUFPSC2** and **SUFPSC2B**. Remaining considerations are the same. For details about the calculations done in **SUFPSC2** and **SUFPSC2B**, see the corresponding description of the shared memory case, replace '*jslbr* varies from **NDGSAG** to **NSLBR**' by '*jslbr*=1'.

2.11 PARAMETER VARIABLES TO BE KNOWN:

2.11.1 Parameter PARFPOS.

Contains basic dimensions for FULL-POS post-processing (in parenthesis, value currently attributed). All variables are DM-global.

- **JPOSDOM** : maximum number of horizontal (sub)domains (15).
- **JPOSFRQ** : maximum number of output frequencies (10).
- **JPOSLEN** : maximum length of a (sub)domain name (7).
- **JPOSLIS** : maximum number of groups of subdomains (10).
- **JPOSDIR** : maximum length of the path (or prefix) for the output files (180).
- **JPOSLE** : maximum number of η -levels on the output subdomain (200, same as **JPMXLE**).
- **JPOSGL** : maximum number of latitude rows of the output Gaussian/Lobatto grid (1001, same as **JPMXGL**).
- **JPOS3DF** : maximum number of specific 3D dynamical fields (45).
- **JPOSSCVA**: maximum number of post-processable passive scalars (3).
- **JPOS2DF** : maximum number of specific 2D dynamical fields (15).
- **JPOSPHY** : maximum number of surface fields (110).

- **JPOSCFU** : maximum number of cumulated fluxes (60).
- **JPOSXFU** : maximum number of instantaneous fluxes (50).
- **JPOS3P** : maximum number of post-processing pressure levels (30).
- **JPOS3H** : maximum number of post-processing height (above orography) levels (65).
- **JPOS3TH** : maximum number of post-processing potential temperature levels (10).
- **JPOS3PV** : maximum number of post-processing potential vorticity levels (10).
- **JPOS3S** : maximum number of post-processing η -levels (200, same as **JPOSLE**).
- **JPOSVSO** : maximum number of climatologic fields of output format (20).

2.12 COMMON/MODULE AND NAMELIST VARIABLES TO BE KNOWN:

2.12.1 PTRFPB2.

Contains variables and pointers for grid point fields used for horizontal post-processing. Variables are initialised in setup routine **setup**/**SUFPSC2B**. No variable in namelist. All variables are DM-global.

List of variables:

- **NFPVT0**: number of auxiliary surface grid point fields.
- **MATS0** : pointer for output surface temperature.
- **MATSI** : pointer for interpolated surface temperature.
- **MAHS0** : pointer for output surface relative moisture.
- **MASDO** : pointer for (standard deviation of orography)**g*.
- **MADOU** : pointer for zonal component of topography principal axis.
- **MADOV** : pointer for meridian component of topography principal axis.
- **MAACT** : pointer for anisotropy coefficient of topography.

PTRFPB2 has to be split later into **YOMFPB2** containing **NFPVT0** and a pointer (directory **pointer**) **PTRFPB2** containing pointers.

2.12.2 YOM4FPOS.

Contains variables relative to FULL-POS working arrays (level 4). Variables are initialised in setup routine **setup**/**SU4FPOS**. No variable in namelist. All variables are DM-global.

Post-processing on 3D dynamical variables:

- **NTFPP** : total number of fields in pressure levels.
- **NTFPH** : total number of fields in height levels.
- **NTFPTH**: total number of fields in θ -levels.
- **NTFPPV**: total number of fields in potential vorticity levels.
- **NTFPS** : total number of fields in η -levels.
- **NOFPP** : total number of pressure levels for each field. Array of dimension **NFP3DF**.
- **NOFPH** : total number of height levels for each field. Array of dimension **NFP3DF**.
- **NOFPTH**: total number of θ -levels for each field. Array of dimension **NFP3DF**.
- **NOFPPV**: total number of potential vorticity levels for each field. Array of dimension **NFP3DF**.
- **NOFPS** : total number of η -levels for each field. Array of dimension **NFP3DF**.
- **NAFPP** : total number of subdomains for each pressure level of each field. Array of dimension (**NFP3P**,**NFP3DF**).

- **NAFPH** : total number of subdomains for each height level of each field. Array of dimension (**NFP3H**,**NFP3DF**).
- **NAFPT** : total number of subdomains for each θ-level of each field. Array of dimension (**NFP3TH**,**NFP3DF**).
- **NAFPV** : total number of subdomains for each potential vorticity level of each field. Array of dimension (**NFP3PV**,**NFP3DF**).
- **NAFPS** : total number of subdomains for each η-level of each field. Array of dimension (**NFP3S**,**NFP3DF**).
- **NCPFP** : field pointers in pressure levels. Array of dimension **NFP3DF**.
- **NCHFP** : field pointers in height levels. Array of dimension **NFP3DF**.
- **NCTFP** : field pointers in θ-levels. Array of dimension **NFP3DF**.
- **NCVFP** : field pointers in potential vorticity levels. Array of dimension **NFP3DF**.
- **NCSFP** : field pointers in η-levels. Array of dimension **NFP3DF**.
- **NPFP** : pressure level pointers for each field. Array of dimension (**NFP3P**,**NFP3DF**).
- **NHFP** : height level pointers for each field. Array of dimension (**NFP3H**,**NFP3DF**).
- **NTFP** : θ-level pointers for each field. Array of dimension (**NFP3TH**,**NFP3DF**).
- **NVFP** : potential vorticity level pointers for each field. Array of dimension (**NFP3PV**,**NFP3DF**).
- **NSFP** : η-level pointers for each field. Array of dimension (**NFP3S**,**NFP3DF**).
- **NIPFP** : subdomain index for each pressure level of each field. Array of dimension (**NFPDOM**,**NFP3P**,**NFP3DF**).
- **NIHFP** : subdomain index for each height level of each field. Array of dimension (**NFPDOM**,**NFP3H**,**NFP3DF**).
- **NITFP** : subdomain index for each θ-level of each field. Array of dimension (**NFPDOM**,**NFP3TH**,**NFP3DF**).
- **NIVFP** : subdomain index for each potential vorticity level of each field. Array of dimension (**NFPDOM**,**NFP3PV**,**NFP3DF**).
- **NISFP** : subdomain index for each η-level of each field. Array of dimension (**NFPDOM**,**NFP3S**,**NFP3DF**).

Post-processing on 2D dynamical variables:

- **NOFP2F**: total number of 2D fields.
- **NOFP2A**: total number of subdomains for each 2D field. Array of dimension **NFP2DF**.
- **NC2FP** : 2D field pointers. Array of dimension **NFP2DF**.
- **NI2FP** : subdomain index for each 2D field. Array of dimension (**NFPDOM**,**NFP2DF**).

Post-processing on lagged physical and instantaneous fluxes variables:

- **NPHYFPB9**: number of physical fields.
- **NXFUFPB9**: number of instantaneous fluxes.
- **NAPHYFP9**: number of subdomains for each physical field. Array of dimension **NFPPHY**.
- **NAXFUFP9**: number of subdomains for each instantaneous flux. Array of dimension **NFPXFU**.
- **NCPHYFP9**: field pointers of physics. Array of dimension **NFPPHY**.
- **NCXFUFP9**: field pointers of instantaneous fluxes. Array of dimension **NFPXFU**.
- **NIPHYFP9**: subdomain index for each physical field. Array of dimension (**NFPDOM**,**NFPPHY**).
- **NIXFUFP9**: subdomain index for each instantaneous flux. Array of dimension (**NFPDOM**,**NFPXFU**).

2.12.3 YOMAFN.

Contains ARPEGE fields descriptors. Variables are initialised in setup routine **setup**/**SUAFN**. All variables are DM-global.

ARPEGE fields descriptors:

- **CNAM3DS**, **C1NAM3**: 3D dynamical fields. Global variable in equivalence with the 43 variables **CNZ** to **CNUA16** described in paragraph "3D dynamical fields".
- **CNAM2DS**, **C1NAM2**: 2D dynamical fields. Global variable in equivalence with the 11 variables **CNSP** to **CNLNSP** described in paragraph "2D dynamical fields".
- **CNAMPDS**, **C1NAMP**: physical surface fields. Global variable in equivalence with the 109 variables **CNLSM** to **CNPSU30** described in paragraph "Physical surface fields". Fields **CNSTL1** to **CNPSU30** are used only in the ECMWF physics.
- **CNAMCDS**, **C1NAMC**: surface cumulated fluxes (CFU). Global variable in equivalence with the 53 variables **CNCLSP** to **CNCTP** described in paragraph "Surface cumulated fluxes".
- **CNAMXDS**, **C1NAMX**: surface instantaneous fluxes (XFU). Global variable in equivalence with the 41 variables **CNXTCC** to **CNXSNS** described in paragraph "Surface instantaneous fluxes".

3D dynamical fields (DYN3D):

TABLE 2.3 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAF**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
01) Geopotential	CNZ	GEOPOTENTIEL	NBZ
02) Temperature	CNT	TEMPERATURE	NBT
03) Wind zonal (x) component	CNU	VENT_ZONAL	NBU
04) Wind meridian (y) component	CNV	VENT_MERIDIE	NBV
05) Specific humidity	CNQ	HUMI_SPECIFI	NBQ
06) Relative humidity	CNR	HUMI_RELATIV	NBR
07) Solid water	CNS	SOLID_WATER	NBS
08) Vertical velocity	CNVV	VITESSE_VERT	NBVV
09) Relative vorticity	CNVOR	VORTICITY	NBVOR
10) Divergence	CNDIV	DIVERGENCE	NBDIV
11) Potential temperature	CNTH	TEMPE_POTENT	NBTH
12) Velocity potential	CNPSI	POT_VITESSE	NBPSI
13) Stream function	CNKHI	FONC_COURANT	NBKHI
14) Liquid water	CNW	LIQUID_WATER	NBW
15) Moist irreversible adiabatic potential temperature	CNTPW	THETA_PRIM_W	NBTPW
16) Cloud fraction (ECMWF)	CNCLF	CLOUD_FRACTI	NBCLF
17) Wind velocity	CNWND	WIND_VELOCIT	NBWND
18) Equivalen potential temperature	CNETH	THETA_EQUIVA	NBETH
19) Absolute vorticity	CNABS	ABS_VORTICIT	NBABS

TABLE 2.3 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAF**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
20) Stretching deformation	CNSTD	STRET_DEFORM	NBSTD
21) Shearing deformation	CNSHD	SHEAR_DEFORM	NBSHD
22) Potential vorticity	CNPV	POT_VORTICIT	NBPV
23) Wet potential vorticity	CNWPV	WET_VORTICIT	NBWPV
24) Passive scalars nr 1	**CNSCVA**(1)	SCALAIRE.001	**NBSCVA**(1)
25) Passive scalars nr 2	**CNSCVA**(2)	SCALAIRE.002	**NBSCVA**(2)
26) Passive scalars nr 3	**CNSCVA**(3)	SCALAIRE.003	**NBSCVA**(3)
27) Pressure	CNP	PRESSURE	NBP
28) Free upper air field nr 1	CNUA1	UPPER_AIR.01	NBUA1
29) Free upper air field nr 2	CNUA2	UPPER_AIR.02	NBUA2
30) Free upper air field nr 3	CNUA3	UPPER_AIR.03	NBUA3

Remark:

There are 13 additional fields "free upper air field nr 4" (DYN3D number 31) to "free upper air field nr 16" (DYN3D number 43) which are used only at ECMWF, the corresponding quantities **CNUA4** to **CNUA16** and **NBUA4** to **NBUA16** are not in namelist **NAMAFN**.

2D dynamical fields (DYN2D):

TABLE 2.4 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAFN**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
01) Surface pressure	CNSP	SURFPRESSION	NBSP
02) Mean sea level pressure	CNMSL	MSLPRESSURE	NBMSL
03) Interpolated model orography	CNFIS	SPECSURFGEOPOTEN	NBFIS
04) Mapping factor	CNGM	MAP_FACTOR	NBGM
05) Folding indicator of the iso-2 PVU surface	CNFOL	TROPO_FOLD_INDIC	NBFOL
06) ICAO jet zonal (x) component of wind	CNSU2	JETVENT_ZONAL	NBSU1
07) ICAO jet meridian (y) component of wind	CNSU2	JETVENT_MERIDIEN	NBSU2
08) ICAO jet pressure	CNSU3	JETPRESSURE	NBSU3
09) ICAO tropopause pressure	CNSU4	ICAOTROP.PRESSUR	NBSU4
10) ICAO tropopause temperature	CNSU5	ICAOTROP.TEMPERA	NBSU5

Remarks:

- there is one additional field "logarithm of surface pressure" (DYN2D number 11) which is used only at ECMWF, the corresponding quantities **CNLNSP** and **NBLNSP** are not in namelist **NAMAFN**.
- fields DYN2D number 06 to 10 have a different definition at ECMWF (optional surface fields), what is written in the previous table is valid only for METEO-FRANCE.

Physical surface fields (PHYSOL):

TABLE 2.5 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAFN**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
01) Land/sea mask	CNLSM	SURFIND.TERREMER	NBLSM
02) Output grid-point orography $\times g$	CNGFIS	SURFGEOPOTENTIEL	NBGFIS
03) Surface temperature	CNST	SURFTEMPERATURE	NBST
04) Deep soil temperature	CNDST	PROFTEMPERATURE	NBDST
05) Interpolated surface temperature	CNRDST	INTSURFTEMPERATU	NBRDST
06) Surface soil wetness	CNSSW	SURFRESERV.EAU	NBSSW
07) Deep soil wetness	CNDSW	PROFRESERV.EAU	NBDSW
08) Relaxation deep soil wetness	CNRDSW	RELAPROP.RMAX.EA	NBRDSW
09) Clim. relative surface soil wetness	CNCSSW	SURFPROP.RMAX.EA	NBCSSW
10) Clim. relative deep soil wetness	CNCDSW	PROFPROP.RMAX.EA	NBCDSW
11) Snow depth	CNSD	SURFRESERV.NEIGE	NBSD
12) Surface roughness $\times g$	CNSR	SURFZ0.FOIS.G	NBSR
13) Roughness length of bare surface $\times g$	CNBSR	SURFZ0REL.FOIS.G	NBBSR
14) Albedo	CNAL	SURFALBEDO	NBAL
15) Emissivity	CNEMIS	SURFEMISSIVITE	NBEMIS
16) Standard deviation of orography) $\times g$	CNSDOG	SURFET.GEOPOTENT	NBSDOG
17) Percentage of vegetation	CNVEG	SURFPROP.VEGETAT	NBVEG
18) Percentage of land	CNLAN	SURFPROP.TERRE	NBLAN
19) Anisotropy coefficient of topography	CNACOT	SURFVAR.GEOP.ANI	NBACOT
20) Direction of main axis of topography	CNDPAT	SURFVAR.GEOP.DIR	NBDPAT
21) Soil first level temperature	CNSTL1	LEV1TEMPERATURE	NBSTL1
22) Soil first level wetness	CNSWL1	LEV1RESERV.EAU	NBSWL1
23) Soil second level temperature	CNSTL2	LEV2TEMPERATURE	NBSTL2
24) Soil second level wetness	CNSWL2	LEV2RESERV.EAU	NBSWL2
25) Soil third level temperature	CNSTL3	LEV3TEMPERATURE	NBSTL3
26) Soil third level wetness	CNSWL3	LEV3RESERV.EAU	NBSWL3
27) Soil fourth level temperature	CNSTL4	LEV4TEMPERATURE	NBSTL4

TABLE 2.5 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAFN**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
28) Soil fourth level wetness	CNSWL4	LEV4RESERV.EAU	NBSWL4
29) Temperature of snow layer	CNTSN	TEMP.SNOWLAYER	NBTSN
30) Anisotropy of surface orography	CNISOR	ANISO.SUB.SOROG	NBISOR
31) Angle of surface orography	CNANOR	ANGLE.SUB.SOROG	NBANOR
32) Slope of surface orography	CNSLOR	SLOPE.SUB.SOROG	NBSLOR
33) Logarithm of surface roughness	CNLSRH	LOG.SURF.ROUGH	NBLSRH
34) Skin temperature	CNSKT	SKINTEMPERATURE	NBSKT
35) Apparent surface humidity	CNASQ	AP.SURF.HUMIDITY	NBASQ
36) Skin wetness	CNSRC	SKINRESERV.EAU	NBSRC

Remarks:

- fields PHYSOL numbers 21 to 36 are used only at ECMWF, the corresponding quantities **CNSTL1** to **CNSRC** and **NBSTL1** to **NBSRC** are not in namelist **NAMAFN**.
- there are additional fields PHYSOL numbers 37 to 79 (not detailed here) which are used only at ECMWF, the corresponding quantities **CN...** and **NB...** are not in namelist **NAMAFN**. These fields are diagnostic ones and some of them are present in the list of CFU or XFU fields available at METEO-FRANCE.
- there are additional optional fields PHYSOL numbers 80 to 109 (not detailed here) which are used only at ECMWF, the corresponding quantities **CNPSU1** to **CNPSU30** and **NBPSU1** to **NBPSU30** are not in namelist **NAMAFN**.

Surface cumulated fluxes:

TABLE 2.6 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAFN**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
01) Large scale precipitation	CNCLSP	SURFPREC.EAU.GEC	NBCLSP
02) Convective precipitation	CNCCP	SURFPREC.EAU.CON	NBCLSP
03) Large scale snow fal	CNCLSS	SURFPREC.NEI.GEC	NBCLSS
04) Convective snow fall	CNCCSF	SURFPREC.NEI.CON	NBCCSF
05) *U*-stress	CNCUSS	SURFTENS.TURB.ZO	NBCUSS
06) *V*-stress	CNCVSS	SURFTENS.TURB.ME	NBCVSS
07) Surface sensible heat flux	CNCSSH	SURFFLU.CHA.SENS	NBCSSH
08) Surface latent heat flux	CNCSLH	SURFCHAL LATENTE	NBCSLH
09) Tendency of surface pressure	CNCTSP	SURFPRESSION SOL	NBCTSP
10) Total cloud cover	CNCTCC	ATMONEBUL.TOTALE	NBCTCC

TABLE 2.6 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAFN**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
11) Boundary layer dissipation	CNCBLD	SURFDISSIP SURF	NBCBLD
12) Surface solar radiation	CNCSSR	SURFFLU.RAY.SOLA	NBCSSR
13) Surface thermal radiation	CNCSTR	SURFFLU.RAY.THER	NBCSTR
14) Top solar radiation	CNCTSR	SOMMFLU.RAY.SOLA	NBCTSR
15) Top thermal radiation	CNCTTR	SOMMFLU.RAY.THER	NBCTTR
16) Convective cloud cover	CNCCCC	ATMONEBUL.CONVEC	NBCCCC
17) High cloud cover	CNCHCC	ATMONEBUL.HAUTE	NBCHCC
18) Medium cloud cover	CNCMCC	ATMONEBUL.MOYENN	NBCMCC
19) Low cloud cover	CNCLCC	ATMONEBUL.BASSE	NBCLCC
20) *U*-gravity-wave stress	CNCUGW	SURFTENS.DMOG.ZO	NBCUGW
21) *V*-gravity-wave stress	CNCVGW	SURFTENS.DMOG.ME	NBCVGW
22) Water evaporation	CNCE	SURFFLU.MEVAP.EA	NBCE
23) Snow sublimation	CNCS	SURFFLU.MSUBL.NE	NBCS
24) Latent heat evaporation	CNCLHE	SURFFLU.LAT.MEVA	NBCLHE
25) Latent heat sublimation	CNCLHS	SURFFLU.LAT.MSUB	NBCLHS
26) Cloudiness	CNCC	SURFCUMUL NEBUL	NBCC
27) Soil moisture	CNCWS	SURFCONTENU EAU	NBCWS
28) Snow mass	CNCSNS	SURFRESERV NEIGE	NBCSNS
29) Total precipitable water	CNCQTO	ATMOHUMI TOTALE	NBCQTO
30) Total ozone	CNCTO3	ATMOOZONE TOTALE	NBCTO3
31) Top mesospheric enthalpy	CNCTME	TOPMESO ENTH	NBCTME
32) Solid specific moisture	CNCICE	ATMOHUMI SOLIDE	NBCICE
33) Liquid specific moisture	CNCLI	ATMOHUMI LIQUIDE	NBCLI
34) Contribution of convection to U	CNCCVU	SURFFL.U CONVEC	NBCCVU
35) Contribution of convection to V	CNCCVV	SURFFL.V CONVEC	NBCCVV
36) Contribution of convection to q	CNCCVQ	SURFFL.Q CONVEC	NBCCVQ
37) Contribution of convection to $c_p T$	CNCCVS	SURFFL.CT CONVEC	NBCCVS
38) Contribution of turbulence to q	CNCTUQ	SURFFL.Q TURBUL	NBCTUQ
39) Contribution of turbulence to $c_p T$	CNCTUS	SURFFL.CT TURBUL	NBCTUS
40) Clear sky shortwave radiative flux	CNCSOC	SURFRAYT SOL CL	NBCSOC
41) Clear sky longwave radiative flux	CNCTHC	SURFRAYT THER CL	NBCTHC
42) Surface parallel solar flux	CNCSOP	SURFRAYT DIR SUR	NBCSOP
43) Top parallel solar flux	CNCTOP	TOPRAYT DIR SOM	NBCTOP
44) Surface down solar flux	CNCSOD	SURFRAYT DIFF DE	NBCSOD
45) Surface down thermic flux	CNCTHD	SURFRAYT THER DE	NBCTHD
46) Melt snow	CNCFON	SURFFONTE NEIGE	NBCFON

TABLE 2.6 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAFN**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
47) Heat flux in soil	CNCCHS	SURFCHAL. DS SOL	NBCCHS
48) Water flux in soil	CNCEAS	SURFEAU DANS SOL	NBCEAS
49) Surface soil runoff	CNCSRU	SURFRUISSELLEMEN	NBCSRU
50) Deep soil runoff	CNCDRU	PROFRUISSELLEMEN	NBCDRU
51) Interception soil layer runoff	CNCIRU	SURFRUISS. INTER	NBCIRU
52) Evapotranspiration flux	CNCETP	SURFEVAPOTRANSPI	NBCETP
53) Transpiration flux	CNCTP	SURFTRANSPIRATIO	NBCTP

Surface instantaneous fluxes:

TABLE 2.7 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAFN**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
01) Total cloud cover	CNXTCC	SURFNEBUL.TOTALE	NBXTCC
02) *U*-component of wind at 10 metres (pbl)	CNX10U	CLSVENT.ZONAL	NBX10U
03) *V*-component of wind at 10 metres (pbl)	CNX10V	CLSVENT.MERIDIEN	NBX10V
04) Temperature at 2 metres (pbl)	CNX2T	CLSTEMPERATURE	NBX2T
05) Specific humidity at 2 metres (pbl)	CNX2SH	CLSHUMI.SPECIFIQ	NBX2SH
06) Relative humidity at 2 metres (pbl)	CNX2RH	CLSHUMI.RELATIVE	NBX2RH
07) Convective cloud cover	CNXCCC	SURFNEBUL.CONVEC	NBXCCC
08) High cloud cover	CNXHCC	SURFNEBUL.HAUTE	NBXHCC
09) Medium cloud cover	CNXMCC	SURFNEBUL.MOYENN	NBXLCC
11) Maximum temperature at 2 metres	CNXX2T	CLSMINI.TEMPERAT	NBXX2T
12) Minimum temperature at 2 metres	CNXN2T	CLSMAXI.TEMPERAT	NBXN2T
13) Cloudiness	CNXC	SURFCUMUL NEBUL	NBXC
14) Contribution of convection to *U*	CNXCVU	S000FL.U CONVEC	NBXCVU
15) Contribution of convection to *V*	CNXCVV	S000FL.V CONVEC	NBXCVV
16) Contribution of convection to *q*	CNXCVQ	S000FL.Q CONVEC	NBXCVQ
17) Contribution of convection to $c_p T$	CNXCVS	S000FL.CT CONVEC	NBXCVS
18) Contribution of turbulence to *U*	CNXTUU	S000FL.U TURBUL	NBXTUU
19) Contribution of turbulence to *V*	CNXTUV	S000FL.V TURBUL	NBXTUV
20) Contribution of turbulence to *q*	CNXTUQ	S000FL.Q TURBUL	NBXTUQ
21) Contribution of turbulence to $c_p T$	CNXTUS	S000FL.CT TURBUL	NBXTUS
22) Contribution of gravity wave drag to *U*	CNXGDU	S000FL.U ONDG.OR	NBXGDU
23) Contribution of gravity wave drag to *V*	CNXGDV	S000FL.V ONDG.OR	NBXGDV
24) Large scale precipitation	CNXLSP	S000PLUIE STRATI	NBXLSP

TABLE 2.7 POST-PROCESSABLE FIELDS

Field	Field name in **NAMAFN**	Default name in **NAMAFN**	Number of bits in **NAMAFN**
25) Convective precipitation	CNXCP	S000PLUIE CONVEC	NBXCP
26) Large scale snow fall	CNXLSS	S000NEIGE STRATI	NBXLSS
27) Convective snow fall	CNXCSF	S000NEIGE CONVEC	NBXCSF
28) Surface solar radiation	CNXSSR	S000RAYT.SOLAIRE	NBXSSR
29) Surface thermal radiation	CNXSTR	S000RAYT.TERREST	NBXSTR
30) Top solar radiation	CNXTSR	SOMMRAYT.SOLAIRE	NBXTSR
31) Top thermal radiation	CNXTTR	SOMMRAYT.TERREST	NBXTTR
32) *U* at bottom level	CNXUBL	SURFU NIVBAS	NBXUBL
33) *V* at bottom level	CNXVBL	SURFV NIVBAS	NBXVBL
34) Temperature at bottom level	CNXTBL	SURFT NIVBAS	NBXTBL
35) Specific humidity at bottom level	CNXQBL	SURFQ NIVBAS	NBXQBL
36) Geopotential at bottom level	CNXGBL	SURFGEOP NIVBAS	NBXGBL
37) Surface temperature	CNXST	SURFTEMPE SURF	NBXST
38) Deep soil temperature	CNXDT	SURFTEMPE PROF	NBXDT
39) Surface water content	CNXSW	SURFRESER SURF	NBXSW
40) Deep soil water content	CNXDW	SURFRESER PROF	NBXDW
41) Snow mass	CNXSNS	SURFNEIGE	NBXSNS

Number of bits for coding in FULL-POS:

- **NBIT3DS**: 3D dynamical fields. Global variable in equivalence with the 43 variables **NBZ** to **NBUA16** described in paragraph "3D dynamical fields".
- **NBIT2DS**: 2D dynamical fields. Global variable in equivalence with the 11 variables **NBSP** to **NBLNSP** described in paragraph "2D dynamical fields".
- **NBITPDS**: physical surface fields. Global variable in equivalence with the 20 variables **NBLSM** to **NBDPAT** described in paragraph "Physical surface fields". Fields **NBSTL1** to **NBSRC** are used only in the ECMWF physics and are not yet implemented in the cycles 17 and 18 of ARPEGE/IFS.
- **NBITCDS**: surface cumulated fluxes (CFU). Global variable in equivalence with the 53 variables **NBCLSP** to **NBCTP** described in paragraph "Surface cumulated fluxes".
- **NBITXDS**: surface instantaneous fluxes (XFU). Global variable in equivalence with the 41 variables **NBXTCC** to **NBXSNS** described in paragraph "Surface instantaneous fluxes".
- Default value is 24 for all these quantities.

Scalar/vector descriptor (0 for scalar, 1 for vector):

- **NVECPDS**: physical surface fields. Global variable in equivalence with the 20 variables **NVLSM** to **NVDPAT** (same suffix as variables, the name of which starts by **NB...**). Fields **NVSTL1** to **NVSRC** are used only in the ECMWF physics and are not yet implemented in the cycles 17 and 18 of ARPEGE/IFS.
- **NVECCDS**: surface cumulated fluxes (CFU). Global variable in equivalence with the 53 variables **NVCLSP** to **NVCTP** (same suffix as variables, the name of which starts by **NB...**).
- **NVECXDS**: surface instantaneous fluxes (XFU). Global variable in equivalence with the 41 variables **NVXTCC** to **NVXSNS** (same suffix as variables, the name of which starts by **NB...**).

- Default value is 1 for **NVCUSS**, **NVCVSS**, **NVCUGW**, **NVCVGW**, **NVX10U**, **NVX10V**, 0 for the other quantities.

Use of land-sea mask for horizontal interpolations (0=no; 1=yes):

- **NITMPDS**: physical surface fields. Global variable in equivalence with the 20 variables **NILSM** to **NIDPAT** (same suffix as variables, the name of which starts by **NB...**). Fields **NISTL1** to **NISRC** are used only in the ECMWF physics and are not yet implemented in the cycles 17 and 18 of ARPEGE/IFS.
- **NITMCDS**: surface cumulated fluxes (CFU). Global variable in equivalence with the 53 variables **NICLSP** to **NICTP** (same suffix as variables, the name of which starts by **NB...**).
- **NITMXDS**: surface instantaneous fluxes (XFU). Global variable in equivalence with the 41 variables **NIXTCC** to **NIXSNS** (same suffix as variables, the name of which starts by **NB...**).
- Default value is 0 for **NILSM**, **NIGFIS**, **NISDOG**, 1 for the other physical surface fields quantities, 1 for **NICUSS**, **NICVSS**, **NICUGW**, **NICVGW**, **NIX10U**, **NIX10V**, **NIX2T**, **NIX2SH**, **NIX2RH**, **NIXX2T**, **NIXN2T**, 0 for the other CFU or XFU quantities.

Namelist NAMAFN:

The following variables of **YOMAFN** are in namelist **NAMAFN**: **CNZ**, **CNT**, **CNU**, **CNV**, **CNQ**, **CNR**, **CNS**, **CNVV**, **CNVOR**, **CNDIV**, **CNTH**, **CNPSI**, **CNKHI**, **CNW**, **CNTPW**, **CNCLF**, **CNWND**, **CNETH**, **CNABS**, **CNSTD**, **CNSHD**, **CNPV**, **CNWPV**, **CNSCVA**, **CNP**, **CNUA1**, **CNUA2**, **CNUA3**, **CNSP**, **CNMSL**, **CNFIS**, **CNGM**, **CNFOL**, **CNSU1**, **CNSU2**, **CNSU3**, **CNSU4**, **CNSU5**, **CNLSM**, **CNGFIS**, **CNST**, **CNDST**, **CNRDST**, **CNSSW**, **CNDSW**, **CNRDSW**, **CNCSSW**, **CNCDSW**, **CNSD**, **CNSR**, **CNBSR**, **CNAL**, **CNEMIS**, **CNSDOG**, **CNVEG**, **CNLAN**, **CNACOT**, **CNDPAT**, **CNCLSP**, **CNCCP**, **CNCLSS**, **CNCCSF**, **CNCUSS**, **CNCVSS**, **CNCSSH**, **CNCSLH**, **CNCTSP**, **CNCTCC**, **CNCBLD**, **CNCSSR**, **CNCSTR**, **CNCTSR**, **CNCTTR**, **CNCCCC**, **CNCHCC**, **CNCMCC**, **CNCLCC**, **CNCUGW**, **CNCVGW**, **CNCE**, **CNCS**, **CNCLHE**, **CNCLHS**, **CNCC**, **CNCWS**, **CNCSNS**, **CNCQTO**, **CNCTO3**, **CNCTME**, **CNCICE**, **CNCLI**, **CNCCVU**, **CNCCVV**, **CNCCVQ**, **CNCCVS**, **CNCTUQ**, **CNCTUS**, **CNCSOC**, **CNCTHC**, **CNCSOP**, **CNCTOP**, **CNCSOD**, **CNCTHD**, **CNCFON**, **CNCCHS**, **CNCEAS**, **CNCSRU**, **CNCDRU**, **CNCIRU**, **CNCETP**, **CNCTP**, **CNXTCC**, **CNX10U**, **CNX10V**, **CNX2T**, **CNX2SH**, **CNX2RH**, **CNXCCC**, **CNXHCC**, **CNXMCC**, **CNXLCC**, **CNXX2T**, **CNXN2T**, **CNXC**, **CNXCVU**, **CNXCVV**, **CNXCVQ**, **CNXCVS**, **CNXTUU**, **CNXTUV**, **CNXTUQ**, **CNXTUS**, **CNXGDU**, **CNXGDV**, **CNXLSP**, **CNXCP**, **CNXLSS**, **CNXCSF**, **CNXSSR**, **CNXSTR**, **CNXTSR**, **CNXTTR**, **CNXUBL**, **CNXVBL**, **CNXTBL**, **CNXQBL**, **CNXGBL**, **CNXST**, **CNXDT**, **CNXSW**, **CNXDW**, **CNXSNS**, **NBZ**, **NBT**, **NBU**, **NBV**, **NBQ**, **NBR**, **NBS**, **NBVV**, **NBVOR**, **NBDIV**, **NBTH**, **NBPSI**, **NBKHI**, **NBW**, **NBTPW**, **NBCLF**, **NBWND**, **NBETH**, **NBABS**, **NBSTD**, **NBSHD**, **NBPV**, **NBWPV**, **NBSCVA**, **NBP**, **NBUA1**, **NBUA2**, **NBUA3**, **NBSP**, **NBMSL**, **NBFIS**, **NBGM**, **NBFOL**, **NBSU1**, **NBSU2**, **NBSU3**, **NBSU4**, **NBSU5**, **NBLSM**, **NBGFIS**, **NBST**, **NBDST**, **NBRDST**, **NBSSW**, **NBDSW**, **NBRDSW**, **NBCSSW**, **NBCDSW**, **NBSD**, **NBSR**, **NBBSR**, **NBAL**, **NBEMIS**, **NBSDOG**, **NBVEG**, **NBLAN**, **NBACOT**, **NBDPAT**, **NBCLSP**, **NBCCP**, **NBCLSS**, **NBCCSF**, **NBCUSS**, **NBCVSS**, **NBCSSH**, **NBCSLH**, **NBCTSP**, **NBCTCC**, **NBCBLD**, **NBCSSR**, **NBCSTR**, **NBCTSR**, **NBCTTR**, **NBCCCC**, **NBCHCC**, **NBCMCC**, **NBCLCC**, **NBCUGW**, **NBCVGW**, **NBCE**, **NBCS**, **NBCLHE**, **NBCLHS**, **NBCC**, **NBCWS**, **NBCSNS**, **NBCQTO**, **NBCTO3**, **NBCTME**, **NBCICE**, **NBCLI**, **NBCCVU**, **NBCCVV**, **NBCCVQ**, **NBCCVS**, **NBCTUQ**, **NBCTUS**, **NBCSOC**, **NBCTHC**, **NBCSOP**, **NBCTOP**, **NBCSOD**, **NBCTHD**, **NBCFON**, **NBCCHS**, **NBCEAS**, **NBCSRU**, **NBCDRU**, **NBCIRU**, **NBCETP**, **NBCTP**, **NBXTCC**, **NBX10U**, **NBX10V**, **NBX2T**, **NBX2SH**, **NBX2RH**, **NBXCCC**, **NBXHCC**, **NBXMCC**, **NBXLCC**, **NBXX2T**, **NBXN2T**, **NBXC**, **NBXCVU**, **NBXCVV**, **NBXCVQ**, **NBXCVS**, **NBXTUU**, **NBXTUV**, **NBXTUQ**, **NBXTUS**, **NBXGDU**, **NBXGDV**, **NBXLSP**, **NBXCP**, **NBXLSS**, **NBXCSF**, **NBXSSR**, **NBXSTR**, **NBXTSR**, **NBXTTR**, **NBXUBL**, **NBXVBL**, **NBXTBL**, **NBXQBL**, **NBXGBL**, **NBXST**, **NBXDT**, **NBXSW**, **NBXDW**, **NBXSNS**, **NILSM**, **NIGFIS**, **NIST**, **NIDST**, **NIRDST**, **NISSW**, **NIDSW**, **NIRDSW**, **NICSSW**,

NICDSW, **NISD**, **NISR**, **NIBSR**, **NIAL**, **NIEMIS**, **NISDOG**, **NIVEG**, **NILAN**, **NIACOT**, **NIDPAT**.

2.12.4 YOMAFPB.

Contains buffer for basic fields resulting from APACHE in FULL-POS. Variables are initialised in setup routine **setup**/**SUFPSC2B**. No variable in namelist. All variables are DM-local.

List of variables:

- **NLENAFPBL**: length of buffer.
- **AFPBUF** : buffer.
- **NSTAAFPB** : start adresses for post-processed packets of points within buffer.

2.12.5 YOMAFPDS.

Absolute pointers of the basic fields which have been horizontally then vertically post-processed. Variables are initialised in setup routine **setup**/**SUAFPDS**. No variable in namelist. All variables are DM-global.

Field pointers:

- **MFPUPT2**: for upper air pressure.
- **MFPAPT2**: for cloud fraction.
- **MFPUT2** : for zonal component of wind.
- **MFPVT2** : for meridian component of wind.
- **MFPTT2** : for temperature.
- **MFPQT2** : for specific humidity.
- **MFPWT2** : for liquid water.
- **MFPST2** : for ice.
- **MFPSVT2**: for passive scalar variables.
- **MFPSPT2**: for surface pressure.

Other variables:

- **NAFPOS** : number of fields.
- **NFPALEV**: maximum number of fields.

2.12.6 YOMCT0.

Contains control variables. The following variables also present in namelist **NAMCT0** can be useful for FULL-POS (these variables are DM-global).

- **NCONF**: configuration (default value is 1).
- **LFPOS**: FULL-POS main switch (default value is .FALSE.).
- **NFRPOS**: post-processing frequency (default value depends on configuration).
- **NPOSTS**: post-processing control array (default value depends on configuration).
- **LIOGAUX**: I/O on unfitted vertically post-processed fields (default value depends on configuration).
- **LIOFOU1**: I/O on Fourier data (default value depends on configuration).
- **LIOLPOL**: I/O on Legendre polynomial (default value depends on configuration).
- **LIOSPEC**: I/O on saved spectral data (default value depends on configuration).
- **LIOSUA**: I/O on saved upper air grid-point data (default value depends on configuration).
- **LIOSSU**: I/O on saved surface data (default value depends on configuration).
- **LIOSCF**: I/O on saved cumulated fluxes data (default value depends on configuration).
- **LIOSXF**: I/O on saved instantaneous fluxes data (default value depends on configuration).

- **LMLTSK**: switch for multitasking (default value depends on configuration).
- **NTASKS**: number of processors (default value depends on configuration).
- **CNPPATH**: path name for selection files (default value is ' ').
- **CFPNCF**: file name for FULL-POS control file (default value is 'ncf927').

The following variables also present in namelist **NAMPAR0** can be useful for FULL-POS when running distributed memory jobs (these variables are DM-global).

- **NPRGPNS**: number of processors used in A-direction during grid-point phase in North-South direction.
- **NPRGPEW**: number of processors used in B-direction during grid-point phase in East-West direction.
- **NPRTRW**: number of processors used in A-direction during transform phase in wave space. Default value is the same as **NPRGPNS**.
- **NPRTRNS**: number of processors used in A-direction during transform phase in North-South direction (used to dimension some arrays used in Fourier transforms). Default value is the same as **NPRGPNS**.
- **NPRTRV**: number of processors used in B-direction during transform phase in vertical direction. Default value is the same as **NPRTREW**.
- **NPROCK**: not currently used.
- **NPROC**: total number of processors requested (DM-global). **NPROC** has to be equal to the products **NPROCA*NPROCB**, **NPRGPNS*NPRGPEW**, **NPRTRW*NPRTRV**. Default value is 1 for distributed memory, 0 for shared memory.
- **NOUTPUT**: type of output (DM-global).
  - 0 = no diagnostic output;
  - 1 = only diagnostic output from processor number 1.
  - 2 = diagnostic output from all processors into separate files.

Default value is 1.

- **LMESSP**: .TRUE./.FALSE.: distributed memory/ shared memory run (DM-global). Default value is .FALSE. .
- **LMPDIAG**: extensive message passing diagnostic output requested if .TRUE. (DM-global). Default value is .FALSE. .

The following variables also present in namelist **NAMPAR1** can be useful for FULL-POS when running distributed memory jobs (these variables are DM-global). They are read and initialised in **setup**/**SUMP0**.

- **NINSTR1**: number of time steps instrumented (DM-global). Default value is 0.
- **NINSTR2**: number of time steps instrumented (DM-global). Default value is 0.

2.12.7 YOMDFPB.

Contains buffer for fully post-processed dynamical fields. Variables are initialised in setup routine **setup**/**SUFPSC2B**. No variable in namelist. All variables are DM-local.

List of variables:

- **NLENDFPBL**: length of buffer.
- **GDFPBUF** : buffer.
- **NSTADFPB** : start adresses for post-processed packets of points within buffer.

2.12.8 YOMDIM.

Contains dimensioning variables. The following variables are related with FULL-POS.

Variables computed in SUDIM:

- **NDLON**: maximum number of Gaussian/Lobatto longitudes (DM-global).
- **NFPXLEV**: maximum number of post-processing levels (DM-global).
- **NPMAX**: post-processing truncation (DM-global, default value is generally **NSMAX**).
- **NFPPHYB**: maximum number of post-processed physical fields (DM-global).
- **NPROMA**: working dimension of (vertical) post-processing rows (DM-global, default value depends on configuration).
- **NGPBLKS**: number of **NPROMA**-subpackets in a processor (DM-local).
- **NDGSAFPH**: modified lower bound for latitude (DM-local).
- **NDGENFPH**: modified upper bound for latitude (DM-local).

Variables computed under SUFPDIM:

- **NFPGT1**: maximum number of fields to be vertically post-processed (DM-global).
- **NFPAUXB**: maximum number of vertically post-processed fields to remain unfitted (DM-global).
- **NFPSPA**: maximum number of underived vertically post-processed fields to be fitted (DM-global).
- **NFPSPD**: maximum number of derived vertically post-processed fields to be fitted, given one horizontal subdomain (DM-global).
- **NFPSPB**: maximum number of derived vertically post-processed fields to be fitted, given the maximum number of horizontal subdomains (DM-global).
- **NFPGT0B**: maximum number of fields to be horizontally post-processed (DM-global).
- **NFPDYNB**: maximum number of post-processed dynamical fields (DM-global).
- **NFPAVEC**: maximum number of vectorial underived vertically post-processed fields to be fitted (DM-global).
- **NFPBVEC**: maximum number of vectorial derived vertically post-processed fields to be fitted, given the maximum number of horizontal subdomains (DM-global).
- **NFPASCA**: maximum number of scalar underived vertically post-processed fields to be fitted (DM-global).
- **NFPBSCA**: maximum number of scalar derived vertically post-processed fields to be fitted, given the maximum number of horizontal subdomains (DM-global).
- **NFPLEI**: maximum number of useful fields in the working arrays for the inverse Legendre transforms (DM-global).
- **NFPLED**: maximum number of useful fields in the working arrays for the direct Legendre transforms on wind (DM-global).

Namelist NAMDIM:

The following variables of **YOMDIM** are in namelist **NAMDIM**: **NPROMA**, **NPMAX**.

2.12.9 YOMFP4.

Contains dimensions for FULL-POS on one-level-type dynamical fields. Variables are initialised in setup routine **setup**/**SU4FPOS**. No variable in namelist. All variables are DM-global.

Number of fields in each array/buffer for dynamics:

- **NGT1FP** : number of fields in **GT1** (fields to be fitted).
- **NAUXFPB**: number of fields in **GAUXBUF** (fields to remain unfitted).
- **NAUX3FP**: number of 3D fields in **GAUXBUF** (fields to remain unfitted).

- **NSPAFP** : number of useful fields in **SPAFP**.
- **NSPBFP** : number of useful fields in **SPBFP**.
- **NSPDFP** : number of useful fields in **SPDFP**.
- **NGT0FPB**: number of fields in **GT0BUF** during an horizontal post-processing time step.
- **NGT03FP**: number of 3D fields in **GT0BUF** during an horizontal post-processing time step.
- **NDYNFPB**: number of fields in **GDFPBUF**.

Dimensions for dynamical fields post-processing:

- **NOFP3F** : total number of 3D fields.
- **NOFP3A** : total number of subdomains for each level of each field. Array of dimension (**NFPXLEV**,**NFP3DF**).
- **NOFPLEV**: total number of levels for each field. Array of dimension **NFP3DF**.
- **NAVECFP**: number of vectorial underived vertically post-processed fields to be fitted.
- **NBVECFP**: number of vectorial derived vertically post-processed fields to be fitted.
- **NASCAFP**: number of scalar underived vertically post-processed fields to be fitted.
- **NBSCAFP**: number of scalar derived vertically post-processed fields to be fitted.
- **NLEIFP** : number of fields in the working arrays for the inverse Legendre transforms.
- **NLEDFP** : number of fields in the working arrays for the direct Legendre transforms.

Dimensions for physical fields and fluxes post-processing:

- **NPHYFPB**: number of physical fields.
- **NCFUFPB**: number of cumulated fluxes.
- **NXFUFPB**: number of instantaneous fluxes.

2.12.10 YOMFP4L.

Contains information relative to lagged variables needed to write out post-processed fields. Variables are initialised in the sequence **pp_obs**/**DYNFPOS** → **setup**/**SUVFPOSL**. No variable in namelist. All variables are DM-global.

Horizontally post-processed dynamical variables:

- **NDYNFPL** : number of fields in **GDFPBUF**.
- **NFLDFPL** : field pointer. Array of dimension **NFPDYNB**.
- **RLEVFPL** : level value. Array of dimension **NFPDYNB**.
- **NBITFPL** : number of bits when writing output file. Array of dimension **NFPDYNB**.
- **NDOMFPL** : number of subdomains for each field. Array of dimension **NFPDYNB**.
- **NINDFPL** : indices of subdomains for each field. Array of dimension (**NFPDOM**,**NFPDYNB**).
- **NAUXFPL** : number of fields in **GAUXBUF** (fields to remain unfitted).
- **NAUX3FPL**: number of 3D fields in **GAUXBUF** (fields to remain unfitted).
- **NGT0FPL** : number of fields in **GT0BUF** during an horizontal post-processing time step.
- **NGT03FPL**: number of 3D fields in **GT0BUF** during an horizontal post-processing time step.

Spectral variables:

- **NSPAFPL** : number of fields in **SPAFP**.
- **NSPDFPL** : number of fields in **SPDFP**.
- **NFLDAFPL**: field pointers in **SPAFP**. Array of dimension **NFPSPA**.
- **NFLDDFPL**: field pointers in **SPDFP**. Array of dimension **NFPSPD**.
- **RLEVAFPL**: level values in **SPAFP**. Array of dimension **NFPSPA**.
- **RLEVDFPL**: level values in **SPDFP**. Array of dimension **NFPSPD**.

Grid-point variables:

- **NFLDFPXL**: field pointers in **GAUXBUF**. Array of dimension **NFPAUXB**.
- **RLEVFPXL**: level values in **GAUXBUF**. Array of dimension **NFPAUXB**.
- **NBITFPXL**: number of bit when writing out **GAUXBUF**. Array of dimension **NFPAUXB**.

2.12.11 YOMFPC.

Contains scientific and technical variables for post-processing. Variables are initialised in setup routine **setup**/**SUFPC**. All variables are DM-global.

Technical variables.

- **CFPDIR**: prefix (path) for the output files (default value is 'PF').
- **CFPIDEN**: identificator of the output files.
- **CFPFMT**: format of the output files, can take the following values:
  - 'MODEL' for output in spherical harmonics.
  - 'GAUSS' for output in grid-point space on Gaussian grid (covering the global sphere).
  - 'LELAM' for output on a grid of kind 'ALADIN' (spectral or grid-point coefficients).
  - 'LALON' for a grid of kind "latitudes * longitudes'.

Default is 'GAUSS' in ARPEGE/IFS, 'LELAM' in ALADIN.

- **CFPDOM**, **C1FPDOM**: names of the subdomains. Names have at maximum 7 characters. If **CFPFMT** = 'GAUSS' or 'LELAM' only one output domain is allowed. If **CFPFMT** = 'LALON' the maximum of output subdomains allowed is 10. By default, one output domain is requested, **CFPDOM**(1)='000' and **CFPDOM**(*i*)=" for *i*>1.
- **CFP3DF**, **C1FP3DF**: names of the 3D dynamics fields to be post-processed. Names have at maximum 12 characters. By default **CFP3DF** contains blanks (no 3D dynamical field to post-process).
- **CFP2DF**, **C1FP2DF**: names of the 2D dynamics fields to be post-processed. Names have at maximum 16 characters. By default **CFP2DF** contains blanks (no 2D dynamical field to post-process).
- **CFPPHY**, **C1FPPHY**: names of the physical fields to be post-processed. Names have at maximum 16 characters. By default **CFPPHY** contains blanks (no physical field to post-process).
- **CFPCFU**, **C1FPCFU**: names of the cumulated fluxes fields to be post-processed. Names have at maximum 16 characters. By default **CFPCFU** contains blanks (no cumulated fluxes field to post-process).
- **CFPXFU**, **C1FPXFU**: names of the instantaneous fluxes fields to be post-processed. Names have at maximum 16 characters. By default **CFPXFU** contains blanks (no instantaneous fluxes field to post-process).
- remark: variables **CFP(XXX)** and **C1FP(XXX)** are in equivalence (same quantity) but differently dimensioned.
- **MFP3DF**: gribcodes of the 3D dynamics fields to be post-processed (variable used at ECMWF only).
- **MFP2DF**: gribcodes of the 2D dynamics fields to be post-processed (variable used at ECMWF only).
- **MFPPHY**: gribcodes of the physical fields to be post-processed (variable used at ECMWF only).
- **MFPCFU**: gribcodes of the cumulated fluxes fields to be post-processed (variable used at ECMWF only).

- **MFPXFU**: gribcodes of the instantaneous fluxes fields to be post-processed (variable used at ECMWF only).
- **RFP3P**: list of post-processing pressure levels in Pa (no default value).
- **RFP3H**: list of post-processing height (above orography) levels in meters (no default value).
- **RFP3TH**: list of post-processing potential temperature levels in Kelvins (no default value).
- **RFP3PV**: list of post-processing potential vorticity levels in S.I. units (no default value).
- **NRFP3S**: list of post-processing η -levels (pseudo-configuration 927 only). No default value.
- **LFPCNT**: control varying output variables according to time step.
- **LFPNORM**: print out the mean of each post-processed field for each subdomain (default value is .TRUE.).
- **LTRACEFP**: trace for FULL-POS (additional prints on listing, default value is .FALSE.).
- **NFPGRIB**: level of GRIB coding in output file ARPEGE/ALADIN for grid-point arrays (default value is 2).
- **NFPGRSP**: level of GRIB coding in output file ARPEGE/ALADIN for spectral arrays (default value is 2).
- **LFPBACK**: .TRUE. to return to initial geometry (case **LFPSPEC**) (default value is .FALSE.).
- **NFPDOM**: useful dimension of **CFPDOM**.
- **NFP3DF**: useful dimension of **CFP3DF** (default value is the number of 3D wanted dynamical fields).
- **NFP2DF**: useful dimension of **CFP2DF** (default value is the number of 2D wanted dynamical fields).
- **NFPPHY**: useful dimension of **CFPPHY** (default value is the number of wanted surface physical fields).
- **NFPCFU**: useful dimension of **CFPCFU**.
- **NFPXFU**: useful dimension of **CFPXFU**.
- **NFP3P**: useful dimension of **RFP3P**.
- **NFP3H**: useful dimension of **RFP3H**.
- **NFP3TH**: useful dimension of **RFP3TH**.
- **NFP3PV**: useful dimension of **RFP3PV**.
- **NFP3S**: useful dimension of **NRFP3S**.

Scientific variables.

- **LFPSPEC**: .T. if post-processed dynamical fields are written out as spectral coefficients, .F. if post-processed dynamical fields are written out as grid point values (default value is .FALSE., except if second part of a pseudo-configuration 927 or E927).
- **LFIT2D**: 1 if 2D dynamical post-processed fields should be fitted, 0 otherwise. Default value is .T. .
- **LFITP**: .T. if post-processed fields on pressure levels should be fitted, .F. otherwise. Default value is .T. .
- **LFITH**: .T. if post-processed fields on height levels should be fitted, .F. otherwise.
- **LFITT**: .T. if post-processed fields on θ -levels should be fitted, .F. otherwise. Default value is .F. .
- **LFITV**: .T. if post-processed fields on *PV*-levels should be fitted, .F. otherwise. Default value is .F. .
- **LFITS**: .T. if post-processed fields on η -levels should be fitted, .F. otherwise.
- **LFPUVDZ**: .T. if divergence and vorticity should be post-processed with respect to the wind components (default value is .TRUE.).
- **LFPUVKP**: .T. if stream function and potential velocity should be post-processed with respect to the wind components, otherwise they are post-processed with respect to the divergence and the vorticity (default value is .TRUE.).

- **NFPCLI**: usage level for climatology:
  - 0: no climatology (default value).
  - 1: orography and land-sea mask of output only. In your script, the local name of this external climatology file should be "const.clim.DDDDDDD", where DDDDDDD is the variable **CFPDOM** of the namelist **NAMFPC**. This file should be of the same geometry as the post-processing file you want to make.
  - 2: all available climatological fields of the current month. Use of auxiliary land-sea mask, orography, surface temperature, relative surface wetness, deep soil temperature, relative deep soil wetness, albedo, emissivity, standard deviation of orography, percentage of vegetation, roughness length, anisotropy coefficient of topography, direction of the main axis of topography and snow depth. For that you need to make 2 climatology files: the first one, named "Const.Clim", should be in the input model geometry, the second one, named "const.clim.DDDDDDD", where DDDDDDD is the variable **CFPDOM** of the namelist **NAMFPC**, should be in the output (post-processing) geometry.
  - 3: shifting mean from the climatological fields of the current month to the ones of the closest month.
- **NDOMFP**: domain definition of fields in files; **NDOMFP** = 1: physical fields are grid-points stored on (C+I)+E and dynamical fields are spectral (option available in ALADIN only); **NDOMFP** = -1: physical and dynamical fields are grid-points stored on (C+I)+E; **NDOMFP** = 0: physical and dynamical fields are grid-points stored on (C+I). C, I, E mean respectively conservation domain, intermediate zone and extension zone. Default value depends on configuration:
  - if **CFPFMT**='MODEL', **LFPSPEC**=.F., default value is -1.
  - if **CFPFMT**='GAUSS','LELAM' or 'LALON' and **LFPSPEC**=.F., default value is 0.
  - if **CFPFMT**='GAUSS' or 'LELAM' and **LFPSPEC**=.T., default value is -1 for the first part, 1 for the second part.
- **LFPQ**: .T. if specific humidity is interpolated (then deduct relative humidity), .F. if relative humidity is interpolated (then deduct specific humidity). Default value is .FALSE. . It is better to use the default value (.FALSE.): in this case, relative humidity is conserved within displacement of the planetary boundary layer and it is interpolated vertically. Relative humidity is considered to have better properties for an interpolation than mixing ratio even if it is not a conservative quantity.
- **LASQ**: .T. if apparent surface humidity is set to 80% saturation (default value is .FALSE.).
- **WSXI**: maximum surface moisture in input.
- **WDXI**: maximum deep soil moisture in input.
- **WSXO**: maximum surface moisture in output. Default value is -999. .
- **WDXO**: maximum deep soil moisture in output. Default value is -999. .
- **WCXO**: maximum "climatological" moisture in output. Default value is -999. .
- **LSNOWI**: snow in the input file. Default value is equal to **LNEIGE**.
- **LSNOWO**: snow in the output file. Default value is equal to **LNEIGE**.
- **FPBL**: critical thickness of PBL. Default value is 17500 Pa.
- **RFPCORR**: critical orography difference for correcting surface temperature through standard profile, in J/kg. Default value is 300**g*.
- **RFPMXZ**: non-critical maximum difference between interpolated orography and output orography, in J/kg. Default value is 3000**g*.
- **LFPSPLIT**: .TRUE. if split of FULL-POS post-processing of dynamical fields.
- **NFPSPLIT**: number of dynamical fields to be post-processed at a time.
- **MFP3DYN** : maximum number of 3D-dynamical fields needed for **LFPART2**.
- **MFP2DYN** : maximum number of 2D-dynamical fields needed for **LFPART2**.

- **NFPTRSPLIT**: pointer to actual post-processing package of dynamical fields.

Remarks.

- Note that if you ask for post-processing of dynamical fields which are not spectral in the model, then these post-processed fields will not be spectrally fitted, even if the corresponding switch **LFIT...** is .TRUE. .
- Note that if you wish to post-process upper air dynamical fields on height levels or hybrid levels, it is not possible to apply such spectral fit because the horizontal interpolations are performed before the vertical interpolation in order to respect the displacement of the planetary boundary layer.

Namelist NAMFPC:

The following variables of **YOMFPC** are in namelist **NAMFPC**: **CFPDIR**, **CFPFMT**, **CFPDOM**, **CFP3DF**, **CFP2DF**, **CFPPHY**, **CFPCFU**, **CFPXFU**, **LFPSPEC**, **LFPBACK**, **RFP3P**, **RFP3H**, **RFP3TH**, **RFP3PV**, **NRFP3S**, **NFPGRIB**, **NFPGRSP**, **NDOMFP**, **LFIT2D**, **LFITP**, **LFITT**, **LFITV**, **LFITS**, **RFPCORR**, **LFPUVDZ**, **LFPUVKP**, **NFPCLI**, **LFPQ**, **WSXO**, **WDXO**, **WCXO**, **LSNOWI**, **LSNOWO**, **LFPNORM**, **LTRACEFP**, **FPBL**, **RFPMXZ**, **LASQ**, **MFP3DF**, **MFP2DF**, **MFPPHY**, **NFP3DF**, **NFP2DF**, **NFPPHY**, **LFPSPLIT**, **NFPSPLIT**.

2.12.12 YOMFPD.

Variables concerning the boundaries and the horizontal dimensions of each output subdomain. Variables are initialised in setup routine **setup**/**SUFPD**.

For all kinds of output (sub)domains:

- **NLAT**: number of latitudes for each output (sub)domain (poles not included if a Gaussian grid is required); array of integers; corresponds to the variable **NDGLG** of the output grid(s) (DM-global).
- **NLON**: number of longitudes for each output (sub)domain; array of integers; corresponds to the variable **NDLON** of the output grid(s) (DM-global).
- **RLATN**: northern latitude in degrees for each output (sub)domain; array of reals (DM-global).
- **RLATS**: southern latitude in degrees for each output (sub)domain; array of reals (DM-global).
- **RLONW**: western longitude in degrees for each output (sub)domain; array of reals (DM-global).
- **RLONE**: eastern longitude in degrees for each output (sub)domain; array of reals (DM-global).
- **NFPSIZEG**: number of points in each subdomain (DM-global).
- **NFPSIZEL**: DM-local version of variable **NFPSIZEG**, its value is computed in **SUFPG**.
- **NFPGP**: maximum number of output points (DM-global). This is the sum on all the subdomains of the **NFPSIZEG**. Reduction of the grid near the poles for global output grids is not taken in account, so **NFPSIZEG** can be above the actual total number of points where interpolations will have to be done.
- **NFPD1**: first dimension of the array to be written on output file (DM-global). **NFPD1** can be above **NFPGP** for global domains (**NFPD1** counts polar latitudes contrary to **NFPGP**) and ALADIN domains (**NFPD1** can count extension zone points according to the value of **NDOMFP** in **NAMFPC** but **NFPGP** does not count any point of the extension zone).

These variables have many possible default values, according to the content of the namelist **NAMFPC**.

ARPEGE default values (LELAM=.FALSE.) for NLAT, NLON, RLATN, RLATS, RLONW, RLONE:

- if **CFPFMT**='GAUSS' or **CFPFMT**='MODEL', the default values are: **NLAT**=**NDGL**, **NLON**=**NDLON**, while the other variables are not useful.
- if **CFPFMT**='LALON', there are many possible default values, according to the value of **CFPDOM** in **NAMFPC**:

TABLE 2.8

CFPDOM	NLAT	NLON	RLATN	RLATS	RLONW	RLONE
HENORD	60	180	89.25	0.75	0.	358.
HESUDC	60	180	-0.75	-89.25	0.	358.
HESUDA	30	90	-1.5	-88.5	0.	356.
ATLMED	65	129	72.75	24.75	-84.	44.
EURATL	103	103	71.25	20.25	-32.	36.
ZONCOT	81	81	63.75	33.75	-20.	20.
FRANCE	61	61	53.25	38.25	-8.	12.
GLOB15	121	240	90.	-90.	0.	358.5
EURAT5	105	149	72.	20.	-32.	42.
ATOUR10	81	166	80.	0.	-100.	65.
EUROC25	105	129	61.	35.	-15.	17.
GLOB25	73	144	90.	-90.	0.	357.5
EURSUD	41	54	48.25	28.25	-34./3.	24.
EUREST	39	73	60.25	41.25	-56/3.	88./3
GRID25	21	41	75.	25.	-50.	50.
MAROC	158	171	42.90	19.20	-19.80	5.85

If **CFPDOM**(*j*) is uninitialised or unknown from FULL-POS, then the default values are all null, and you must at least specify in the namelist the values of **NLAT** and **NLON**, otherwise the job will abort.

- If **CFPFMT**='LELAM', there are many possible default values, according to the value of **CFPDOM** in **NAMFPC** (see **SUFPD**, for real quantities there are often more digits than written in the following table):

TABLE 2.9

CFPDOM	NLAT	NLON	RLATN	RLATS	RLONW	RLONE
BELG	72	72	53.6257625	47.0055223	359.727133	10.8165438
SLOV	48	48	50.31862495	41.48230800	8.11809117	19.87921810
MARO	160	160	43.03274783	18.41928424	-19.8912708	9.83167877
OPMA	108	108	43.84932233	17.32436063	-20.7906551	11.39205009
LACE	192	216	56.04274264	33.07606553	1.16026046	41.19323585
ROUM	72	72	52.86547894	35.28895004	14.16917554	39.71861793
FRAN	180	180	55.02332687	36.20003128	-9.18826294	19.26073265

ALADIN default values (LELAM=.FALSE.) for NLAT, NLON, RLATN, RLATS, RLONW, RLONE:

They are not detailed here and computation is sometimes complicated, see routine **SUFPD** for more details.

Additional variables required for a grid ALADIN:

If you ask for a grid ALADIN (**CFPFMT**='LELAM'), there are additionnal variables that you can specify:

- **NFPLUN**: actual first row of longitude (DM-global).
- **NFPLUX**: actual last row of longitude (DM-global).
- **NFPGUN**: actual first row of latitude (DM-global).
- **NFPGUX**: actual last row of latitude (DM-global).
- **NFPZONL**: half the size of area I in abscissa *x* (DM-global).
- **NFPZONG**: half the size of area I in ordinate *y* (DM-global).

There are many possible default values, according to the value of **CFPDOM** in **NAMFPC**; for **NFPLUX** and **NFPGUX** there are two possible default values according to the value of **NFPDOM**, the first one is valid when **NFPDOM**=0, the second one is valid when **NFPDOM**=1 or -1:

TABLE 2.10

CFPDOM	NFPLUN	NFPLUX	NFPGUN	NFPGUX	NFPZONL	NFPZONG
BELG	1	72; 61	1	72; 61	8	8
SLOV	1	48; 37	1	48; 37	8	8
MARO	1	160; 149	1	160; 149	8	8
OPMA	1	108; 97	1	108; 97	8	8
LACE	1	216; 205	1	192; 181	8	8
ROUM	1	72; 61	1	72; 61	8	8
FRAN	1	180; 169	1	180; 169	8	8

For **CFPFMT**='MODEL','GAUSS' or 'LALON' these variables are not useful, they have default values equal to 0 excepted in the case ALADIN when **CFPFMT**='MODEL' or 'LALON': defaults are **NFPLUN**=**NDLUN**, **NFPLUX**=**NDLUX**, **NFPGUN**=**NDGUN**, **NFPGUX**=**NDGUX**, **NFPZONL**=**NBZONL**, **NFPZONG**=**NBZONG**.

Namelist NAMFPD:

The following variables of **YOMFPD** are in namelist **NAMFPD**: **NLAT**, **NLON**, **RLATN**, **RLATS**, **RLONW**, **RLONE**, **NFPLUN**, **NFPLUX**, **NFPGUN**, **NFPGUX**, **NFPZONL**, **NFPZONG**.

2.12.13 YOMFPF.

Variables defining the FULL-POS filter. Variables are initialised in setup routine **setup**/**SUFPF**. All variables are DM-global.

List of variables:

- **NFMAX** : maximum truncation of the output subdomains (default value depends on configuration).
- **LFPFIL**: .TRUE. if "THX" filter for derived fields is active (default value is .TRUE. in ARPEGE excepted in the first part of a 927-type configuration, .FALSE. in ALADIN).
- **RFPEPS**: maximum relative distance between the asymptot and the filtering function outside the window. Default value is 0.1 .
- **NFPWID**: width of the window where the filtering function is strongly decreasing; defined as a deviation. Default value is 12 if \Nc ≤ 24, 24 if \Nc > 24 and \Nc ≤ 48, 48 if \Nc > 48.
- **RFPFIL**: value of the filter for each zonal wavenumber and each subdomain.

- **RFPMAT**: filtering matrixes.
- **LFPBEL**: .TRUE. if "bell-shape" filter is active for other fields than geopotential, temperature or mean sea level pressure.
- **RFPBEL**: coefficient of the exponential function in the "bell-shape" filter for other fields than geopotential, temperature or mean sea level pressure.
- **LFPBEG**: .TRUE. if "bell-shape" filter is active for geopotential.
- **RFPBEG**: coefficient of the exponential function in the "bell-shape" filter for geopotential.
- **LFPBET**: .TRUE. if "bell-shape" filter is active for temperature.
- **RFPBET**: coefficient of the exponential function in the "bell-shape" filter for temperature.
- **LFPBEP**: .TRUE. if "bell-shape" filter is active for mean sea level pressure.
- **RFPBEP**: coefficient of the exponential function in the "bell-shape" filter for mean sea level pressure.
- **LFPBEH**: .TRUE. if "bell-shape" filter is active for relative moisture.
- **RFPBEH**: coefficient of the exponential function in the "bell-shape" filter for relative moisture.
- **LFPBED**: .TRUE. if "bell-shape" filter is active for "derivatives".
- **RFPBED**: coefficient of the exponential function in the "bell-shape" filter for "derivatives".
- **LFPBE** : .TRUE. if any of **LFPBE...** is .TRUE. .

The array below summarizes the existing "Bell-shaped" filters (default values are parenthesis):

TABLE 2.11

Field	Available in model	Switch (.TRUE. if active)	Coefficient
Derivatives		LFPBED	RFPBED
	ALADIN only	(.TRUE.)	(6.)
Geopotential		LFPBEG	RFPBEG
	ARPEGE/IFS	(.TRUE.)	(4.)
	ALADIN	(.TRUE.)	(6.)
Temperature		LFPBET	RFPBET
	ARPEGE/IFS	(.TRUE.)	(4.)
	ALADIN	(.TRUE.)	(6.)
Mean sea level pressure		LFPBEH	RFPBEP
	ARPEGE/IFS	(.TRUE.)	(4.)
	ALADIN	(.TRUE.)	(6.)
Relative humidity		LFPBEP	RFPBEP
	ARPEGE/IFS	(.TRUE.)	(4.)
	ALADIN	(.TRUE.)	(6.)
All other non-derivatives		LFPBEL	RFPBEL
	ARPEGE/IFS	(.FALSE.)	(4.)
	ALADIN	(.FALSE.)	(6.)

Variables **LFPBED** to **LFPBEL** are .FALSE. in the first part of a 927-type configuration **LFPSPEC**=.TRUE. .

Furthermore, in ARPEGE/IFS, it is also possible to perform *at the end* an overtruncation of each post-processed field. This overtruncation is equivalent to the so-called "THX" filter, but it is applied in the stretched spectral space and then $n0$ (see formula ([Ref: deffilterTHXARPEGE])) is equal to the variable **NPMAX** in the namelist **NAMDIM**. Note that this overtruncation is active only if **NPMAX** is less than **NSMAX**.

Namelist NAMFPF:

The following variables of **YOMFPF** are in namelist **NAMFPF**: **NFMAX**, **LFPFIL**, **RFPEPS**, **NFPWID**, **LFPBEL**, **RFPBEL**, **LFPBEG**, **RFPBEG**, **LFPBET**, **RFPBET**, **LFPBEP**, **RFPBEP**, **LFPBEH**, **RFPBEH**, **LFPBED**, **RFPBED**.

2.12.14 YOMFPG.

Variables defining the characteristics of the (transformed) output geometry. Variables are initialised in setup routine **setup**/**SUFPG**.

Variables concerning all kinds of output subdomains:

- **NFPMAX**: truncation order (DM-global).
  - If **CFPFMT**='GAUSS', **NFPMAX** is the truncation corresponding to the output grid; default value is **NSMAX**.
  - If **CFPFMT**='LELAM', **NFPMAX** is the meridian truncation; default value satisfies to the identity 3* **NFPMAX** +1 = **NLAT**.
  - If **CFPFMT**='LALON', **NFPMAX** is the truncation of the Gaussian grid which would have been defined by **NLAT** latitudes and **NLON** longitudes; default value satisfies to the identity 3* **NFPMAX** +1 = min(**NLAT**,**NLON**).
- **RFPLA**: latitudes of the output points (DM-local).
- **RFPLA1**: latitude of the first output point in a DM-global sense (DM-global).
- **RFPLANLAT1**: latitude of the output point number **NLAT**(1) in a DM-global sense (DM-global).
- **RFPLO**: longitudes of the output points (DM-local).
- **NFPRGPG**: effective number of output points (DM-global).
- **NFPRGPL**: DM-local effective number of output points for one processor (DM-local).
- **NFPRGPLX**: maximum of the **NFPRGPL** of all the processors (DM-global).
- **LFPOSHOR**: .TRUE. for actual horizontal post-processing (DM-global).

Variables concerning Gaussian grid or ALADIN subdomain as output subdomain:

- **NFPLEV** : number of levels; default is **NFLEVG** (DM-global).
- **FPMUCEN**: sine of the latitude of the pole of interest (DM-global); default value is **RMUCEN** for global outputs, 1 for limited area outputs.
- **FPLOCEN**: longitude of the pole of interest (DM-global); default value is **RLOCEN** for global outputs, 0 for limited area outputs.
- **FPVALH** : *A* coefficients of vertical system (DM-global); default value is **VAH**.
- **FPVBH** : *B* coefficients of vertical system (DM-global); default value is **VBH**.
- **FPVP00** : reference pressure (DM-global); default value is **VP00**.
- **RFPGM** : mapping factor (DM-local).
- **RFPNORX**: zonal component of rotation matrix for vector outputs (DM-local).
- **RFPNORY**: meridian component of rotation matrix for vector outputs (DM-local).

Variables concerning Gaussian grid as output subdomain:

- **NFPHTYP** (DM-global):
  - 0: regular grid.
  - 1: number of points proportional to $\sqrt{1-\mu_2}$.
  - 2: number of points read on namelist **NAMFPG**.
  - default value is **NHTYP** if **NLAT** = **NDGLG**, zero in the other cases.
- **FPNLGINC**: increment to get non-linear grid (DM-global); default value is 1.

- **NFPRGRI** : number of active points on a parallel (DM-global); default value is **NLOENG** if **NLAT** = **NDGLG**, **NLON** in the other cases.
- **NFPMEN** : maximum zonal wave number for a given parallel (DM-global).
- **FPSTRET** : stretching factor (DM-global); default value is **RSTRET** for global outputs, 1 for limited area outputs.
- **NFPTTYP** (DM-global):
  - 1: no tilting; pole of high resolution at the northern pole of the real Earth.
  - 2: tilting; pole of high resolution at a different location than the northern pole of the real Earth.

Default value is **NSTTYP** for global outputs, 1 for limited area outputs.

- **NFPQUAD** : quadrature (1 : Gauss ; 2 : Lobatto); (DM-global); default value is **NQUAD** for global outputs, 1 for limited area outputs.
- **RFPMU** : array containing the sines of the output Gaussian latitudes (DM-global).
- **LFPOLE** : .TRUE. if points at the poles are required; default value is .TRUE if **CFPFMT**='MODEL' or pseudo-configurations 927 and E927, .FALSE. elsewhere (DM-global).

Variables concerning ALADIN grid as output subdomain:

- **FPBETA**: angle of rotation in degrees (DM-global); default value is zero in ARPEGE/IFS, zero in ALADIN if **CFPFMT**='MODEL' and **EBETA** in ALADIN in the other cases.
- **NMFPMAX**: meridian truncation order (DM-global); default value satisfies to the identity 3* **NFPMAX** +1 = **NLAT** if **CFPFMT**='LELAM' and default value is 1 in the other cases.
- **LFPMAP**: .T./.F. if the domain is defined through its coordinates/wavelengths (DM-global).
- **FPLX**: wavelength in longitude (DM-global); default is 5448300. in ARPEGE/IFS if **CFPFMT**='LELAM', 0 in ARPEGE/IFS if **CFPFMT**='MODEL', 'LALON' or 'GAUSS', 0 in ALADIN if **CFPFMT**='GAUSS' and **ELX** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.
- **FPLY**: wavelength in latitude (DM-global); default is 5448300. in ARPEGE/IFS if **CFPFMT**='LELAM', 0 in ARPEGE/IFS if **CFPFMT**='MODEL', 'LALON' or 'GAUSS', 0 in ALADIN if **CFPFMT**='GAUSS' and **ELY** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.
- **NFPROTEQ**: parameter for rotation (DM-global).
  - 0: no rotation.
  - 1: the reference point of rotation (**FIR**,**LAR**) is rotated to equator, the north pole is on the new Greenwich meridian.

Default is 0 in ARPEGE/IFS, 0 in ALADIN if **CFPFMT**='GAUSS' and **NROTEQ** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.

- **FPLONR**: **LAR** geographical longitude of reference point of rotation, in degrees (DM-global); default value is zero in ARPEGE/IFS, 0 in ALADIN if **CFPFMT**='GAUSS' and **ELONR** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.
- **FPLATR**: **FIR** geographical latitude of reference point of rotation, in degrees (DM-global); default value is zero in ARPEGE/IFS, 0 in ALADIN if **CFPFMT**='GAUSS' and **ELATR** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.
- **FPLON0**: **LA0** geographical longitude of reference for the projection, in degrees (DM-global); default value is 0. in ARPEGE/IFS excepted in the case **CFPFMT**='LELAM' (various default values according to **CFPDOM**, see routine **SUFPG1**), 0. in ALADIN if **CFPFMT**='GAUSS' and **ELON0** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.

- **FPLAT0**: **FI0** geographical latitude of reference for the projection, in degrees (DM-global); default value is 0. in ARPEGE/IFS excepted in the case **CFPFMT**='LELAM' (various default values according to **CFPDOM**, see routine **SUFPG1**), 0. in ALADIN if **CFPFMT**='GAUSS' and **ELAT0** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.
- **FPRPK**: *K* projection parameter and definition (DM-global); default value is -9. in ARPEGE/IFS excepted in the case **CFPFMT**='LELAM' (various default values according to **CFPDOM**, see routine **SUFPG1**), -9. in ALADIN if **CFPFMT**='GAUSS' and **ERPK** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.
- **NFPSOTRP**: isotropy parameter under projection (DM-global); default value is 0. in ARPEGE/IFS excepted in the case **CFPFMT**='LELAM' (generally 1.), 0. in ALADIN if **CFPFMT**='GAUSS' and **NSOTRP** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.
- **NFPGIV0**: choice of reference point for projection (DM-global); default value is zero in ARPEGE/IFS, 0. in ALADIN if **CFPFMT**='GAUSS' and **NGIV0** in ALADIN if **CFPFMT**='MODEL', 'LALON' or 'LELAM'.

Variables concerning ALADIN or 'LALON' grid as output subdomain:

- **RFPDX**: grid size in meters along *x* if projection (ALADIN grid); longitude increment in radians if spherical geometry (DM-global).
- **RFPDY**: grid size in meters along *y* if projection (ALADIN grid); latitude increment in radians if spherical geometry (DM-global).

Namelist NAMFPG:

The following variables of **YOMFPG** are in namelist **NAMFPG**: **NFPMAX**, **NFPLEV**, **FPMUCEN**, **FPLOCEN**, **FPVALH**, **FPVBH**, **NFPHTYP**, **NFPRGRI**, **FPSTRET**, **NFPTTYP**, **NFPQUAD**, **FPBETA**, **LFPOLE**, **FPVP00**, **NMFPMAX**, **FPLX**, **FPLY**, **NFPROTEQ**, **FPLONR**, **FPLATR**, **FPLON0**, **FPLAT0**, **FPRPK**, **NFPSOTRP**, **NFPGIV0**, **FPNLGINC**.

2.12.15 YOMFPIOS.

Contains control variables for FULL-POS IO scheme. Variables are initialised in setup routine **setup**/**SUFPIOS**.

Writing to output files: variable NFPXFLD.

To write post-processed fields on an output file, you first extract them from an array (or a work file). Rather than extracting the fields one after the other, the fields are grouped in packets, and these packets of fields are extracted one after the other. You can specify the maximum number of fields in a packet by modifying the variable **NFPXFLD** in the namelist **NAMFPIOS**. The default value is the maximum possible depending on the number of fields to produce. Raising the value of **NFPXFLD** should save CPU time to the detriment of the memory cost, and vice versa.

Variables for auxiliary fields:

- **LIOFPR** : switch for IO scheme (DM-global).
- **CIOFPR** : pathname (DM-global).
- **NIOFPR** : MIO package file reference (DM-global).
- **NBLIOFPR**: buffer length for IO (DM-local).
- **NIOBFFPR**: number of buffers for IO (DM-global).
- **NPCKFFPR**: packing factor (DM-global).
- **NEXPBFPR**: number of bits used for exponent when packing (DM-global).

Variables for working fields:

- **LIOFPW** : switch for IO scheme (DM-global).
- **CIOFPW** : pathname (DM-global).
- **NIOFPW** : MIO package file reference (DM-global).
- **NBLIOFPW**: buffer length for IO (DM-local).
- **NIOBFFPW**: number of buffers for IO (DM-global).
- **NPCKFFPW**: packing factor (DM-global).
- **NEXPBFPW**: number of bits used for exponent when packing (DM-global).

Variables for dynamics fields:

- **LIOFPD** : switch for IO scheme (DM-global).
- **CIOFPD** : pathname (DM-global).
- **NIOFPD** : MIO package file reference (DM-global).
- **NBLIOFPD**: buffer length for IO (DM-local).
- **NIOBFFPD**: number of buffers for IO (DM-global).
- **NPCKFFPD**: packing factor (DM-global).
- **NEXPBFPD**: number of bits used for exponent when packing (DM-global).

Variables for physics fields:

- **LIOFPP** : switch for IO scheme (DM-global).
- **CIOFPP** : pathname (DM-global).
- **NIOFPP** : MIO package file reference (DM-global).
- **NBLIOFPP**: buffer length for IO (DM-local).
- **NIOBFFPP**: number of buffers for IO (DM-global).
- **NPCKFFPP**: packing factor (DM-global).
- **NEXPBFPP**: number of bits used for exponent when packing (DM-global).

Variables for fields from APACHE:

- **LIOFPA** : switch for IO scheme (DM-global).
- **CIOFPA** : pathname (DM-global).
- **NIOFPA** : MIO package file reference (DM-global).
- **NBLIOFPA**: buffer length for IO (DM-local).
- **NIOBFFPA**: number of buffers for IO (DM-global).
- **NPCKFFPA**: packing factor (DM-global).
- **NEXPBFPA**: number of bits used for exponent when packing (DM-global).

Additional remarks and some default values:

For the horizontal post-processing, the data flows have been separated in 4 parts:

- post-processed dynamical data.
- post-processed physical and fluxes data.
- weights for horizontal interpolations.
- output geometry and climatology.
- horizontally pre-processed dynamical data (for post-processing on height levels or hybrid levels).

To spare memory space, each of these data flows can be controlled by a specific I/O scheme. By default, no I/O scheme is activated. If you wish to activate I/O schemes, then use the variables which are in the namelist **NAMFPIOS** as the array below describes it (defaults values are the ones in parenthesis).

TABLE 2.12

Data	.TRUE. to activate I/O scheme	Work file name	Packing factor	Number of bits used for exponent when packing
Post-processed dynamical fields	LIOFPD (.FALSE.)	CIOFPD ('miofpdyn')	NPCKFFPD (1)	NEXPBFPD (6)
Post-processed phys flds/fluxes	LIOFPP (.FALSE.)	CIOFPP ('miofpphy')	NPCKFFPP (1)	NEXPBFPP (6)
Weights	LIOFPW (.FALSE.)	CIOFPW ('miofpwork')	NPCKFFPW (1)	NEXPBFPW (6)
Output geometry, climatology	LIOFPR (.FALSE.)	CIOFPR ('miofpaux')	NPCKFFPR (1)	NEXPBFPR (6)
Pre-processed dynamical data	LIOFPA (.FALSE.)	CIOFPA ('miofpapa')	NPCKFFPA (1)	NEXPBFPA (6)

Namelist NAMFPIOS:

The following variables of **YOMFPIOS** are in namelist **NAMFPIOS**: **NFPXFLD**, **CIOFPR**, **CIOFPW**, **CIOFPD**, **CIOFPP**, **CIOFPA**, **LIOFPR**, **NPCKFFPR**, **NEXPBFPR**, **LIOFPW**, **NPCKFFPW**, **NEXPBFPW**, **LIOFPD**, **NPCKFFPD**, **NEXPBFPD**, **LIOFPP**, **NPCKFFPP**, **NEXPBFPP**, **LIOFPA**, **NPCKFFPA**, **NEXPBFPA**.

2.12.16 YOMFPOLE.

Useless, has to be removed.

2.12.17 YOMFPOP.

Contains post-processing file-handling variables. Variables are initialised in setup routine **setup**/**SUFPOPH**. No variable in namelist.

List of variables:

- **CFPFN**: path file name for output files (DM-global).
- **CFPCA**: names of output frames (DM-global).

2.12.18 YOMFPSC2.

Contains control variables for **SCAN2MSM** or **SCAN2MDM** FULL-POS rows. Variables are initialised in setup routine **setup**/**SUFPSC2**.

List of variables:

- **NFPLENR**: number of active points in each post-processing row (DM-local).
- **NFPROMAG**: working dimension for horizontal post-processing (DM-global if its value is specified in the namelist **NAMFPSC2**, DM-local in the contrary, default value depends on the processor if distributed memory).
- **NFPROMAL**: working dimension for horizontal post-processing, DM-local version of **NFPROMAG**.
- **NFPBSR**: number of sub-rows in each post-processing row (DM-local).
- **NFPTBSRL**: total number of sub-rows in post-processing buffer (DM-local).

Additional remarks about variables NFPROMAG and NFPROMAL:

Variable **NFPROMAG** is very useful to find a compromise between the vectorization and the memory cost in the post-processing for the lagged computations (horizontal interpolations in **HPOSLAG** and additional calculations in **ENDVPOS**), and it is also very useful to make efficient I/O schemes. The variable **NFPROMAL** represents the horizontal depth of work in the lagged computations of post-processing; it is obvious that this variable has some similarity with the famous variable **NPROMA** (in namelist **NAMDIM**) or the less famous variable **NSEGM** (in namelist **NAM926**), but its use is more flexible. For shared memory **NFPROMAL**=**NFPROMAG** if **NFPROMAG** is an odd integer, **NFPROMAL**=**NFPROMAG**+1 if **NFPROMAG** is an even integer; this is also the case if **NFPROMAG** is taken to its (DM-local) default value for distributed memory, this is not necessary to define a unic DM-global value (but desirable in the future). For distributed memory when a DM-global value of **NFPROMAG** is given in the namelist **NAMFPSC2**, **NFPROMAL** depends on the processor, its maximum value is equal to **NFPROMAG** if **NFPROMAG** is an odd integer, **NFPROMAG**+1 if **NFPROMAG** is an even integer. Variable **NFPROMAG** is used only in **SUFPSC2**, the remainder of the code uses only **NFPROMAL**. The way how the post-processing data flow is managed in the post-processing arrays/work files is explained in detail in the sections 'Some shared memory features' and 'Some distributed memory features'. For summary:

- The model data is split in packets of points of fixed size **NPROMA**; each model packet is treated by one task, and is independent from the other packets. For distributed memory one processor treats **NGPTOT** model grid-points, with a subdivision into **NPROMA**-packets, but this subdivision is active for not lagged computations only.
- Each output point (i.e. where post-processing is performed) is related to one model packet: the one which contains the model point which is the nearest from it. That way, the output points are sorted out, so that each post-processing packet (i.e. the group of all the output points which are related to the same model packet) has a specific size. For shared memory (resp. distributed memory) the model-packets used to compute **NFPROMAG** and **NFPROMAL** are the **NPROMA**-packets (resp. the **NGPTOT**-packets).
- In order to use the I/O subroutines already available in ARPEGE/IFS, each post-processing packet has been split in sub-rows of fixed size **NFPROMAL**. Each post-processing packet is treated by one task which performs a loop on the number of sub-rows in the current post-processing packet.
- Note that an empty post-processing packet is characterized by a number of sub-rows equal to zero; as a consequence, empty post-processing packets do not lead to memory spill. On the opposite, each non-empty post-processing packet contains a specific number of unused words.

Searching for a compromise between vectorization and memory cost, the value of **NFPROMAG** can be any integer from 1 to the size of the biggest post-processing packet. The bigger **NFPROMAG** will be, the more the vectorization will be efficient, and the more memory will be needed. The default value of **NFPROMAG** is the mean size of the post-processing packets if shared memory, the total number of post-processing points treated by the current processor if distributed memory, as this value appeared to be the best compromise. However, if the I/O schemes are activated, the size of the biggest post-processing packet may be a more efficient value. In the listing, the user will find more information to find out the value to adopt: just before "COMMON YOMFPSC2" is printed out, the following information is written (information depends on the processor for distributed memory but is not very useful in this case because there is only one value per processor which is used to compute the following statistics: the standard deviation is zero):

- size of the smallest post-processing packet.
- size of the biggest post-processing packet.
- mean size of the post-processing packets.
- standard deviation of the size of the post-processing packets.

You can also know the memory you have lost in the post-processing arrays: just locate "TOTAL MEMORY LOST IN WORDS =" .

Namelist NAMFPSC2:

The following variable of **YOMFPSC2** is in namelist **NAMFPSC2**: **NFPROMAG**.

2.12.19 YOMFPSC2B.

Contains control variables for **SCAN2MSM** or **SCAN2MDM** FULL-POS sub-rows. Variables are initialised in setup routine **setup**/**SUFPSC2B**. No variable in namelist.

List of variables:

- **NFPROF**: number of active points in each post-processing sub-row (DM-local).
- **NFPINT**: number of points to be computed through 12-points interpolations (**NFPINT**(1,.)) and through averaging of a box (**NFPINT**(2,.)). DM-local variable.
- **NFPSORT**: pointer of the interlaced post-processing points in the sorted array (DM-local).
- **NFPSTAP**: sub-row start pointer for each row (DM-local).

2.12.20 YOMFPSP.

Contains FULL-POS spectral arrays. Variables are initialised after direct spectral transforms. All variables are DM-global. No variable in namelist.

List of variables:

- **SPAFP**: underived vertically post-processed fields.
- **SPBFP**: derived vertically post-processed fields for all required subdomains.
- **SPDFP**: derived vertically post-processed fields for one subdomain.

2.12.21 YOMFPT0.

Contains buffer **FPTS0**: Auxiliary surface grid point array for post-processing on height levels above an output orography and for surface horizontal post-processing above an output orography (DM-local). Variable is computed in routine **pp_obs**/**GPOS**. No variable in namelist.

2.12.22 YOMIOS.

Some variables in **YOMIOS** (which does not contain only FULL-POS variables) can be useful for FULL-POS. These variables are DM-global. For the vertical post-processing, the data flows have been separated in 2 parts:

- fitted post-processed fields.
- unfitted post-processed fields.

To spare memory space, the data flow of unfitted post-processed fields can be controlled by a specific I/O scheme. By default, this I/O scheme is activated only at ECMWF (**LECMWF**=.TRUE.). If you wish to activate/deactivate it, just set **LIOGAUX**=.TRUE./.FALSE. in the namelist **NAMCT0**. Then you can change other related variables in the namelist **NAMIOS**:

- **CIOGAUX**: work file name of unfitted vertically post-processed fields; default is 'miogaux'.
- **NPCKFGX**: packing factor for I/O on vertically post-processed fields; default is 1.
- **NEXPBGX**: number of bits used for exponent when packing I/O on vertically post-processed fields; default is 6.

Other useful variables in YOMIOS and NAMIOS:

- **CIOFOU1**: work file name for Fourier data.
- **NBLFOU** : segment length for complex I/O on Fourier data.
- **CIOLPOL**: work file name for Legendre polynomials.
- **CIOTMDT**: work file name of grid-point fields at t-dt.
- **NPCKFT9**: packing factor for I/O on grid-point fields at t-dt.
- **NEXPBT9**: number of bits for exp. when packing grid-point fields at t-dt.
- **CIOGPUA**: work file name of upper air grid-point fields.
- **NPCKFGU**: packing factor for I/O on upper air grid-point fields.
- **NEXPBGU**: number of bits for exp. when packing upper air grid-point fields.
- **CIOGPP** : work file name of surface fields.
- **NPCKFGP**: packing factor for I/O on surface fields.
- **NEXPBGP**: number of bits for exp. when packing surface fields.
- **CIOCFU** : work file name of cumulated fluxes.
- **NPCKFCF**: packing factor for I/O on cumulated fluxes.
- **NEXPBCF**: number of bits for exp. when packing cumulated fluxes.
- **CIOXFU** : work file name of instantaneous fluxes.
- **NPCKFXF**: packing factor for I/O on instantaneous fluxes.
- **NEXPBXF**: number of bits for exp. when packing instantaneous fluxes.
- **CIOSPEC**: work file name of saved spectral data.
- **CIOSUA** : work file name of saved upper air grid-point data.
- **CIOSSU** : work file name of saved surface data.
- **CIOSCF** : work file name of saved cumulated fluxes data.
- **CIOSXF** : work file name of saved instantaneous fluxes data.

Default value of a variable of the type **CIONAME** (for ex. **CIOGPP**) is 'mioname' (for ex. 'miogpp'). Default value of **NBLFOU** depends on configuration. Default value of **NPCK...** variables is 1. Default value of **NEXP...** variables is 6. See routine **SUIOS** for more details.

2.12.23 YOMMP.

Variables specific to the distributed memory environment. Some of these variables can be used also for shared memory. The following variables of **YOMMP** (which does not contain only FULL-POS variables) can be useful for FULL-POS. Variables are generally computed in **setup**/**SUMP0** (called by **setup**/**SU0Y0MA**), **setup**/**SUMP**, sometimes **setup**/**SUCT0**.

Variables describing distributed memory parallelization:

- **MYPOS**(1): own position in x-direction (DM-global).
- **MYPOS**(2): own position in y-direction (DM-global).
- **MYPROC**: own processor, quasi 1D position in torus (DM-local).
- **MYSETA**: own processor set A (set A DM-local, set B DM-global).
- **MYSETB**: own processor set B (set A DM-global, set B DM-local).
- **MYPID**: own process identifier (DM-local).
- **MYFATHER**: father process (DM-local).
- **NSONS**: number of son processes (DM-local).
- **NPHASE**: number of phases in the recursive transposition (DM-global). Default value is 1.
- **NLAGA**: lagging factor for tranpositions in A-direction (DM-global). Default value is 0.
- **NLAGB**: lagging factor for tranpositions in B-direction (DM-global). Default value is 0.

- **NSLPAD**: number of pad words initialised to a huge number at either of side of the interpolation buffer halo, used to trap halo problem (DM-global). Default value is 0.
- **LSPLITIN**: input data provided in sequential files if .TRUE. or in directories if .FALSE. (DM-global). Default value is .FALSE. .
- **LSPLITOUT**: output data provided in sequential files if .TRUE. or in directories if .FALSE. (DM-global). Default value is .TRUE. .
- **NSTRIN**: number of processors required to perform input processing (DM-global). Default value is 1.
- **NSTROUT**: number of processors required to perform output processing (DM-global). Default value is 0.
- **NFLDIN**: number of input fields to be buffered during distribution (DM-global). Default value is 0.
- **NFLDOUT**: number of output fields to be buffered during gathering (DM-global). Default value is 0.
- **NPRCIDS**: array containing the process identificators (DM-local).
- **LAPPLE**: defines apple or orange grid point decomposition (DM-global). Default value is .FALSE. .
- **LSPLIT**: .TRUE./.FALSE.: latitudes are shared/not shared between sets (DM-global). Default value is .TRUE. for configurations between 100 and 199, .FALSE. for other configurations.
- **LSLSYNC**: .TRUE. if communication between processors for interpolation buffers reads/writes in synchronisation (DM-global). Default value is .TRUE. .

Variables describing the partitioning of data:

- **NDWIDE**: width of interpolation buffer halo region (DM-global). Default value is 32.
- **NPROCM**: gives process which is responsible for the zonal wave number *m* (DM-global).
- **NUMPROCGP**: gives processor which is responsible for model grid point (DM-global).
- **NUMPROCFP**: gives processor which is responsible for FULL-POS horizontal interpolation point *jfp*, where *jfp* = 1 to **NFPRGPG** (DM-global).
- **NUMPP**: tells for how many wave numbers the processes are responsible for the truncation **NSMAX** (DM-local).
- **NALLMS**: global wave numbers for all processes (DM-global).
- **NPTRMS**: pointer to first wave number of given process in **NALLMS** (DM-global).
- **NLATLS**: first latitude for which current process calculates Legendre polynomials (DM-global).
- **NLATLE**: last latitude for which current process calculates Legendre polynomials (DM-global).
- **NPROCL**: gives process which is responsible for latitude in grid-point space (DM-global).
- **NPROCLF**: gives process which is responsible for latitude in Fourier space (DM-global).
- **NULTPP**: number of latitudes in given process in grid-point space (DM-global).
- **NULTPPF**: number of latitudes in given process in Fourier space (DM-global).
- **MYLATS**: latitude numbers mapped to current process in grid-point space (DM-global).
- **NPTRLS**: pointer to first global latitude of given process in grid-point space (DM-global).
- **NPTRLSF**: pointer to first global latitude of given process in Fourier space (DM-global).
- **NFRSTLAT**: first latitude of each set in grid-point space (DM-global).
- **NFRSTLATF**: first latitude of each set in Fourier space (DM-global).
- **NFRSTLOFF**: offset for first lat of own set in grid-point space (DM-local).
- **NFRSTLOFFF**: offset for first lat of own set in Fourier space (DM-local).
- **NLSTLAT**: last latitude of each set in grid-point space (DM-global).
- **NLSTLATF**: last latitude of each set in Fourier space (DM-global).
- **NPTRFRSTLAT**: pointer to first latitude in grid-point space (DM-global).

- **NPTRFRSTLATF**: pointer to first latitude in Fourier space (DM-global).
- **NPTRLSTLAT**: pointer to last latitude in grid-point space (DM-global).
- **NPTRLSTLATF**: pointer to last latitude in Fourier space (DM-global).
- **NPTRFLOFF**: offset for pointer to first latitude of own set in grid-point space (DM-local).
- **NPTRFLOFFF**: offset for pointer to first latitude of own set in Fourier space (DM-local).
- **NPTRLAT**: pointer to start of latitude in grid-point space (DM-global).
- **NPTRLATF**: pointer to start of latitude in Fourier space (DM-global).
- **LSPLITLAT**: .TRUE. if latitude is split over two A sets (DM-global).
- **MYFRSTACTLAT**: first actual latitude of each processor in grid-point space (DM-global).
- **MYLSTACTLAT**: last actual latitude of each processor in grid-point space (DM-global).
- **MYSENDA**: processors to which current processor will send messages during the (recursive) A-transposition (set A DM-local, set B DM-global).
- **MYRECVA**: processors from which current processor will receive messages during the (recursive) A-transposition (set A DM-local, set B DM-global).
- **MYSENDB**: processors to which current processor will send messages during the (recursive) B-transposition (DM-local).
- **MYRECVB**: processors from which current processor will receive messages during the (recursive) B-transposition (DM-local).
- **MYSENDSETA**: process-set to which current processor will send messages during the (recursive) A-transposition (set A DM-local, set B DM-global).
- **MYRECVSETA**: process-set to which current processor will receive messages during the (recursive) A-transposition (set A DM-local, set B DM-global).
- **MYSENDSETB**: process-set to which current processor will send messages during the (recursive) B-transposition (set A DM-global, set B DM-local).
- **MYRECVSETB**: process-set to which current processor will receive messages during the (recursive) B-transposition (set A DM-global, set B DM-local).
- **MYSENDG**: processors to which current processor will send messages during global communication (DM-local).
- **MYRECVG**: processors from which current processor will receive messages during global communication (DM-local).
- **MYMSGS**: identification of the origin of the messages during recursive transposition (DM-local).
- **NPTRSV**: pointer to first spectral wave column on given process for **NSMAX** arrays (DM-global).
- **NPTRSVV**: as **NPTRSV** but for full *m*-columns (DM-global).
- **NPTRMF**: distribution of *m*-columns among B sets used for semi-implicit calculations on this processor (full *m*-columns case, set A DM-global, set B DM-local).
- **NSPSTAF**: pointer to where each *m*-column starts; used for semi-implicit (full *m*-columns case, set A DM-global, set B DM-local).
- **NUMLL**: distribution of levels on B-processor set (DM-global).
- **NPTRLL**: pointer to first level on this B-processor set (set A DM-global, set B DM-local).
- **NPSP**: equals 1 if pressure is handled by this B-processor set (set A DM-global, set B DM-local).
- **NPSURF**: equals 1 if surface pressure is handled by this B-processor set (set A DM-global, set B DM-local).
- **NBSETLEV**: on which B-set do the level belongs (set A DM-global, set B DM-local).
- **NBSETSP**: min(**NFLEVG**+1,**NPROCB**) (set A DM-global, set B DM-local).
- **MYLEVS**: level numbers mapped to current process (set A DM-global, set B DM-local).
- **NVMODIST**: normal modes mapped to the different B-sets (set A DM-global, set B DM-local).
- **NSPEC2V**: number of spectral columns computed by this process for **NSMAX** arrays (DM-local).

- **NSPEC2VF**: number of spectral columns computed by this process if complete columns are required (DM-local).
- **NSTA**: start position of grid columns on latitudes (set A DM-global, set B DM-local).
- **NONL**: number of grid columns on latitudes (set A DM-global, set B DM-local).
- **NGPSET**: number of grid points of this processor in grid point space which also belong to it in Fourier space (set A DM-global, set B DM-local).
- **NAPLAT**: number of apple latitudes at the poles (DM-global). Default value is 0.
- **NGPTRSEND**: defines grid columns to be sent to each B-set during **TRGTOL** (DM-local).
- **NGPTRRECV**: defines B-sets to receive from grid columns during **TRGTOL** (DM-local).

SUFPCSET and SUFPRSET variables (based on NFPWIDE) used for FULL-POS interpolations:

- **NAFPB1**: local inner dimension of interpolation buffer used for FULL-POS interpolations (DM-local).
- **NFPSTA**: interpolation buffer start position of grid columns on latitudes (set A DM-global, set B DM-local).
- **NFPONL**: interpolation buffer number of grid column on latitudes (set A DM-global, set B DM-local).
- **NFPOFF**: interpolation buffer offset to start of each row in **FPBUF1** and **FPBUF2** (set A DM-global, set B DM-local).
- **NFPEXT**: interpolation buffer extend latitudes over poles support (set A DM-global, set B DM-local).
- **NFPSTLAT**: latitudinal position in the interpolation buffer during the data sending (DM-local).
- **NFPSTLON**: longitudinal position in the interpolation buffer during the data sending (DM-local).
- **NFPRSTLAT**: latitudinal position in the interpolation buffer during the data reception (DM-local).
- **NFPRSTLON**: longitudinal position in the interpolation buffer during the data reception (DM-local).
- **NFPSENDNUM**: length of the sent interpolation buffer (DM-local).
- **NFPRECVNUM**: length of the received interpolation buffer (DM-local).
- **NFPCORE**: offsets to core region points in interpolation buffer (set A DM-global, set B DM-local).

Some other variables:

- **NCOMBFLEN**: size of communication buffer. This is the maximum per processor buffer space (in words) that the ARPEGE/IFS should use for one or more sends before receives are issued from destination processors (DM-global). Default value is 1800000.
- **LBIDIR**: .TRUE. if bi-directional transpositions are preferred (DM-global). Only implemented in the A-direction and require even number of processors. Default value is .FALSE. .
- **LSNDS**: logical indicating whether to send (resp. receive) data to southern set in **TRLTOG** (resp. **TRGTOL**) (DM-local).
- **LSNDN**: logical indicating whether to send (resp. receive) data to northern set in **TRLTOG** (resp. **TRGTOL**) (DM-local).
- **LRCVS**: logical indicating whether to receive (resp. send) data from southern set in **TRLTOG** (resp. **TRGTOL**) (DM-local).
- **LRCVN**: logical indicating whether to receive (resp. send) data from northern set in **TRLTOG** (resp. **TRGTOL**) (DM-local).
- **NSNDS**: number of messages to send (resp. receive) to southern set in **TRLTOG** (resp. **TRGTOL**) (DM-local).

- **NSNDN**: number of messages to send (resp. receive) to northern set in **TRLTOG** (resp. **TRGTOL**) (DM-local).
- Other distributed memory environment variables which are currently in another place, and for which YOMMP would be a better place:
- **NUMP** (in **YOMDIM**): number of spectral waves handled by this processor (DM-local). **NUMP**=**NSMAX**+1 for shared memory.
- **NUMCP** (in **YOMDIM**): same as **NUMP**, but related to **NCMAX** (DM-local). **NUMCP**=**NCMAX**+1 for shared memory.
- **NUMTP** (in **YOMDIM**): same as **NUMP**, but related to **NTMAX** (DM-local). **NUMTP**=**NTMAX**+1 for shared memory.
- **MYMS** (in **YOMLAP**): actual wave numbers handled by each processor (DM-global).

Namelist NAMPAR1:

Initialize variables that control layout of distribution. The following variables of **YOMMP** are in namelist **NAMPAR1**, namelist reading and default values calculations are done in **SUMP0**: **LSPLIT**, **LSLSYNC**, **LAPPLE**, **LBIDIR**, **NSLPAD**, **NAPLAT**, **NPHASE**, **NLAGA**, **NLAGB**, **NSTRIN**, **NSTROUT**, **NFLDIN**, **NFLDOUT**, **NVALAG**, **NCOSTLAG**, **NLAGBDY**, **LSPLITIN**, **LSPLITOUT**, **NCOMBFLEN**

2.12.24 YOMMPG.

Variables specific to the distributed memory environment. DM-global variables mainly used to handle reading and writing of grib data. Some of these variables can be used also for shared memory. Some variables in **YOMMPG** (which does not contain only FULL-POS variables) can be useful for FULL-POS. No variable in namelist.

- **NPOSSP**: defines partitioning of global spectral fields among processors (DM-global).
- **NPOSCP**: as **NPOSSP** for **NCMAX** arrays (DM-global).
- **NPOSTP**: as **NPOSSP** for **NTMAX** arrays (DM-global).
- **NDIM0G**: defines partitioning of global spectral fields among processors (DM-global).

2.12.25 YOMOP.

Some variables in **YOMOP** (which does not contain only FULL-POS variables) and namelist **NAMOPH** can be useful for FULL-POS.

- **LINC**: .T. to get time stamp in hours rather than in time steps (default value is .FALSE.). DM-global variable.

2.12.26 YOMPFPB.

Contains buffer for fully post-processed physical fields. Variables are initialised in setup routine **setup**/**SUFPSC2B**. No variable in namelist.

List of variables:

- **NLENPFPBL**: length of buffer (DM-local).
- **GPFPBUF** : buffer (DM-local).
- **NSTAPFPB** : start adresses for post-processed packets of points within buffer (DM-local).

2.12.27 YOMRFPB.

Contains buffer of auxiliary fields (geometry, climatology) for horizontal post-processing. Variables are initialised in setup routines **setup**/**SURFPBUF** for buffer **RFPBUF** and **setup**/**SUFPSC2B** for other variables. No variable

in namelist.

List of variables:

- **NLENRFPBL**: length of buffer (DM-local).
- **RFPBUF** : buffer containing output climatology and geometry (DM-local).
- **NSTARFPB** : start adresses for post-processed packets of points within buffer (DM-local).

2.12.28 YOMRFPDS.

Description of auxiliary fields for horizontal post-processing. Variables are initialised in setup routine **setup**/**SURFPDS**. Variables are DM-global. No variable in namelist.

Field pointers:

- **MFPGMO** : for output mapping factor.
- **MFPGNXO**: for output *x*-component of rotation matrix.
- **MFPGNYO**: for output *y*-component of rotation matrix.
- **MFPLSMO**: for output land-sea mask.
- **MFPFISO**: for output surface geopotential.
- **MFPFISI**: for interpolated input surface geopotential.
- **MFPCSTO**: for output climatological surface temperature.
- **MFPCSWO**: for output climatological relative surface moisture.
- **MFPSDO** : for output climatological snow cover.
- **MFPRSTO**: for output relaxation deep soil temperature.
- **MFPRSWO**: for output relaxation deep soil moisture.
- **MFPDSTO**: for output climatological deep soil temperature.
- **MFPDSWO**: for output climatological relative deep soil moisture.
- **MFPALBO**: for output albedo.
- **MFPEMIO**: for output emissivity.
- **MFPDEVO**: for output (standard deviation) $\times g$.
- **MFPVEGO**: for output vegetation.
- **MFPGZ0O**: for output $z_0 \times g$.
- **MFPANIO**: for output anisotropy.
- **MFPDIRO**: for output direction.

Other variables:

- **NFPVSO**: number of climatological fields.
- **CFPVSO**: ARPEGE field names of the climatological fields.
- **NRFPOS**: total number of fields in **RFPBUF**.

2.12.29 YOMSC2.

Contains parameters used to control vectorisation and memory space. Some of these parameters are used in FULL-POS. No variable in namelist.

- **NSLBCT**: packet control array for **SCAN2MSM** or **SCAN2MDM** (DM-local). Packets are **NPROMA**-packets for shared memory, **NGPTOT**-packets for distributed memory.
  - **NSLBCT**(1,.): start row of packet.
  - **NSLBCT**(2,.): end row of packet.
  - **NSLBCT**(3,.): 1 if non-lagged grid-point computations on packet, 0 if not.
  - **NSLBCT**(4,.): 1 if lagged grid-point computations on packet, 0 if not.

- **NSLBCT**(5,.): usage of **SLBUF1** for packet (semi-Lagrangian scheme).
- **NSLBCT**(6,.): usage of **SLBUF2** for packet (semi-Lagrangian and Eulerian schemes).
- **NSLBCT**(7,.): usage of **FPBUF1** for packet (FULL-POS interpolations for fields other than surface physics).
- **NSLBCT**(8,.): usage of **FPBUF2** for packet (FULL-POS interpolations for surface physics).
- **NSLBCT**(5,.) to **NSLBCT**(8,.) are useless for distributed memory. For distributed memory **NSLBCT**(1,1)=**NSLBCT**(2,1)=**NSLBCT**(3,1)=**NSLBCT**(4,1)=1, the other values are useless.

- **NSLBR**: number of packets of rows (DM-local). Packets are **NPROMA**-packets for shared memory, **NGPTOT**-packets for distributed memory. **NSLBR**=1 for distributed memory.
- **NLOCKFP1**: multitasking lock for interpolation buffer **FPBUF1** used in FULL-POS. Useless for distributed memory.
- **NLOCKFP2**: multitasking lock for interpolation buffer **FPBUF2** used in FULL-POS. Useless for distributed memory.
- **NLOCKOBL**: multitasking lock for **COBSLAGAD** and **COBSLAG**. Useless for distributed memory.
- **NFPB1EV**: event for interpolation buffer **FPBUF1** used in FULL-POS. Useless for distributed memory.
- **NFPB2EV**: event for interpolation buffer **FPBUF2** used in FULL-POS. Useless for distributed memory.
- **NWEVFP1**: number of tasks waiting for interpolation buffer **FPBUF1** event used in FULL-POS. Useless for distributed memory.
- **NWEVFP2**: number of tasks waiting for interpolation buffer **FPBUF2** event used in FULL-POS. Useless for distributed memory.
- **NSLWIDE**: number of rows the model lagged part runs behind (DM-global).
- **NFPWIDE**: number of rows the FULL-POS lagged part runs behind (DM-global).
- **NFPB1**:
  - for shared memory: number of buffers (last dimension) for interpolation buffer **FPBUF1** used in FULL-POS.
  - for distributed memory: used to dimension an interpolation buffer but only in the non lagged part of calculations; **NFPB1**=**NGPBLKS** (DM-local).
- **NFPB2**:
  - for shared memory: number of buffers (last dimension) for interpolation buffer **FPBUF2** used in FULL-POS.
  - for distributed memory: used to dimension an interpolation buffer but only in the non lagged part of calculations; **NFPB2**=**NGPBLKS** (DM-local).
- **NSTABUF**: start address for each row in grid-point buffers (DM-local).
- **NLATPBF**: packet pointer for each row in interpolation buffers (DM-local).
- **NDIST**: start address for each row in grid-point calculations (DM-local).
- **NDIEND**: end address for each row in grid-point calculations (DM-local).
- **NCIST**: start address for each row of grid-point field in buffer (DM-local).
- **NCIEND**: end address for each row of grid-point field in buffer (DM-local).
- **NBFPFP1** (resp. **NBFPFP2**):
  - elements (1,.): pointer to packet contained in each buffer (FPBUF1, resp. FPBUF2).
  - elements (2,.): remaining users of buffer (FPBUF1, resp. FPBUF2).

DM-global variables.

- **NSC2EVH**: event for each row indicating non-lagged part complete for FULL-POS. Useless for distributed memory.

2.12.30 YOMVFP.

Contains switches to use FULL-POS in an incremental variational application. Variables are initialised in setup routine **var**/**SUVAR** (move to directory **setup** will be desirable in the future). Variables are DM-global.

List of variables:

- **LARCHFP**: .T. => FULL-POS will be used for communication between the high and low resolution run. Default is .F.
- **LREFFP** : .T. => the reference field is post-processed at high resolution. Default is .F.
- **LRFTLFP**: .T. => the reference field is post-processed at low resolution. Default is .F.
- **LANAFP** : .T. => the final (analysis) field is post-processed. Default is .F.

Namelist NAMVFP:

The following variable of **YOMVFP** is in namelist **NAMVFP**: **LARCHFP**.

2.12.31 YOMVPOS.

Contains control variables for vertical post-processing. Variables are initialised in setup routine **setup**/**SUVFPOS** and can be modified by routine **transform**/**UPDVPOS**. All variables are DM-global. No variable in namelist.

Control variables for post-processing of 3D variables.

- **LFIT3** : .T./.F. if 3D post-processed fields are fitted/not fitted.
- Scalar variables **NADD...**: address in array **GT1**.
  - **NADDWIND** : address of wind.
  - **NADDPSI** : address of potential velocity.
  - **NADDDIV** : address of divergence.
  - **NADDVOR** : address of relative vorticity.
  - **NADDABS** : address of absolute vorticity.
- **NXFPLEV** : number of post-processing levels.
- **XFPLEV** : values of post-processing levels. Array dimensioned to **NXFPLEV** .
- Scalar variables **NLEV...**: number of post-processing (pp) levels for different variables.
  - **NLEVGEOP** : number of pp levels for geopotential.
  - **NLEVTEMP** : number of pp levels for temperature.
  - **NLEVUMOM** : number of pp levels for *U*-wind component.
  - **NLEVVMOM** : number of pp levels for *V*-wind component.
  - **NLEVSHUM** : number of pp levels for specific humidity.
  - **NLEVRHUM** : number of pp levels for relative humidity.
  - **NLEVVVEL** : number of pp levels for vertical velocity.
  - **NLEVLWAT** : number of pp levels for liquid water.
  - **NLEVSNOW** : number of pp levels for solid water.
  - **NLEVCLFR** : number of pp levels for cloud fraction.
  - **NLEVSCVA** : number of pp levels for passive scalars.
  - **NLEVDIVE** : number of pp levels for divergence.
  - **NLEVVORT** : number of pp levels for relative vorticity.
  - **NLEVPSI** : number of pp levels for velocity potential.

- **NLEVKHI** : number of pp levels for stream function.
- **NLEVTETA** : number of pp levels for potential temperature.
- **NLEVTHPW** : number of pp levels for $\theta' w$.
- **NLEVWIND** : number of pp levels for wind velocity.
- **NLEVEPTH** : number of pp levels for equivalent potential temperature.
- **NLEVABSV** : number of pp levels for absolute vorticity.
- **NLEVSTDF** : number of pp levels for stretching deformation.
- **NLEVSHDF** : number of pp levels for shearing deformation.
- **NLEVPOTV** : number of pp levels for potential vorticity.
- **NLEVWEPV** : number of pp levels for wet potential vorticity.
- **NLEVPRES** : number of pp levels for pressure.
- **NLEVUA01** to **NLEVUA16** : number of pp levels for upper air fields nr 01 to 16.

- Arrays **NLVP...** (dimensioned with **NFPXLEV**): level pointers for different variables.
  - **NLVPGEOP** : level pointers for geopotential.
  - **NLVPTEMP** : level pointers for temperature.
  - **NLVPUMOM** : level pointers for *U*-wind component.
  - **NLVPVMOM** : level pointers for *V*-wind component.
  - **NLVPSHUM** : level pointers for specific humidity.
  - **NLVPRHUM** : level pointers for relative humidity.
  - **NLVPVVEL** : level pointers for vertical velocity.
  - **NLVPLWAT** : level pointers for liquid water.
  - **NLVPSNOW** : level pointers for solid water.
  - **NLVPCLFR** : level pointers for cloud fraction.
  - **NLVPSCVA** : level pointers for passive scalars.
  - **NLVPDIVE** : level pointers for divergence.
  - **NLVPVORT** : level pointers for relative vorticity.
  - **NLVPPSI** : level pointers for velocity potential.
  - **NLVPKHI** : level pointers for stream function.
  - **NLVPTETA** : level pointers for potential temperature.
  - **NLVPTHPW** : level pointers for $\theta' w$.
  - **NLVPWIND** : level pointers for wind velocity.
  - **NLVPEPTH** : level pointers for equivalent potential temperature.
  - **NLVPABSV** : level pointers for absolute vorticity.
  - **NLVPSTDF** : level pointers for stretching deformation.
  - **NLVPSHDF** : level pointers for shearing deformation.
  - **NLVPPOTV** : level pointers for potential vorticity.
  - **NLVPWEPV** : level pointers for wet potential vorticity.
  - **NLVPPRES** : level pointers for pressure.
  - **NLVPUA01** to **NLVPUA16** : level pointers for upper air fields nr 01 to 16.
- Arrays **NSDO...** (dimensioned with **NFPXLEV**): number of subdomains for each level of different variables.
  - **NSDOGEOP** : number of subdomains for each level of geopotential.
  - **NSDOTEMP** : number of subdomains for each level of temperature.
  - **NSDOUMOM** : number of subdomains for each level of *U*-wind component.
  - **NSDOVMOM** : number of subdomains for each level of *V*-wind component.
  - **NSDOSHUM** : number of subdomains for each level of specific humidity.
  - **NSDORHUM** : number of subdomains for each level of relative humidity.

- **NSDOVVEL** : number of subdomains for each level of vertical velocity.
- **NSDOLWAT** : number of subdomains for each level of liquid water.
- **NSDOSNOW** : number of subdomains for each level of solid water.
- **NSDOCLFR** : number of subdomains for each level of cloud fraction.
- **NSDOSCVA** : number of subdomains for each level of passive scalars.
- **NSDODIVE** : number of subdomains for each level of divergence.
- **NSDOVORT** : number of subdomains for each level of relative vorticity.
- **NSDOPSI** : number of subdomains for each level of velocity potential.
- **NSDOKHI** : number of subdomains for each level of stream function.
- **NSDOTETA** : number of subdomains for each level of potential temperature.
- **NSDOTHPW** : number of subdomains for each level of $\theta' w$.
- **NSDOWIND** : number of subdomains for each level of wind velocity.
- **NSDOEPTH** : number of subdomains for each level of equivalent potential temperature.
- **NSDOABSV** : number of subdomains for each level of absolute vorticity.
- **NSDOSTDF** : number of subdomains for each level of stretching deformation.
- **NSDOSHDF** : number of subdomains for each level of shearing deformation.
- **NSDOPOTV** : number of subdomains for each level of potential vorticity.
- **NSDOWEPV** : number of subdomains for each level of wet potential vorticity.
- **NSDOPRES** : number of subdomains for each level of pressure.
- **NSDOUA01** to **NSDOUA16** : number of subdomains for each level of upper air fields nr 01 to 16.

- Arrays **NSDP...** (dimensioned with **(NFPDOM,NFPXLEV)**): subdomains pointers for different variables.
  - **NSDPGEOP** : subdomains pointers for geopotential.
  - **NSDPTEMP** : subdomains pointers for temperature.
  - **NSDPUMOM** : subdomains pointers for U-wind component.
  - **NSDPVMOM** : subdomains pointers for V-wind component.
  - **NSDPSHUM** : subdomains pointers for specific humidity.
  - **NSDPRHUM** : subdomains pointers for relative humidity.
  - **NSDPVVEL** : subdomains pointers for vertical velocity.
  - **NSDPLWAT** : subdomains pointers for liquid water.
  - **NSDPSNOW** : subdomains pointers for solid water.
  - **NSDPCLFR** : subdomains pointers for cloud fraction.
  - **NSDPSCVA** : subdomains pointers for passive scalars.
  - **NSDPDIVE** : subdomains pointers for divergence.
  - **NSDPVORT** : subdomains pointers for relative vorticity.
  - **NSDPPSI** : subdomains pointers for velocity potential.
  - **NSDPKHI** : subdomains pointers for stream function.
  - **NSDPTETA** : subdomains pointers for potential temperature.
  - **NSDPTHPW** : subdomains pointers for $\theta' w$.
  - **NSDPWIND** : subdomains pointers for wind velocity.
  - **NSDPEPTH** : subdomains pointers for equivalent potential temperature.
  - **NSDPABSV** : subdomains pointers for absolute vorticity.
  - **NSDPSTDF** : subdomains pointers for stretching deformation.
  - **NSDPSHDF** : subdomains pointers for shearing deformation.
  - **NSDPPOTV** : subdomains pointers for potential vorticity.
  - **NSDPWEPV** : subdomains pointers for wet potential vorticity.

- **NSDPPRES** : subdomains pointers for pressure.
- **NSDPUA01** to **NSDPUA16** : subdomains pointers for upper air fields nr 01 to 16.

Control variables for post-processing of 2D variables.

- **LFIT2** : .T./.F. if 2D post-processed fields are fitted/not fitted.
- Logical variables **LFP(XXX)**: .T./.F. if post-processing/no post-processing (pp/no pp) of variable **(XXX)**
  - **LFPMSLP** : .T./.F. if pp/no pp of mean sea level pressure.
  - **LFPSP** : .T./.F. if pp/no pp of surface pressure.
  - **LFPORO** : .T./.F. if pp/no pp of orography in the model.
  - **LFPGM** : .T./.F. if pp/no pp of mapping factor (grid-point only).
  - **LFPFOL** : .T./.F. if pp/np pp of tropopause folding indicator (grid-point only).
  - **LFPSU1** : .T./.F. if pp/no pp of surface field nr 1.
  - **LFPSU2** : .T./.F. if pp/no pp of surface field nr 2.
  - **LFPSU3** : .T./.F. if pp/no pp of surface field nr 3.
  - **LFPSU4** : .T./.F. if pp/no pp of surface field nr 4.
  - **LFPSU5** : .T./.F. if pp/no pp of surface field nr 5.
- Scalars **NSDO...**: number of subdomains for different variables.
  - **NSDOMSLP** : number of subdomains for mean sea level pressure.
  - **NSDOSP** : number of subdomains for surface pressure.
  - **NSDOORO** : number of subdomains for orography in the model.
  - **NSDOGM** : number of subdomains for mapping factor.
  - **NSDOFOL** : number of subdomains for tropopause folding indicator.
  - **NSDOSU1** : number of subdomains for surface field nr 1.
  - **NSDOSU2** : number of subdomains for surface field nr 2.
  - **NSDOSU3** : number of subdomains for surface field nr 3.
  - **NSDOSU4** : number of subdomains for surface field nr 4.
  - **NSDOSU5** : number of subdomains for surface field nr 5.
- Arrays **NSDP...** (dimensioned with **NFPDOM**): subdomain pointers for different variables.
  - **NSDPMSLP** : subdomain pointers for mean sea level pressure.
  - **NSDPSP** : subdomain pointers for surface pressure.
  - **NSDPORO** : subdomain pointers for orography in the model.
  - **NSDPGM** : subdomain pointers for mapping factor.
  - **NSDPFOL** : subdomain pointers for tropopause folding indicator.
  - **NSDPSU1** : subdomain pointers for surface field nr 1.
  - **NSDPSU2** : subdomain pointers for surface field nr 2.
  - **NSDPSU3** : subdomain pointers for surface field nr 3.
  - **NSDPSU4** : subdomain pointers for surface field nr 4.
  - **NSDPSU5** : subdomain pointers for surface field nr 5.

2.12.32 YOMWFPB.

Contains buffer of working fields (indices, weights) for horizontal post-processing. Variables are initialised in setup routines **setup**/**SUWFPBUF** for buffer **WFPBUF** and **setup**/**SUFPSC2B** for other variables. No variable in namelist.

List of variables:

- **NLENWFPBL**: length of buffer (DM-local).

- **WFPBUF** : buffer containing working fields for horizontal interpolations (size of the box, indices, weights, land/sea mask) (DM-local).
- **NSTAWFPB** : start adresses for post-processed packets of points within buffer (DM-local).

2.12.33 YOMWFPDS.

Description of working fields for horizontal post-processing. Variables are initialised in setup routine **setup**/**SUWFPDS**. Variables are DM-global. No variable in namelist.

Field pointers:

- **MBOX** : for the size of the box.
- **MILA** : pointer for the array containing the nearest northern latitude (*lat* 1 of figure 5.2).
- **MILON** : pointer for the array containing the longitude index of the point *A*1 of figure 5.2 .
- **MILOS** : pointer for the array containing the longitude index of the point *A*2 of figure 5.2 .
- **MILONN**: pointer for the array containing the longitude index of the point *A*0 of figure 5.2 .
- **MILOSS**: pointer for the array containing the longitude index of the point *A*3 of figure 5.2 .
- **MWSTD** : for the first weight without land-sea mask.
- **MWLSM** : for the first weight with land-sea mask for scalars.

Other variables:

- **NWFPOS**: total number of fields in **WFPBUF**.

2.12.34 Additional namelists, containing local variables.

These namelists are **NAMFPPHY** (read in routine **setup**/**SUFPPHY**), **NAMFPDY2**, **NAMFPDYP**, **NAMFPDYH**, **NAMFPDYV**, **NAMFPDYT** and **NAMFPDYS** (read in routine **setup**/**SUFPDYN**). These namelists can be used to make an accurate list of post-processed fields.

In ordinary case, at each post-processing time step, all the fields that are written in the namelist **NAMFPC** are post-processed at all the levels and for all the output domains written in the namelist **NAMFPC**. However, it is possible to get, at each post-processing time step, exactly the fields you wish, and nothing more: in that case, you have to make other namelists file which will contain the selection of the fields you wish to get. First, you have to set in **NAMCT0** the variable **CNPPATH** as the directory where the selection files will be. Under this directory, the name of a selection file must be xxtDDDDHHMM, where DDDDHHMM specifies the date/time of the post-processing time step. Then a selection file must contain 7 new namelists (see above).

Namelist NAMFPPHY:

Namelist **NAMFPPHY** is for physical fields and fluxes; it contains the following DM-global parameters:

- **CLPHY**: names of selected physical fields; array of 16 characters.
- **CLDPHY**: names of selected subdomains, separated by ":", for each selected physical field; characters array (example: 'DOM1:DOM2:DOM3').
- **CLCFU**: names of selected cumulated fluxes fields; array of 16 characters.
- **CLDCFU**: names of selected subdomains, separated by ":", for each selected cumulated flux; characters array (example : 'DOM1:DOM2:DOM3').
- **CLXFU**: names of selected instantaneous fluxes fields; array of 16 characters.
- **CLDXFU**: names of selected subdomains, separated by ":", for each selected instantaneous flux; characters array (example : 'DOM1:DOM2:DOM3').

Namelist NAMFPDY2:

Namelist **NAMFPDY2** is for 2D dynamical fields; it contains the following DM-global parameters:

- **CL2DF**: names of selected fields; array of 16 characters.
- **CLD2DF**: names of selected subdomains, separated by ":", for each selected field; characters array (example : 'DOM1:DOM2:DOM3').

Namelists NAMFPDYP, NAMFPDYH, NAMFPDYV, NAMFPDYT and NAMFPDYS:

These namelists are for fields post-processed respectively on pressure levels, height levels, potential vorticity levels, isentropic levels and hybrid levels; they contain the same following DM-global parameters:

- **CL3DF**: names of selected fields; array of 16 characters.
- **IL3DF**: list of selected indexes of post-processing levels, for each selected field; integer array of 2 dimensions; first subscript is for selected post-processing level; second one is the local field index.
- **CLD3DF**: names of selected subdomains, separated by ":", for each selected level and each selected field; characters array of 2 dimensions; first subscript is the local index of the selected level; second one is the local field index (example : 'DOM1:DOM2:DOM3').

Note that all the fields or levels or domains written in a selection file should be included in the main namelist **NAMFPC**, otherwise the job will abort. Default value of namelist quantities is 0 for **IL3DF** and ' ' for character quantities.

2.12.35 Pointer variables to be known:

Pointer PTRFP4.

Contains pointers for FULL-POS on one-level-type dynamical fields. All variables are DM-global. Variables are initialised in setup routine **setup**/**SU4FPOS**.

Horizontally post-processed dynamical array:

- **NFLDFPD**: field pointer. Array of dimension **NFPDYNB**.
- **RLEVFPD**: level value. Array of dimension **NFPDYNB**.
- **NBITFPD**: number of bits when writing output file. Array of dimension **NFPDYNB**.
- **NDOMFPD**: number of subdomains for each field. Array of dimension **NFPDYNB**.
- **NINDFPD**: indexes of subdomains for each field. Array of dimension (**NFPDOM**,**NFPDYNB**).

Physics and fluxes:

- **NFLDFPP**: field pointer. Array of dimension **NFPPHYB**.
- **FPARITP**: parity array (+1 for scalar; -1 for vector). Array of dimension **NFPPHYB**.
- **NITMFPP**: land-sea mask array for interpolation (0 if without; 1 if with). Array of dimension **NFPPHYB**.
- **NBITFPP**: number of bits when writing output file. Array of dimension **NFPPHYB**.
- **NDOMFPP**: number of subdomains for each field. Array of dimension **NFPPHYB**.
- **NINDFPP**: indexes of subdomains for each field. Array of dimension (**NFPDOM**,**NFPPHYB**).

Spectral arrays:

- **NFLDAFP**: field pointers in **SPAFP**. Array of dimension **NFPSPA**.
- **NFLDDFP**: field pointers in **SPDFP**. Array of dimension **NFPSPD**.
- **RLEVAFP**: level values in **SPAFP**. Array of dimension **NFPSPA**.
- **RLEVDFP**: level values in **SPDFP**. Array of dimension **NFPSPD**.

Grid point array:

- **NFLDFPX**: field pointers in **GAUXBUF**. Array of dimension **NFPAUXB**.
- **RLEVFPX**: level values in **GAUXBUF**. Array of dimension **NFPAUXB**.
- **NBITFPX**: number of bit when writing out **GAUXBUF**. Array of dimension **NFPAUXB**.

Other variables:

- **NC3FP**: field pointers. Array of dimension **NFP3DF**.
- **X3FP** : level values per field. Array of dimension (**NFPXLEV**,**NFP3DF**).
- **NI3FP**: subdomain index for each level of each field. Array of dimension (**NFPDOM**,**NFPXLEV**,**NFP3DF**).

Part VI: TECHNICAL AND COMPUTATIONAL PROCEDURES

CHAPTER 3 Parallel Implementation

Table of contents

3.1 INTRODUCTION

In order to achieve efficient execution on computers with multiple processors and distributed memory, a message-passing programming model has been adopted. This enforces a local view of data for each processor and requires that data belonging to other processors be copied into the local memory before it can be read. This is accomplished using messages which are communicated by means of message-passing library routines such as MPI.

Given a strong desire to protect the scientific code from details of the parallel implementation, a *transposition strategy* is used to handle the distributed memory code. With this approach, the complete data required are redistributed at various stages of a time step so that the arithmetic computations between two consecutive transpositions can be performed without any interprocess communication. Such an approach is feasible because data dependencies in the spectral tranform method exist within only one coordinate direction, this direction being different for each algorithmic component. An overwhelming practical advantage of this technique is that the message-passing logic is localized in a few routines. It also turns out to be a very efficient method. The transpositions are executed prior to the appropriate algorithmic stage, so that the computational routines (which constitute the vast bulk of the IFS source code) need have no knowledge of this activity.

In each of the algorithmic stages the source code is similar on each processor but the data are distributed among the processors. The distribution of data among the processors is determined during the setup phase of the IFS and involves, in some cases, quite complex calculations to achieve load balancing of the work among the processors. Care has been taken to be able to re-use the algorithms from the original serial code so that reproducible results are retained. For example, it has been possible to keep much of the physics code untouched by allowing for a local or a global view, respectively, when it is appropriate. The call to the physics routines just handles a set (vector) of grid-point columns, in principle from anywhere on the globe. The setup phase makes sure that the proper Coriolis parameters, orography, etc. are correctly associated with each grid-point vector. The definitions of many of the variables used to describe the local data distributions will be described in Appendix A.

For the purpose of further description of the parallel strategy, the algorithmic steps are referred to in the following order:

1) grid-point computations
2) Fourier transforms
3) Legendre transforms
4) spectral computations

3.2 THE PARALLEL ENVIRONMENT

The forecast model (IFS configuration 1) is capable of executing in three modes, each suitable for a different machine environment:

1) serial, requiring no message passing whatsoever (LMESSP = FALSE)
2) distributed memory parallel, requiring the existence of a (suitable) message passing library (LMESSP = TRUE)
3) shared memory parallel, requiring a shared memory 'macrotasking' interface such as that provided by sgi/Cray on their parallel vector platforms (YMP, C90, J90) (LCRAYPVP = TRUE).

The parallel environment is initialized in SU0DMINIT by calling MPE_INIT.

Array NPRCIDS defines process identifiers in the range 1–NPROC for the complete parallel environment. MY-

PROC is a scalar which identifies the current processor (one element out of NPRCIDS).

3.3 DATA INDEPENDENCE

The achievable parallelism within the IFS differs for each algorithmic step and depends primarily on the data independence within that step. For practical and efficiency reasons, not all of the available parallelism is exploited, as discussed below.

3.3.1 Grid point computations

With the exception of the *semi-Lagrangian* advection and mixed fine/coarse grained radiation calculations, grid-point computations contain only vertical dependencies. Thus, all grid columns can be considered to be independent of each other, allowing an arbitrary distribution of columns to processors. There are 138346 columns in a T_L319L31 resolution model. An extreme decomposition of one grid point per processor would work but would suffer from fine-grain granularity (high overheads in calling all the physics subroutines compared to useful work) and from *dynamic imbalances.* Additionally, performance on a computer with vector architecture would suffer greatly due to short vector lengths.

The semi-Lagrangian advection requires access to data from columns in the upwind direction in order to compute departure points. This is handled by means of additional messages in which the data (potentially) required as input to this calculation are transferred prior to their need.

The coarse-grained radiation calculations are performed on a grid that is finer near the poles than towards the equator. Therefore, the calculations require a redistribution of columns to processors in order to achieve reasonable static load balancing.

3.3.2 Fourier transform.

For the fast Fourier transforms (FFT), the Fourier coefficients are fully determined for each field from the grid-point data on a latitude. The individual latitudes are all independent as are the vertical levels and the fields. In practice, independence across the fields is not exploited in the current code, so the quantity of exploitable parallelism is limited to (number of **latitudes x levels**) or 9920 for a T_L319L31 model. This approach allows an efficient serial FFT routine to be used and precludes fine-grain parallelism which is unavoidable in parallel FFT implementations. An additional performance constraint is imposed when running on a vector machine. Best performance is obtained by using a vectorized FFT whereby the vector dimension is across multiple transforms. In the IFS the vector dimension is over **(field types) * (levels)** so that any parallelism greater than the number of latitude rows (320) has to be gained at the expense of vector performance.

3.3.3 Legendre transform

In the Legendre transforms, the zonal wave numbers, *m*, are dependent in the north–south direction, but are independent of each other, of the vertical levels, and of the fields. As with FFTs, independence across fields is not exploited. Because of the triangular nature of the truncation, the work content of each transform is inversely proportional to the value of *m*, so to achieve a good load balance, the zonal waves are coupled together in pairs. This leaves parallelism of order (**m/2 x levels** = 4960). The reduction factor of 2 comes from pairing a long and short wave number in order to achieve good load balance. Again, in practice, vector performance issues reduce this somewhat. The transform is accomplished using a matrix product routine with the vector dimension maximized by combining all fields to obtain a vector length of **(field types) * (levels)**. Parallelism greater than 160 may be achieved at the expense of vector performance.

3.3.4 Spectral computations.

Computations in spectral space have different data dependencies in various phases. The horizontal diffusion calculations have no dependencies. The semi-implicit time-stepping scheme imposes a vertical coupling which means that data independence is restricted to wave space (*m*, *n*). However, if the stretched/rotated grid options are used, then coupling exists again in the *n* components and the *m* components. A similar *n* dependency exists for some semi-implicit schemes not used operationally at ECMWF. As a minimum, data independence is always available over the zonal waves *m* or total wave number *n*. For practical and communication efficiency reasons, horizontal computations are performed with levels and *m* waves distributed as in the Legendre transforms. If the levels are distributed among processors, a transposition is needed to bring all levels together for a subset of the *(m, n)* spectral coefficients.

3.4 DATA STRUCTURES

There are several data structures utilized within the various computational phases of the IFS. Most belong to either the grid-point, Fourier, or spectral domain. These are described in this section. Due to the transposition strategy combined with the spectral-transform method, very few global data structures are required. The large majority of data structures and variables define quantities for the subdomain belonging to the processor, i.e. the number of grid columns on the processor, the number of local grid-point latitudes, and the number of spectral waves this processor is responsible for.

3.4.1 Grid-point data

The fundamental data organization in grid space stores points so that, for a given field, neighbouring storage locations contain field values in a west–east direction. This storage order reflects the nature of the indexing in nearly all inner DO loops of grid-point computational routines. A consequence is that the natural DO loop length is limited to the number of points in each latitude circle. Since the 'reduced grid' (reference here) is in widespread use, this leads to a performance penalty on vector based architectures, because the polar rows of a reduced grid would generate very short vector computation (typically only 18 grid points).

3.4.2 Grid-point blocking.

A solution to this performance problem is to introduce a 'blocking' mechanism whereby latitude rows are packed into a 'super-row' buffer which is then treated computationally as a single entity. The size of the buffer can be defined at run time by a namelist variable NPROMA and provides a way to trade memory for performance. The basic grid-point data structure reflects this mechanism. We first describe the case LCRAYPVP = TRUE which was used in the original (still functioning) IFS version used on Cray C90 and other shared memory computers. Here it is required that complete latitudes plus three extra wrap-around points fit into the NPROMA buffer. So the minimum NPROMA size is three more than the number of points on a latitude near the equator. Two extra points are required at the end of each latitude because of the FFT routines, and one extra point at the start and two point at the end are required by the semi-Lagrangian routines to simplify the interpolation routines. The three extra points are kept for all grid-point and FFT arrays to simplify the code design.

An example of the contents of a block with 2 packed latitudes would be:

F[0, 1], F[1, 1].....F[NLOENG(1), 1], F[NLOENG(1)+1, 1], F[NLOENG(1)+2, 1],
F[0, 2], F[1, 2].....F[NLOENG(2), 2], F[NLOENG(2)+1, 2], F[NLOENG(2)+2, 2]

where NLOENG(*nlat*) is the number of points in the (reduced) grid for latitude *nlat*.

3.4.3 The NPROMA block

In general, the NPROMA block will contain an arbitrary number of complete latitudes for a given field, with some unused space at the end.

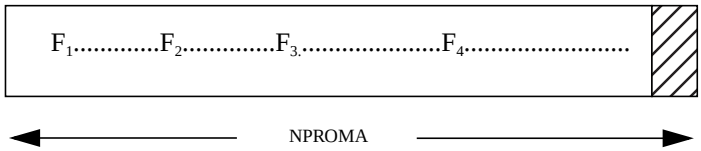


Figure 3.1 Diagram of the NPROMA block.

NPROMA is contained in namelist NAMDIM. There is a similar variable called NRPROMA which allows control over buffering within the radiation sparse grid.

In the case where LMESSP = TRUE the grid-point arrays do not have wrap-around points, i.e. they only contain the 'real' grid points on the globe. The distribution is also generalized so that a subset of latitudes can be kept in the NPROMA buffer. NPROMA can have any value right down to 1. On a vector machine one benefits from choosing as high a value as the memory constrains can afford. On a cache-based machine it is favourable to choose a small value so cache re-utilization is maximized. The extra two points required for FFT calculations are only present in local short-lived arrays used during the FFT. The three wrap-around points required for the semi-Lagrangian calculations are only present in the semi-Lagrangian buffer, and here in a generalized way (see Section 3.7 below) in order to cope with the more flexible grid-point distribution scheme utilized.

An NPROMA data block is followed in memory-storage order by a similar block from a different field or level. This is described in Fortran terms by the declaration: REAL F(NPROMA,NFIELDS,NGPBLKS) where NFIELDS is the total number of fields and levels, and NGPBLKS is the number of blocks necessary to hold a complete field. Within the IFS, the data is structured in this way after conversion to grid-point space by the FFT routine and reorganization by the transposition routine. Subsequently, the fields are passed to the computational routines block by block, as subroutine arguments. Examples are the local arrays ZGT0, ZGT1 and ZGT5 in SCAN2MDM.

3.4.4 Fourier data

In Fourier space, the data are organized for the convenience of the FFT routines. For LMESSP = FALSE each latitude has 'wrap-around' points, one located before the first point, and two located after the last point. If LMESSP = TRUE only two points at the end are present. Latitudes are located sequentially for a given field, in a buffer of length NPROMAG.

The data layout used for input and output to the FFT routines is shown in Fig. 3.2 . Note that since the values of N are not the same for each latitude of a reduced grid, there is wasted space at the end of the short rows. Additionally, when the data have been converted to Fourier waves there is again waste space, since there are always fewer waves than grid-points for each latitude. This last extra space is required during the FFT, but to save memory a packed version of this layout (GT0BUF) is used for long-term retention of data in Fourier space. In this case, each row holds only the wave information appropriate for that latitude, the wave-number cut-off is defined by the array NMEN.

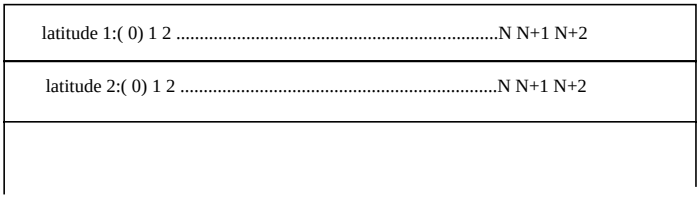


Figure 3.2 The data layout used for input and output to the FFT routines

The flow of data between these buffers is illustrated in Fig. 3.3 :

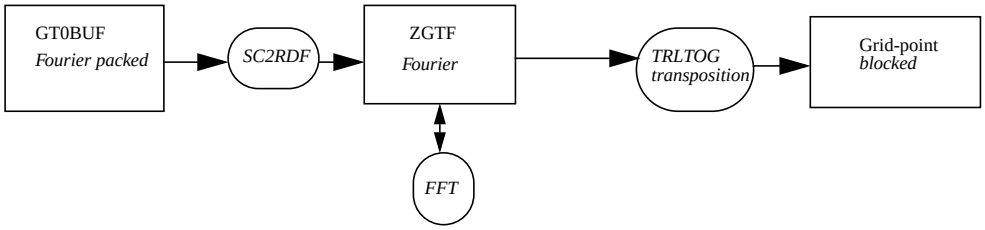


Figure 3.3 Data flow in the FFT routines.

3.5 DATA DISTRIBUTION

In order to ensure equal quantities of work for each processor flexibility in the data distribution algorithms is important. The goal is to achieve near perfect static work-load balance for any (reasonable) machine partition for any model resolution.

The following YOMCT0 module variables are used to define the data distribution for a parallel execution (most are also available as NAMELIST variables):

LMESSP = FALSE—for a serial execution or for a 'macrotasked' execution;
NPROC = n—total number of processors for the run (partition size);
LMESSP = TRUE, NPROC = 1—behaves identically to LMESSP = FALSE, except for increased memory usage and the execution of distributed memory specific code;
NPRGPNS—Number of PRocessors for Grid-Point computations in North–South direction;
NPRGPEW—Number of PRocessors for Grid-Point computations in East–West direction;
NPRTRW—Number of PRocessors for spectral TRansform computations in Wave number space,
NPRTRV—Number of PRocessors for spectral TRansform computations in vertical direction; and
LSPLIT—control various aspects of the data distribution.

Namelist defaults, the reading of NAMPAR0, and interdependencies are handled in subroutine SUMPINI. SUMP0 reads NAMPAR1 and initializes other relevant variables for the parallel execution.

3.5.1 Grid-point calculations

A number of options are provided, controlled by namelist variables NPRGPNS, NPRGPEW, and LSPLIT. The simplest distribution is achieved by setting:

NPRGPEW =1
NPRGPNS = NPROC
LSPLIT = FALSE.

This option subdivides complete lines of latitude and complete sets of levels amongst available processors and gives a good static load balance only for very specific values of NPROC for a particular model resolution. Its advantage lies in the fact that the data distribution is identical for the grid-point computations and for the Fourier transforms, thus eliminating the need for a transposition between these algorithmic stages. Use of the reduced grid with differing number of points for each latitude circle makes an even distribution of load particularly difficult.

Setting LSPLIT=TRUE allows a line of latitude to be split so that part of it is assigned to one processor and the remainder is assigned to the next processor. This provides for a near perfect static work load distribution for grid-point calculations, but the amount of parallelism remains limited by approximately 2/3 times the number of latitude rows owing to the FFT and Legendre Transforms. There are also efficiency disadvantages in the semi-Lagrangian message passing because relatively more data has to be communicated to and from neighbouring processors.

Another distribution type is achieved by setting:

NPRGPEW = n
NPRGPNS = m

where NPROC = n * m

It provides for considerable increased parallelism and allows the geographic area assigned to a processor to be more 'square-shaped', with consequent reduced communication volume in the semi-Lagrangian scheme. Data distribution is first carried out in the north–south direction over NPRGPNS processors. Each sub-area is then further split in the east–west direction over NPRGPEW processors (see Figs. A.3 and A.4).

Because the static distribution employed is fully determined by the input namelist variables, all required information about distributions and communication patterns can be calculated in the setup phase.

SUMP and its descendents SUMPLAT, SUSTAONL and SUSPLITLAT compute, among other things, the control structures which define the grid-point distributions. SUSTAONL is responsible for the grid-point distribution in the east–west direction.

Scalars define the processor location (MYSETA, MYSETB) within the logical 2-dimensional processor grid defined by NPRGPNS and NPRGPEW, where the A direction is north–south and the B direction is east–west (see Fig. A.1).

For each processor in the A direction, the NFRSTLAT and NLSTLAT arrays define the first and last latitude row numbers allocated to that processor, and LSPLITLAT is TRUE for those latitudes which are split over 2 processors. NPTRLAT points to the start of each latitude. (See Fig. A.2).

3.5.2 Fourier-space calculations

In order to retain good vector performance, it may be desirable to spread the latitudes over a different number of processors than that chosen for grid-point space. For this reason, separate variables NPRTRNS and NPRTRV control the data distribution, under the restriction that

NPROC = NPRTRNS x NPRTRV.

Achieving an even distribution of work north–south is a similar problem to that in grid-point space since, with the reduced grid option, the number of Fourier waves are reduced as the poles are approached. Fields and levels are combined together to create the vector dimension for the multiple FFT package (FFT992) used. Splitting this dimension across multiple processors is achieved by setting NPRTRV > 1. This will result in shorter vectors, but may

be required when many processors are used. For cache-based computers the performance will not be affected by this splitting. Note that, if NPRGPEW is not 1, a transposition has to be performed in order to gather together the distributed portions of each latitude row needed as input to each FFT. This is also the case if LSPLIT=TRUE. If NPRTRNS is not the same as NPRGPNS, the north–south distribution of rows or partial rows differs between grid-point computations and Fourier computations, resulting in a more complex transposition.

Subroutine SUMPLAT computes the Fourier-space data distribution. (See Fig. A.2).

Array NULTPP—defines the number of latitudes for each processor in the north–south direction.

Array NPTRLS—points to the first (global) latitude that each processor handles.

Array NPROCL—defines which processor computes the FFTs of that latitude.

Array NGPSET—contains the number of grid points which the processor owns, both in grid space and in Fourier space.

Array NPTRLL(1: NPRTRV) points to the first level and NUMLL contains the number of levels owned by each processor in the vertical decomposition.

3.6 DISTRIBUTED VECTORS

3.6.1 Introduction

Distributed vectors are widely used by subroutines which manipulate the analysis control vector, including much of the analysis and the Hessian singular vector code. They provide a convenient way to parallelize code which manipulates entire vectors rather than accessing individual elements. Their main advantage is that they hide message passing from the user.

The basis of the parallelization method is the introduction into the IFS of a fortran 90 derived type called a 'distributed_vector'. Variables of this type may be thought of as a pointers to one dimensional arrays whose contents are spread over the available processors.

Fortran 90 allows the programmer to define what is meant by assignment, the arithmetic operators, and generic functions, when they are applied to derived types. This feature has been used to allow distributed vectors to appear in expressions and as arguments to the dot product routine. For example, if d1 and d2 are distributed vectors of (say) 10000 elements each, and r1 is a one-dimensional array real array, also of 10000 elements, then the following are legal fortran 90 statements:

```
r1(:) = d1
d1 = (3.0*d1 -5.5) / DOT_PRODUCT(d1,d2)
```

What is not immediately obvious is that both statements perform message passing. The first statement copies all the elements of d1 into the array r1. Since many elements of d1 are stored on other processors, they must be sent to the local processor. Similarly, the generic DOT_PRODUCT subroutine has been extended to allow one or both arguments to be distributed vectors, in which case a parallel algorithm is used.

The second statement above illustrates another feature of distributed vector syntax. The arithmetic operations are written as if they apply to the whole vector. However, each processor will perform operations only on the locally held part of the vector, thereby distributing the work.

3.6.2 Using distributed vectors

The following statement declares d1 and d2 to be distributed vectors:

```
TYPE (distributed_vector) :: d1,d2
```

(Note that any of the usual qualifiers (INTENT, DIMENSION, ALLOCATABLE, etc.) may appear in the type statement.)

As with any pointer, we must allocate space for the data before we use a distributed vector, and remember to deallocate the space when we have finished with it. The following statements allocate and then deallocate a distributed vector of 10000 elements:

```
CALL allocate_vector (d1,10000)
CALL deallocate_vector (d1)
```

Once allocated, distributed vectors may appear in arithmetic expressions with other distributed vectors, with real scalars, or with one dimensional real arrays which are either of the same length as the full vector, or of the same length as the part of the vector which is held on the local processor. The result of such an expression is a real array of the size of the locally held part of the vector.

A distributed vector may appear on either side of an assignment statement. If it appears on the left hand side, then the right hand side must evaluate to a real scalar, to another distributed vector of the same length, to a one dimensional real array of the same length as the full vector, or to a one dimensional array of the same length as the part of the vector which is held locally. If the right hand side of an assignment statement is a distributed vector, then the left hand side must be either another distributed vector, or a real one dimensional array of the same length as the full vector.

A distributed vector may appear as one argument of the DOT_PRODUCT function. The other argument must be either another distributed vector of the same length, or a real array of the same length as the full vector. The functions SUM and MAXVAL may also be called with distributed vector arguments.

A sub-vector may be extracted from a distributed vector into a local array using the function dvsection:

```
r(i:j) = dvsection (d1,i,j)
```

The routines gather_vector and scatter_vector may be used to copy between distributed vectors and arrays which are defined on a single processor (for example the IO processor). For examples of the use of these subroutines, see GETMINI and SAVMINI.

3.6.3 Optimizing distributed vector code

The overloaded assignment and arithmetic operators are intended to make it easy to convert existing code, and to make the parallelized code easy to read. However, it should be noted that each assignment and each arithmetic operation results in a subroutine call. This is potentially expensive and inhibits some compiler optimizations. For this reason, an alternative syntax may be used to code frequently executed expressions involving distributed vectors. This syntax is less elegant, but is more efficient. It is illustrated by the following example.

Consider the following statement, where d1 and d2 are distributed vectors of 10000 elements each, and where r1 is a one dimensional real array, also of 10000 elements:

```
d1 = d2 + r1(:)
```

This statement results in two subroutine calls: one for the addition and one for the assignment. The following statement is equivalent, but does not result in subroutine calls:

```
d1%local(:) = d2%local(:) + r1(d1%local_start:d1%local_end)
```

The notation is explained as follows. Each distributed vector is a structure containing a pointer to the locally held part of the vector, and three integers holding the full length of the vector and the indices of the first and last elements of the locally held part. As illustrated in the example above, the components of a distributed vector are accessible using the "%" notation as (respectively) %local, %global_length, %local_start and %local_end.

3.6.4 Warnings

There is an implicit assumption that statements involving distributed vectors will be executed by all PEs. Be particularly careful in parallelizing IF statements. Care should also be taken in coding loops which use a mixture of distributed vectors and real arrays. It is easy to generate large amounts of message passing by frequent copying between arrays and distributed vectors.

Another potential danger is a DOT_PRODUCT statement whose arguments are both expressions. For example:

dot = DOT_PRODUCT (d1+5.0 , d2-2.0)

This will not work as expected. Both "d1+5.0" and "d2-2.0" will evaluate to real arrays of the length of the locally held part of d1 and d2. Each processor will apply the standard DOT_PRODUCT routine to its local part of "d1+5.0" and "d2-2.0" and each will calculate a different value for "dot".

This problem arises from the fact that the result of an expression involving a distributed vector is a real array rather than a distributed vector. Unfortunately, it is not possible to make such expressions return distributed vectors. The reason is rather subtle. Consider the following statement, where d1, d2 and d3 are distributed vectors:

d1 = (d2+d3)*d1 + (d2+d3)

The compiler will recognize that (d2+d3) is a common expression, and will generate code of the form:

temp = d2+d3
d1 = dtemp*d1 + temp

If arithmetic operations on distributed vectors returned distributed vectors, then the compiler-generated temporary vector temp would also be a distributed vector. Unfortunately, there is no way in fortran 90 to tell the compiler that allocate_vector and deallocate_vector must be called when temp is created and destroyed. (It would be possible to do this in an object-oriented language, such as C++.)

3.7 SEMI-LAGRANGIAN CALCULATION

3.7.1 Introduction

The semi-Lagrangian calculation in the IFS consists of two parts: the computation of a trajectory from a grid point backwards in time to determine the departure point; and the interpolation of various quantities at the departure point and at the mid point of the trajectory.

The issue for distributed memory computer systems is that both these parts require access to grid-point data held on neighbouring processors. For shared memory systems this issue can be resolved by processors simply accessing data in the shared memory while, for distributed-memory systems, message passing is required to obtain these data.

The grid-column data that could potentially be required on a processor from neighbouring processors (called the **halo**) is, in the present implementation, calculated assuming a conservative estimate of the global maximum wind likely to be encountered (VMAX2 (m sec^{-1})) and the model time step TSTEP (s). Typical values for VMAX2 are 150–200 m sec^{-1}. The advantage of using a fixed (large) VMAX2 is that semi-Lagrangian communication tables can be calculated once and for all during the setup phase after the distribution of grid columns to processors has been defined. This pre-fixed pattern then allows for efficient block transfers of halo data between processors. Debugging is also easier than in a dynamic scheme. The disadvantage is that a large amount of the halo data communicated may not really be required because the wind speed may be much lower than VMAX2. The wind may also blow from the core region towards the edges of this processors domain, i.e. the departure point is within the processor's core region. The semi-Lagrangian communication tables are calculated by the subroutine SLRSET called from SLCSET within SUSC2.

When LMESSP is FALSE, the distribution of grid-point space to processors is done in partitions of contiguous equally spaced latitudes, which allows the halo to be computed (in SUSC2) simply as a number of latitudes (NSLWIDE) north and/or south of each processor partition.

However, when LMESSP is TRUE, the distribution of grid-point space is more general, supporting both north–south and east–west partitioning—in general a processor can have any continuous block of grid columns on the sphere (see Figs. A.6 and A.7). As a result, a processor's halo cannot be expressed just by NSLWIDE. In this case SLCSET is called on each processor to calculate the halo of grid-point columns required by itself, based on (VMAX2, TSTEP) and on the additional stencil requirements of the semi-Lagrangian interpolation method. Once this is done, SLRSET is called to exchange this halo information with other processors so that each processor knows what data need to be sent and received and which processors to communicate with. All this is done only once at initialization time.

Then during each model time step the SL halo is constructed by calling SLCOMM to perform the message passing and SLEXTPOL to initialize those halo grid points that only require to be copied from other grid points held on the local processor. To simplify the interpolation routines, halo points are cyclically mirrored for complete latitudes in the east–west direction, and mirror-extended near the poles.

3.7.2 SLCSET

SLCSET is called mainly to determine the SL halo for the local processor. This halo is described by array variables NSLSTA, NSLONL and NSLOFF, which are dimensioned by the number of latitudes that cover the halo and core region (see Fig. A.7) and are, briefly,

NSLSTA(JN)—starting (most westerly) grid point (relative to Greenwich) for halo on relative latitude JN (is negative if the area starts west of Greenwich)

NSLONL(JN)—number of halo and core (i.e. belonging to this processor) grid points on relative latitude JN

NSLOFF(JN)—offset from beginning of SL buffer to first halo grid point on relative latitude JN

The semi-Lagrangian buffer SLBUF1 contains the variables needed for semi-Lagrangian interpolation. It has a 1-dimensional data structure representing the latitude fractions for each latitude within this processor's core plus halo region. The storage is organized from north towards south. The total size is calculated in SLCSET and called NASLB1. To improve vector efficiency and cache performance the 'horizontal' collapsed dimension is the inner-most loop in the semi-Lagrangian buffer. NASLB1 is just the container size and it may be increased slightly in SLCSET to avoid bank conflicts on vector machines. The second dimension represents the fields in the semi-Lagrangian buffer and will vary according to the chosen semi-Lagrangian configuration. This strategy makes it simple to add new fields to the semi-Lagrangian buffer—no changes in the message-passing routines are needed.

The calculation of the halo is done as follows:

For each latitude,

1) the minimum (i.e. most westerly) and maximum (i.e. most easterly) angles on the sphere are determined for the local processor's core region by considering NSLWIDE latitudes to the north and south.

2) the angular distance a particle can travel on the sphere (given the maximum wind speed VMAX2 and timestep TSTEP) is then subtracted and added respectively from the above minimum and maximum angles.

3) the angular distances are converted to grid points and at the same time a further grid point is added to satisfy the requirements of the interpolation method used. For more complex interpolation methods more points are required.

4) NSLSTA, NSLONL and NSLOFF are then updated for this latitude, such that the number of grid points required for the halo and core region is never greater than the number of grid points on the whole latitude plus the extra points (IPERIOD) required for the interpolation. In addition, the NSLWIDE latitudes at the north and south poles are forced to require full latitudes to simplify the design.

To aid debugging, space is also reserved on each latitude for NSLPAD grid points east and west of the halo. As these points are initialized to huge(), any attempt to use this data in an interpolation routine will result in an immediate floating point exception, which can simplify the detection of programming errors in SL interpolation routines. NSLPAD is 0 by default.

Once the halo has been determined, SLCSET then initializes NSLCORE to contain the position of each core point in the SL buffer. This data structure is used during the semi-Lagrangian calculations every time step.

Finally NSLEXT is initialized, which is used to simplify a SL buffer (lat, lon) offset calculation in LASCAW. This reduces an 'IF TEST' (to account for phase change over poles) and a 'modulo function', to a simple array access. As a result LASCAW becomes more efficient and more maintainable.

3.7.3 SLRSET

SLRSET is called by SLCSET at initialization time to determine the detailed send- and receive-list information that will be used later by SLCOMM during model execution. This is achieved by a global communication where send and receive lists are exchanged in terms of global (LAT, LON) coordinates.

The data structures initialized by SLRSET are as follows (see Fig. A.7):

NSLPROCS is a scalar which defines the number of processors that the local processor has to communicate with during SL halo communication.

Array NSLCOMM contains the list of processors that the local processor has to communicate with, and is dimensioned 1: NSLPROCS.

Arrays NSENDNUM, NRECVNUM contain the number of send and receive (lat, lon) pairs that the local processor has to communicate. The difference between elements (N) and (N+1) contain the number of entries that apply to processor N.

Arrays NRLSTLAT, NRLSTLON describe the global latitude and longitude of the grid-point columns to be received during SL halo communication. Columns to be received from processor N start at entry NRECVNUM(N) in these arrays.

Arrays NSLSTLAT, NSLSTLON describe the global latitude and longitude of the grid-point columns to be sent during SL halo communication. Columns to be sent to processor N start at entry NSENDNUM(N) in these arrays.

3.7.4 SLCOMM

SLCOMM is called at each model time step to obtain grid-point halo data from neighbouring processors. As the data volume for these communications can be very large, a strategy is used to control the amount of memory needed for the message-passing mailbox. This is done by blocking the data to be sent and controlling how many such blocks can be queued in a processors mailbox. Control is achieved by recognising that processor pairs involved in SL communication send and receive a similar amounts of data, and by only sending the next block of data to a processor when a corresponding block has been received from that processor. With this approach we avoid waiting for messages from processors that are still computing and use a probe function to detect if a message has arrived before issuing a receive for it. In this protocol, it is possible for 2 blocks to be queued at a destination processor from another processor. At initialization time (SLRSET) we compute the maximum number of processors that any proc-

essor has to communicate with (NSLPROCSMX), which is simply the maximum of NSLPROCS. Given NCOMBFLEN is the maximum number of words that we want to use at any processors mailbox and NSLPROCSMX we can compute the maximum block size by NSLMPBUFSZ = NCOMBFLEN/(3*NSLPROCSMX–1). Note that the factor 2*NSLPROCSMX corresponds to the maximum number of blocks that can be queued at a processor's mailbox for SL communication. The extra factor NSLPROCSMX–1 is needed to take into account mailbox fragmentation.

3.8 PROGRAM FLOW WITH THE FOCUS ON DISTRIBUTED-MEMORY DETAILS

3.8.1 Overview

The state variable of the model consists of the spectral components of the pseudo-divergence, pseudo-vorticity and surface pressure (or its logarithm) which gives 2*NFLEV+1 horizontal fields. In addition, we have p thermodynamic variables and q dynamically-passive scalar variables. In the 1997 ECMWF operational model, p = 1 (R * virtual temperature). Specific humidity is usually kept in grid-point space, where also three prognostic cloud fields are used. No passive scalar variables are used in the model.

The size of the dynamical part of the state variable of the model is then:

(NSMAX+1)*(NSMAX+2)*((2+p+q)*NFLEV +1), where NSMAX is the degree of the truncation and NFLEV the number of levels.

3.8.2 Constraints

The IFS has been designed to the following constraints:

1) it supports a direct integration, and also an adjoint and tangent-linear integration with associated management of the trajectory—de facto there are three models in one;
2) it supports two- and three-time-level Lagrangian advection schemes and an Eulerian advection scheme;
3) it supports a variable-resolution (reduced) grid;
4) the post-processing is built in;
5) possibility of a restart which gives the same result;
6) a normal-mode initialization process is included;
7) the Legendre functions and the normal modes are recomputed for each experiment (essential since the geometry may vary);
8) coding conventions—DOCTOR norms are used;
9) message passing and macro-tasking with reproducibility;
10) flexible placement in memory during grid-point calculations and transforms, in order to avoid memory bank conflicts;
11) message passing and macro-tasking intended to be performed at high level;
12) language—fixed format FORTRAN 90;
13) unification of the vertical interpolation routines;
14) the collocation grid is independent of the truncation.

The tangent-linear model and its adjoint are provided by coding the tangent linear and the adjoint of each subroutine. This is not the most efficient solution in terms of CPU but it is much easier to debug since it can be done routine by routine and the maintenance will obviously be greatly simplified.

Implicit in the following is the fact that we have three grid-point arrays (t–dt and t+dt, with only the fields, and t

with the fields plus their horizontal derivatives). For the integration of the model alone, we need only one spectral array (for the state variable), even considering the post-processing time steps. However during NMI and the adjoint integration, we need a second one.

3.8.3 Computer code organization

Note: The physical package is taken in this part as a 'black box'. Its input are fields at time **t1** and **t2** and its output are fields at the same times **t1** and **t2**, only those at time **t2** being modified.

Here follows a description of the data and control flow. The displacement of the routine name and the number in square brackets indicate the nesting level within IFS.

- MAIN
  - cnt0
    - su0yoma
      - sumpini

Read namelist NAMPAR0.

Determine if distributed-memory version or shared-memory version is used (LMESSP). If LMESSP is TRUE NMESS is defined to be 1, otherwise it is set to 0. NMESS is used to dimension some arrays used only in the distributed-memory version.

Processors are logically divided in a 2D decomposition (**NPRGPNS*NPRGPEW** for grid-point calculations and **NPRTRW*NPRTRV** for Fourier and Legendre transforms). It is possible to supply only **NPROC = (NPRG–PNS*NPRGPEW** and **NPRTRW*NPRTRV)** and let the code decide how to make the 2D decomposition.

NPROCK is the number of processors to be used simultaneously to execute **NUMKF** Kalman filter forecasts.

Decide the amount of diagnostic output (NOUTPUT and LMPDIAG)

- su0dminit
  - mpe_init

Initialize Message Passing Interface (MPI),
Determine MYPROC, A-set (MYSETA) and B-set (MYSETB)

- sulun
  - sumpout

Based on NOUPUT determine which PE's are writing diagnostics, the other PE's do not write because LOUTPUT = FALSE, or output data is dumped to /dev/null

- suarg

Read command line arguments and GRIB headers

- suct0

Define variables fixed at level zero. If LMESSP = TRUE we disable Cray macro-tasking and the use of 'out-of-core' work files (LCRAYPVP = FALSE, LMLTSK = FALSE, LIOXXX = FALSE)

- sump0

Set defaults for NAMPAR1 variables. These control the layout of data distribution (LSPLIT, LAPPLE, NAPLAT), tuning communication for specific architectures (LSLSYNC, LBIDIR, NPHASE, NCOMBFLEN, NLAGA, NLAGB, NVALAG, NCOSTLAG, NLAGBDY), control of I/O (NINSTR1, NINSTR2, NSTRIN, NSTROUT, NFLDIN, NFLDOUT, LSPLITIN, LSPLITOUT), and for debugging semi-Lagrangian code (NSLPAD)

Read NAMPAR1 namelist and initialize timing arrays

- sutag

Set up separate communication tag ranges to be able to distinguish communication performed by different message passing routines

- sumpi

Set up MPI identifiers (MINTET, MREALT, MLOGIT, MCHART).

- sudim

Calculate NFLEVL, NPSURF(), NPSP, NFLEVMX and NUMLL(), based on NFLEVG and NPROCB. NPROMA is allowed to be any value greater than zero if LMESSP = TRUE

- suallo

Allocate arrays. At this point only distributed memory arrays based on NPROCA, NPROCB and NFLEVL can be allocated.

- su0yomb
- sugem1a

Defines data structures for lat-long geometry that are independent of distributed memory configuration. Mainly NLOENG(NDGSAG: NDGENG) and NMENG(NDGSAG: NDGENG).

- sump

Master initialization routine for message-passing related data structures. Data distributions in grid-point space, Fourier space and spectral space are defined.

- sualmp1

Allocate data structures to be defined in **sump.** The data distributions are not yet defined, so only NPROCA/B and NFLEVL can be used to determine dimensions.

- suwavedi

Determines partitioning of zonal wave numbers to PEs for all truncations involved (NSMAX, NCMAX, NTMAX and NXMAX). Derived quantities are also calculated here (NASM0(:), NSPOLEGL, NPROCM(:), NUMPP(:), NSPEC, NSPEC2, NSPEC2MX, NPOSSP and IMYMS(:))

Back in **sump** NUMP is defined. Each PE calculates Legendre polynomials for a subset of latitudes in the northern hemisphere. The latitude distribution is determined here.

- sumplat

Calculate latitude distribution in grid-point space and Fourier space. Define a number of related data structures.

- sustaonl

Define the grid-point columns that belong to this processor. This is most complex if NPROCB >1 where a splitting in east–west direction is applied. The splitting depends on the LAPPLE and NAPLAT values. The grid-point column distribution on all other processors are communicated via message passing to this PE. This information is stored in NSTA(: , :) and NONL(: , :).

- susplitlat

Defines variables that control the handling of the latitude split among A-set (LSNDS, LSNDN, LRCVS, LRCVN, NSNDS, NSNDN).

- sumpretr

Calculates MYSENDA(:) and MYRECVA(:) for recursive A-set transposition case. Values depend on NPHASE

- subidir

Calculates MYSENDA(:) and MYRECVA(:) for bi-directional communication case

- suallt

If it is a variational job allocate trajectory arrays, most dimensioned using NSPEC2.

- *c*nt3
- opdis

Generate an operator display. Only processor 1 writes status to the **ifs.stat** file

- sualspa

Allocate SPA3(: , : , :) in case NCONF=1

- reresf

Read restart files if they are available. Fields are read in on processor 1 and the proper parts are distributed to other processors. Both spectral fields, surface fields and upper-air grid-point fields are treated

- csta
- suinif

Read initial data files if restart files are not available

- *[8]* suspec

Initialize spectral fields

- *[9]* suspecb

Initialize 2D-model versions

- *[9]* suspecg

Initialize 3D-model versions. Spectral GRIB files are (usually) read on PE1 and decoded using the GRIBEX subroutine. The correct spectral subsets are distributed to the other processors using MPE_SEND and MPE_RECV

- *[10]* gribex

GRIB decoding

- *[10]* suspgpg

If required some spectral fields (e.g. humidity) are transformed from spectral to grid point space because they are represented in grid point space in the state vector

- *[9]* suorog

Initialize grid-point orography and derivatives of orography from spectral-input orography. This is optional because a grid-point orography might be available.

- *[8]* specrt

Calculate spectral coefficients of 'TV' (TV defined as RT/R_d) based on humidity and, if available, cloud liquid/ice properties. Calculation for distributed memory version is done in the dm-code version.

- *[9]* specrtdm

Perform transforms of field to grid point space where RT/R_d is calculated.

- *[11]* gprcp

Calculates explicit normal modes (SPHBUF(:), HOUBUF(:)) for the zonal wave numbers treated by this processor, and frequencies required for tidal-wave initialization (FREQ(: , :), FREQ2(: , :)) for all NXPECG zonal wave numbers.

- sumode3l

Calculate logical arrays defining Rossby and gravity modes LTYPGR(: , :), diabatic subset LTYPDB(: , :) , tidal wave subset LTYPTD (: , :). They are dimensioned using the global NXPECG.

Optional writing of normal modes on external files (which is not done if LMESS = TRUE).

- sumode3i

Calculates implicit normal modes (RPINBUF(:) with dimensioning based on the local NCPEC).

- surinc

Initialize incremental variational job variables. No dependencies on distributed memory.

- sulcz

Initialize Lanczos common variables. No dependencies on distributed memory.

- sujbcov

Setup of the background error constraint J_B. No dependencies on distributed memory.

- cnt1
- cnt2

- suallt

If it is a variational job allocate trajectory arrays, most dimensioned using NSPEC2.

- cnt3
- opdis

Generate an operator display. Only processor 1 writes status to the **ifs.stat** file

- sualspa

Allocate SPA3(: , : , :) in case NCONF=1.

- reresf

Read restart files if they are available. Fields are read in on processor 1 and the proper parts are distributed to other processors. Both spectral fields, surface fields and upper air grid point fields are treated.

- csta
- uinif

Read initial data files if restart files are not available.

- *[8]* suspec

Initialize spectral fields.

- *[9]* suspecb

Initialize 2D-model versions.

- *[9]* suspecg

Initialize 3D-model versions. Spectral GRIB files are (usually) read on PE1 and decoded using the GRIBEX subroutine. The correct spectral subsets are distributed to the other processors using MPE_SEND and MPE_RECV.

- *[10]* gribex
- *[10]* suspgpg

If required some spectral fields (e.g. humidity) are transformed from spectral to grid-point space because they are represented in grid-point space in the state vector.

- *[9]* suorog

Initialize grid-point orography and derivatives of orography from spectral-input orography. This is optional because a grid-point orography might be available.

- *[8]* specrt

Calculate spectral coefficients of 'TV' (TV defined as RT/R_d) based on humidity and, if available, cloud liquid/ice properties. Calculation for distributed memory version is done in the dm-code version.

- *[10]* specrtdm

Perform transforms of field to grid-point space where RT/R_d is calculated.

- *[11]* gprcp
- *[10]* gprcp

Calculate R based on humidity and, if available, cloud liquid/ice properties.

- spnorm

Calculate spectral norm diagnostics. Partial norm contributions are calculated on each PE and communicated to PE1 (using the subroutine commspnorm). On PE1 the global norms are calculated in a reproducible manner.

- cnmi

Normal-mode initialization.

- cmac
- fltmode

Projection onto subsets of normal modes.

- fltrg

Hough filter of increments

- edfi

Controls integration for digital filter

- dealnmi

Deallocates arrays used in normal-mode calculations

- cnt4

Integration at level 4

- stepo

Basic time-stepping flow control

- *[8]* ltinvh

Inverse Legendre transforms

- *[9]* ltinvm
- *[10]* ltinv
- *[11]* asre1

Recombine symmetric and antisymmetric parts

- *[12]* asre1b
- *[9]* trmtol

Transposition from wave to Fourier space

- *[8]* scan2h

Multitasking interface to SCAN2M

- *[9]* scan2m

Computations in grid-point space

- *[10]* ftinvh

Control routine for inverse FFTs

- *[11]* ftinv

Inverse Fourier-transform driver

- *[12]* fft99

Fourier transform

- *[11]* trltog

Transposition from Fourier to grid-point space

- *[10]* cpg

Control of grid-point computations

- *[11]* gpxyb

Computes auxiliary arrays

- *[11]* gprh

Computes E_S and RH from T and Q

- *[11]* lacdyn

Computation of t and t - dt at grid points

- *[11]* cpcuddh

DDH accumulation

- *[11]* cpdyddh

Calculation of variables and dynamical flux-tendency diagnostics for DDH

- *[11]* gprcp

Computes C_P, R and R/C_P from Q

- *[10]* slcomm

Semi-Lagrangian halo communications

- *[10]* raddrv

Control of radiation calculations

- *[10]* cpglag

Lagged grid-point computations

- *[11]* lapine

Interface for Semi-Lagrangian interpolations

- *[11]* callpar

Interface for ECMWF Physics package

- *[11]* postphy

Post Physics computations

- *[11]* gpendtr

Final memory transfers in cpglag

- *[11]* gppref

Compute full-level pressure

- *[11]* sc2rdg

Copy grid-point arrays from buffer

- *[10]* ftdirh

Control for Fourier transforms

- *[11]* trgtol

Transposition from grid-point to Fourier space

- *[8]* ltdirh

Control for direct Legendre transforms

- *[9]* trltom

Transposition from Fourier to spectral wave space

- *[9]* ltdirm
  - *[10]* ltdir

Legendre transforms

- *[8]* spch

Control spectral-space calculations

- *[10]* trmtos

Transposition from horizontal to vertical spectral coefficients

- *[10]* trstom

Transposition from vertical to horizontal spectral coefficients

- spnorm

Compute spectral norms

- comtim

Parallel-processing timing.

APPENDIX A DESCRIPTIONS OF DATA STRUCTURES

Extensive use is made in the code of a set of scalar integers held in module YOMMP. These are used to assist the description of the data partitioning across processors and are based on the strategy of first partitioning in the north–south direction (NPRGPNS) and then further subpartitioning east–west, according to the value of NPRGPEW. The subdivision in each dimension is called a SET, leading to:

MYSETA: subdivision of latitudes in N–S direction, where MYSETA = 1..NPRGPNS

MYSETB: subdivision of latitudes in E–W direction, where MYSETB = 1..NPRGPEW

Similarly, for the wave-space partitioning:

MYSETW: subdivision of meridional waves, where MYSETW = 1..NPRTRW

MYSETV: subdivision of vertical levels ,where MYSETV = 1..NPRTRV

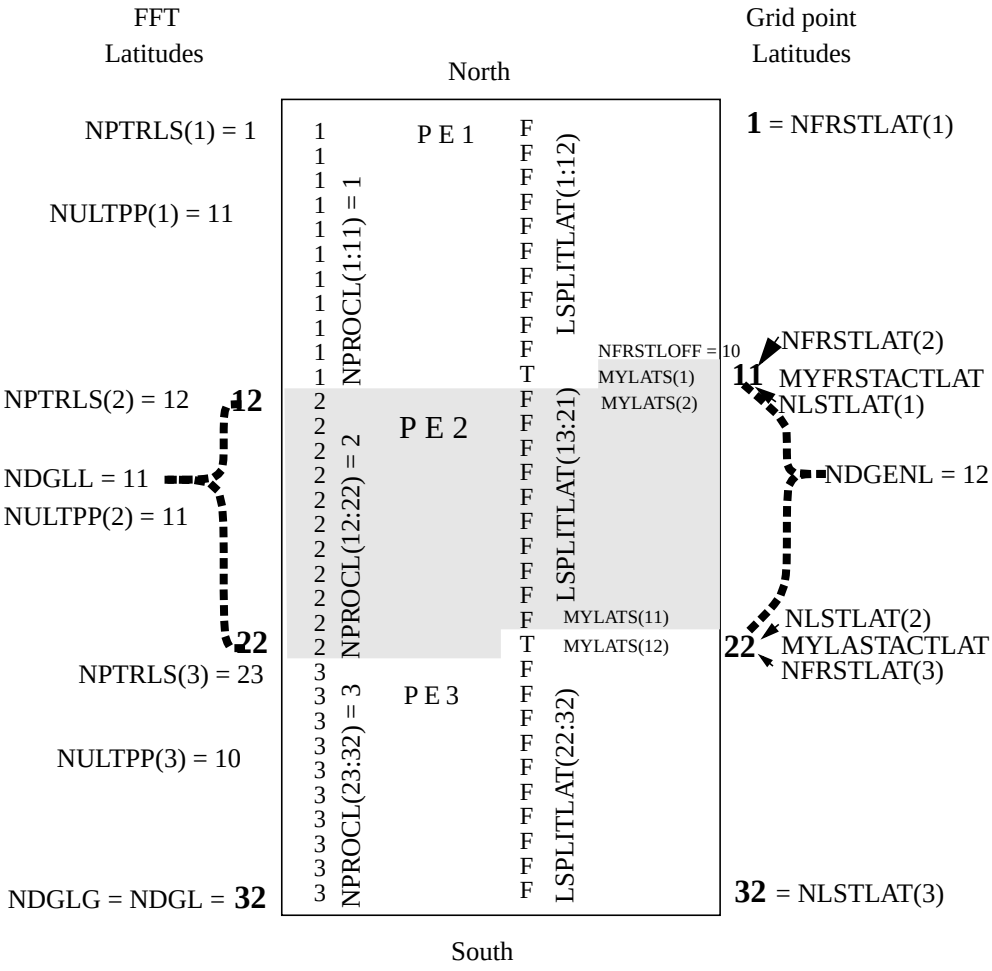


Figure A.1 : An example of a grid-point data distribution

The NPROC PEs are logically distributed in a 2-dimensional processor grid NPRGPNS * NPRGPEW. Due to the transposition method, most communication takes place along a column or a row in the logical grid. Each processor set of rows and columns can communicate independently of other sets.

Each processor has two logical set coordinates: MYSETA and MYSETB and a logical processor id MYPROC. A number of data structures describe which other PEs belong to the A-set and B-set of a PE. This is used to control communication in the A and B direction. Global communication is defined by the arrays

MYSENDG(1: NPROC-1) and
MYRECVG(1: NPROC-1)

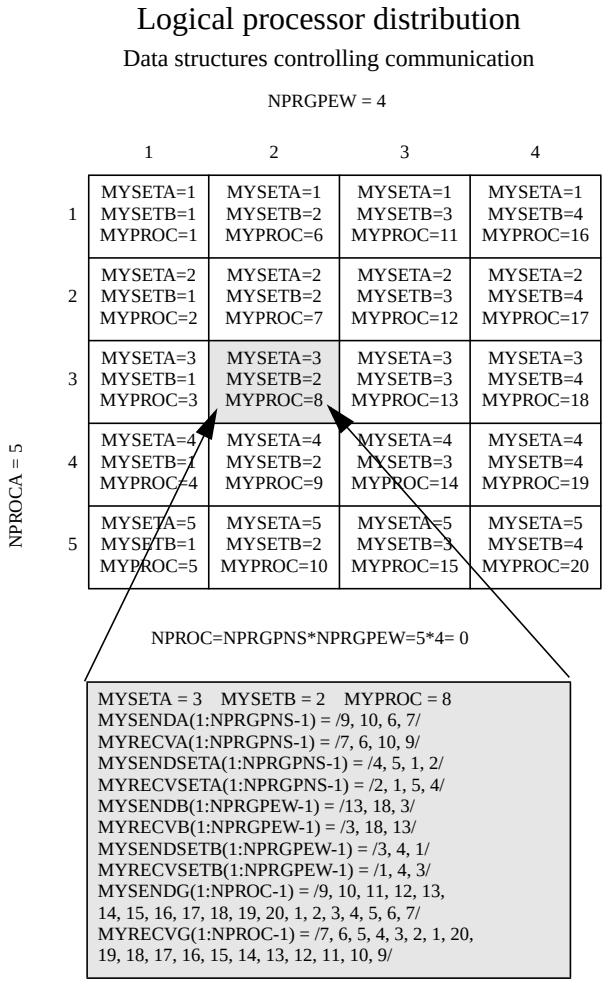


Figure A.2 : Distribution of grid-point and Fourier latitudes among processors

The example in Fig. A.2 shows NPRGPNS = NPRTRW = 3 , NPRGPEW = NPRTRV = 1, for processor 2 (PE 2). Most variables are uniquely associated with each PE, but some like NPROCL(:) are linked to the latitudes on the globe.

Grid-points and Fourier latitudes are disributed among PEs from north to south. The total number of (Gaussian) latitudes is called NDGLG. The array NPROCL((1: NDGLG) contains the (logical) number of the PE responsible for the Fourier transform calculations of each latitude, i.e. PE2 is responsible for latitudes 12 to 22. Since FFT calculations require all grid points on a latitude, complete latitudes are distributed among processors. The variable NDGLL is the number of Fourier latitudes on the PE (= 11). This information is also available for all NPRTRW processors in the array NULTPP(1: NPRTRW). Thus, NDGLL = NULTPP(2). An array NPTRLS(1: NPRTRW) contains integer pointers to the first latitude (in global numbering) within the Fourier latitude range of a PE. For PE2 the value is 12. It is assumed that a latitude range from North to South is associated with one PE. The NULTPP(:) and NPTRLS(:) information from other PEs are used in the transposition routines to and from Fourier space.

The right hand side of the diagram describes variables and data structures associated with grid point calculations. Due to the spectral transform method, there are only vertical dependencies for grid point calculations in IFS, with

the exception of the semi-Lagrangian calculations, described in section 3.7. This makes it possible simply to split latitudes in any convenient way to achieve load balance, i.e. equal amounts of grid point calculation cost on each PE. If the logical variable LSPLIT = TRUE , each PR is assigned the same number of grid points (+ or - 1). This achieves a fairly good load balance on partitions of up to several hundred PEs. The choice LSPLIT = FALSE is convenient for debugging purposes and for when (new) code portions have not yet been generalized to cope with split latitudes. The array LSPLITLAT(1: NDGLG) records if a latitude is split between 2 of the NPRGPNS PEs. The number of PEs used for the east-west grid splitting (NPRGPEW) does not influence any of the variables in Fig. A.2 .

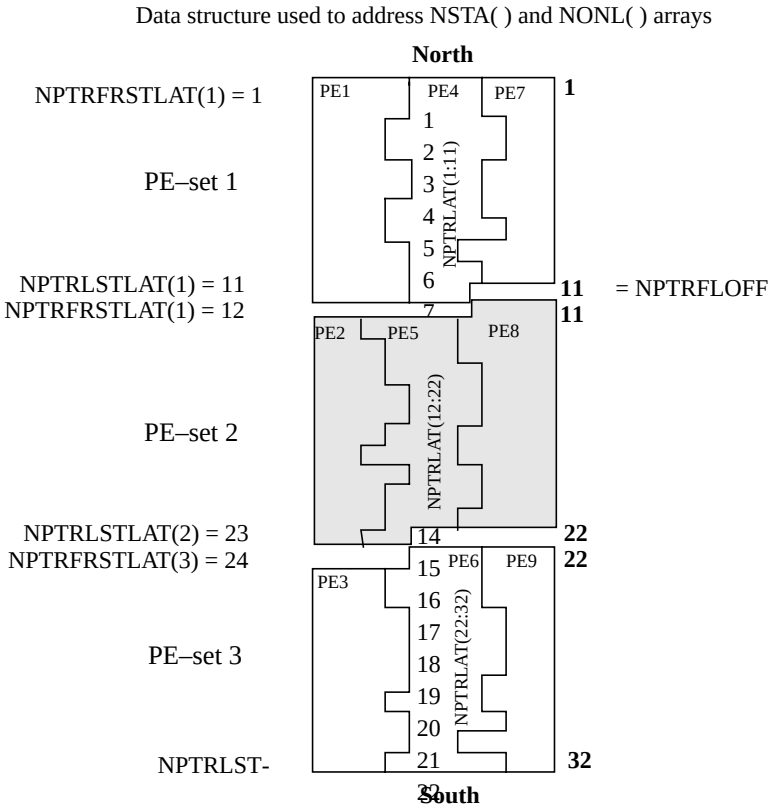


Figure A.3 : Discription of grid point distribution of data columns

PE2 has both the first and last latitude split. The number of latitudes from which PE2 has grid columns is NDGENL (= 12). The consecutive latitude range goes from MYFRSTACTLAT (= 11) to MYLSTACTLAT (= 22). This latitude range information for all NPRGPNS PEs is required for the transpositions to and from grid point space. It is defined in NFRSTLAT(1: NPRGPNS) and NLSTLAT(1: NPRGPNS) with respectively the first and last grid point latitude on each PE. As special cases, we have MYFRSTACTLAT = NFRSTLAT(2) and MYLSTACTLAT= NLSTLAT(2). The offset to the first grid point latitude of the PE is frequently used and stored in NFRSTLOFF (= MYFRSTACTLAT-1).

The array MYLATS(1: NDGENL) links the local grid point latitude numbering to the actual latitude number on the globe. In the example, MYLATS(1) = 11.

The arrays NSTA(NDGLG+NPRGPNS-1, NPRGPEW) and NONL(NDGLG+NPRGPNS-1, NPRGPEW) describe how grid columns are distributed among processors. A latitude is split into a number of consecutive strips,

the number depending on NPRGPEW and LSPLIT. Each latitude strip is uniquely determined by the starting point measured from Greenwich (NSTA) and the number of points on that latitude belonging to the processor (NONL).

Split latitudes complicate the picture because a physical latitude (latitude 11 in the example) is split among several PEs. So latitude 11 in the example has to be represented twice in the NSTA and NONL data structures. This explains the over-dimensioning of the two arrays. The variables in Fig. A.3 are used to control the proper handling of NSTA and NONL array references on different processor sets, as indicated in Fig. A.4 .

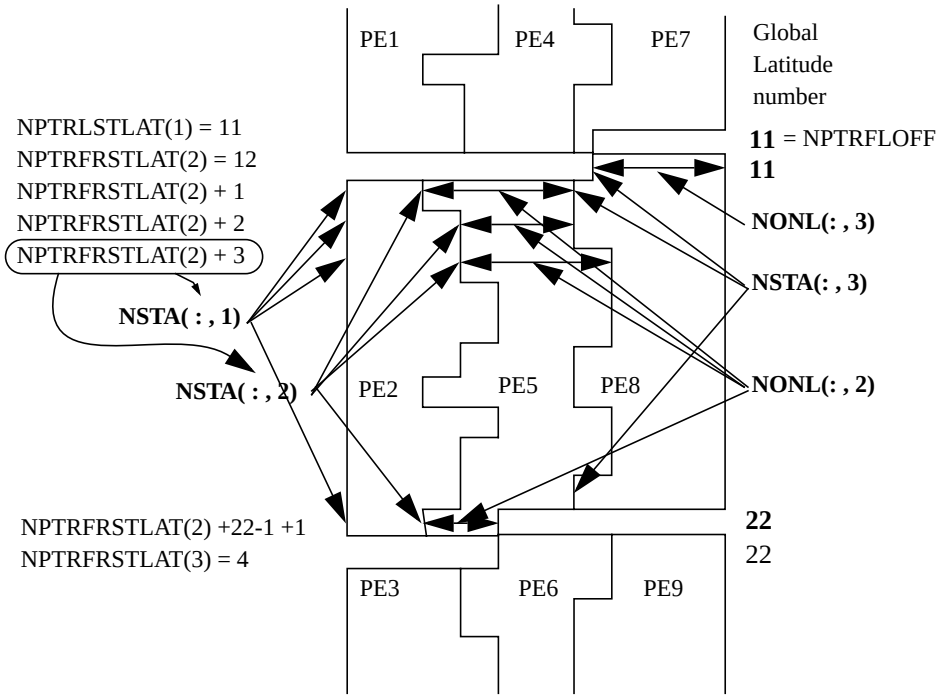


Figure A.4 : Layout of NSTA and NONL arrays

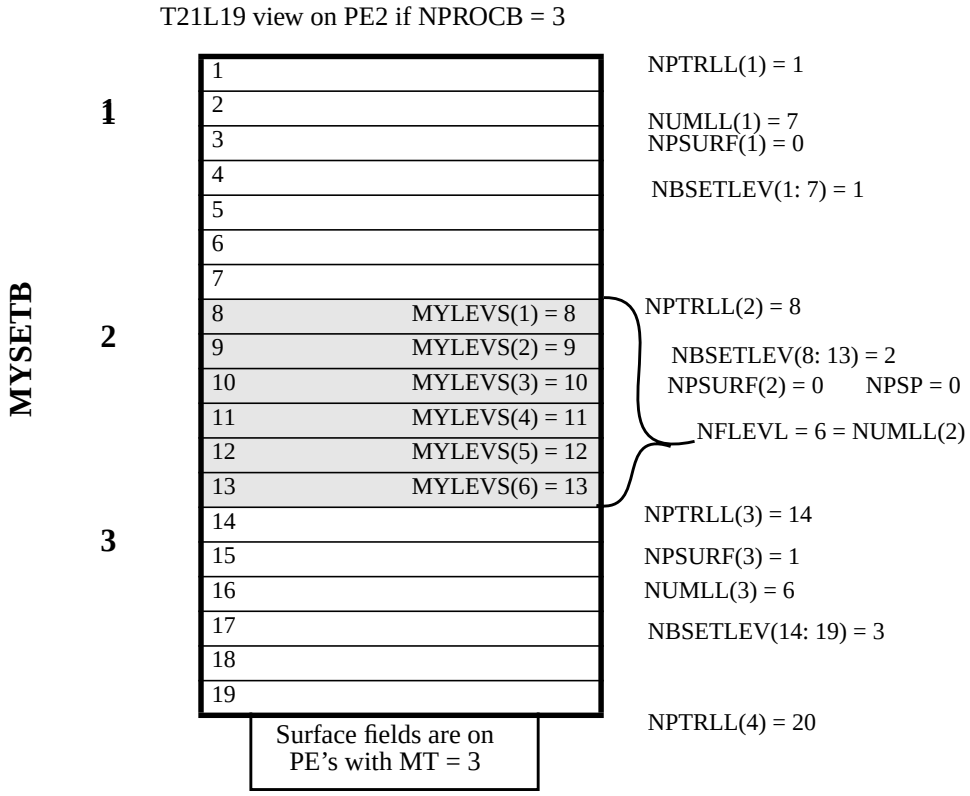


Figure A.5 : Parallelism by distributing vertical levels

When a high level of parallelism is required, it is necessary to distribute vertical levels among PEs for Fourier and Legendre transform calculations. NPRTRV is the number of PEs among which the vertical levels are distributed. In the diagram example, 19 levels are distributed among 3 PEs. The levels are always distributed as equally as possible in consecutive blocks. In Fig. A.5 , PE2 is responsible for the 6 levels from 8 to 13. The variables NFLEVL (=6) and MYLEVS(1: NFLEVL) hold this information. The array NUMLL(1: NPRTRV) contains the number of levels each of the NPRTRV PEs are responsible for. As a special case, NFLEVL = NUMLL(2).

The integer pointer array NPTRLL(1: NPRTRV) points to the actual first level on each processor, so NPTRLL(2)=8. To simplify code design, we define NPTRLL(NPRTRV+1) = NFLEVG + 1.

A help array NBSETLEV (1: NFLEVG) records which PE-set is responsible for the calculation of each vertical level, i.e. NBSETLEV(10) = 2. These help arrays are required to control the vertical transposition, and in cases where global vertical gathering of data is needed.

The surface pressure and other surface fields taking part in the spectral transform are typically assigned to the last PE in the NPRTRV set. The variable NPSP=1 on this set and 0 on all other sets. An additional help array NPSURF(1:NPRTRV) contains the NPSP value for each of the NPRTRV processor sets. It is necessary to have this information in order to perform vertical transpositions.

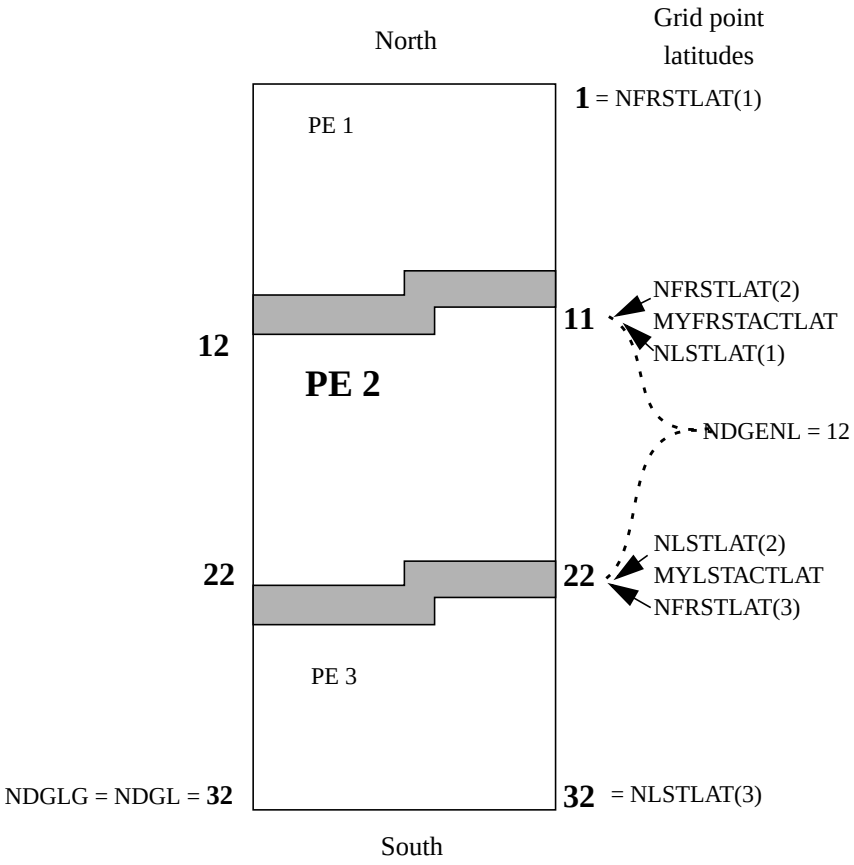


Figure A.6 : Semi-Lagrangian halos

The shaded areas in Fig. A.6 show the so-called halo regions for PE2 for a simple configuration using 3 PEs. The halo region contains the grid columns required by PE2 from neighbouring PEs in order to calculate semi-Lagrangian departure points for all grid columns owned by this PE.

Fig. A.7 shows the semi-Lagrangian core and halo regions in the most general configuration where the PEs (here PE11) have a 2-dimensional irregular shape with many neighbour PEs. For the data structures associated with halo communication, see the Subsections 3.7.2 SLCSET/Subsection 3.7.3 SLRSET description.

The spectral wave numbers are distributed in a round-robin fashion among the NPRTRW PEs so as to create a good load balance. In Fig. A.8 , using a T21 resolution, the waves are distributed among 3 PEs. The contents of the variables reflect the view seen from PE2. NUMP (= 7) is the number of spectral waves treated by PE2. The array MYMS(1: NPRTRW) contains the list of the actual wave numbers on PE2. The transposition routines use the array NUMPP(1: NPRTRW) containing the number of wave numbers on each of the NPRTRW PEs.

The number of real spectral coefficients on the PE is NSPEC (= 68) and, likewise, the number of complex spectral coefficients NSPEC2 (= 168). The NSPEC values will usually vary among PEs. The maximum number of spectral coefficients over the NPRTRW PEs is called NSPEC2MX (= 170) and is required for the dimensioning of arrays.

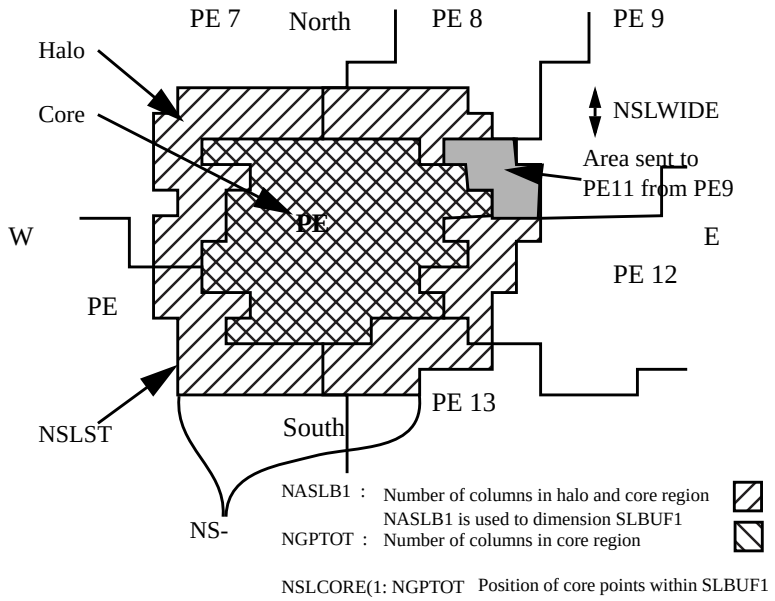


Figure A.7 : Semi-Lagrangian core and halo regions: some of the data structures involved.

NPROCM(0: NSMAX) defines which PE set is responsible for the calculation of each spectral wave. The NSPEC2 spectral coefficients are stored as a 1-dimensional data structure. NASM0(0: NSMAX) has the starting position of the NUMP waves on this PE. Only NUMP values in the range 0: NSMAX are assigned positive meaningful values. NASM0(:) is widely used throughout the code when calculations for specific zonal wave numbers are required.

The semi-implicit spectral calculations have only vertical dependencies so spectral coefficient columns can be distributed without constraints amoung the NPRTRV PEs (see Fig. A.9). To achieve good load balance, zonal waves are usually cut in the middle (see wave number 4 above). For some configurations, there are dependencies among the total waves (n) within a zonal wave number and for these cases, splitting in the middle of a wavenumber is not possible. This restricts the load-balanced parallelism to one half of the spectral truncation.

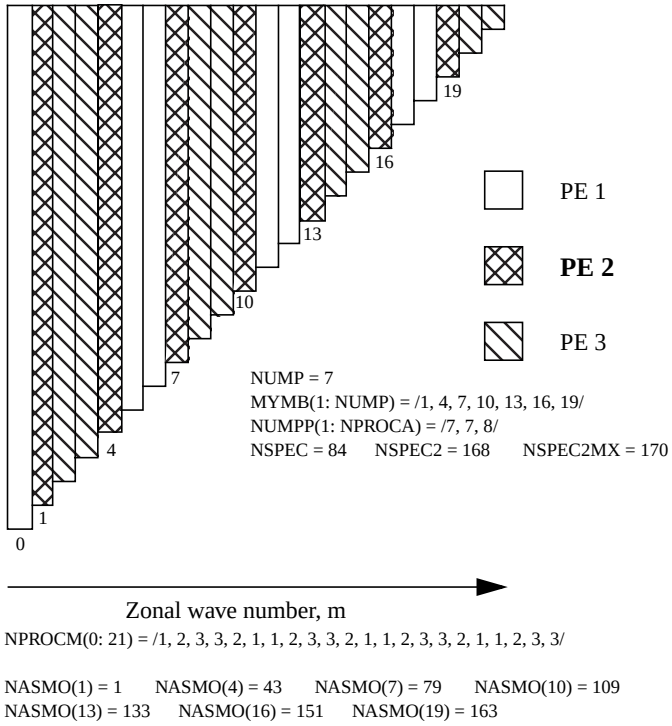


Figure A.8 : Distribution of zonal wave numbers. T21 spectral triangle: view on PE2 if NPROCA = 3

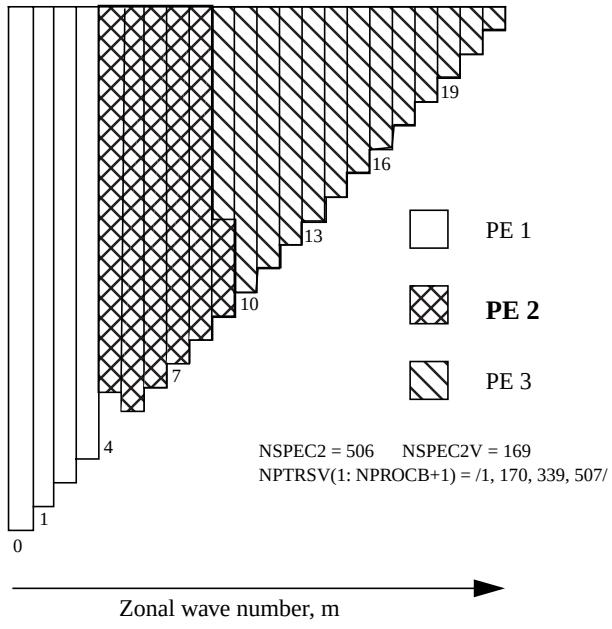
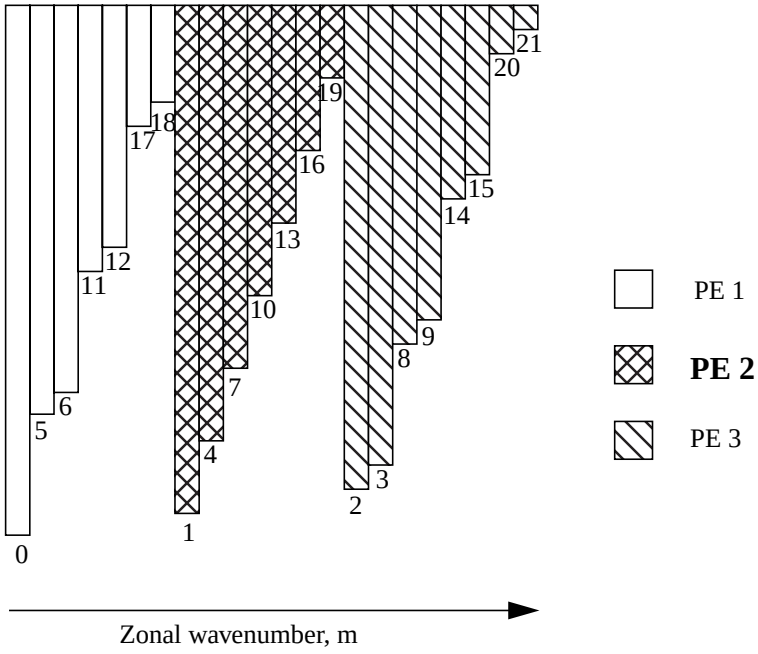


Figure A.9 T21 semi-implicit spectral calculations: view on PE2 if NPROCB = 3

NPTRMS(1: NPROCA) = /1, 8, 15/ (Points to the first wavenumber from each PE)



NALLMS(1: 22) = /0, 5, 6, 11, 12, 17, 18, 1, 4, 7, 10, 13, 16, 19, 2, 3, 8, 9, 14, 15, 20, 21/ (Global ordering of wavenumbers)

NPOSSP(1: NPROCA) = /1, 171, 339/ (Pointer to start position of data from each PE in global array)

NDIMOG(0: 21) = /1, 171, 339, 379, 213, 45, 79, 249, 417, 445, 279, 111, 133, 303, 471, 487/ (Start position of each wavenumber)

Figure A.10 Global representation of T21 spectral triangle (used for distribution of initial data and gathering cost function contributions).

In almost all parts of IFS, it is sufficient to have a subset of the spectral coefficients, namely the subset this PE is responsible for (see Fig. A.8). However, a global view is required when initial data is read, when post processed spectral fields are gathered, and when spectral cost function contributions are accumulated. The global spectral data structure (see Fig. A.10) is designed so that local parts from each PE (in processor order) within an A-set are stored next to each other. To avoid memory waste, the data are stored in a one-dimensional structure. An index to the spectral zonal waves in the global data structure is defined by an array NALLMS(1:NSMAX+1). The array ND1M0G(0:NSMAX) points to the first spectral coefficient for each spectral wave within the one-dimensional structure. Finally, the array NPOSSP(1: NPRTRW) records where the spectral coefficients from each A-set start within the global one-dimensional structure.

REFERENCES

Andersson, E. and Courtier, Ph., 1992: Post-processing changes for IFS cycle 9. (Research Department memorandum, internal note).

El Khatib, R., 1997: FULL-POS users' guide in ARPEGE/IFS cycle 16T1_al07 and ALADIN cycle AL07 with ARPEGE/IFS cycle 16T1_al07 (internal note).

Estrade J.-F., 1997: Développement ARPEGE/ALADIN. Mémoire distribuée/ mémoire partagée (internal note in French).

Rochas, M. and Courtier, Ph., 1992: La méthode spectrale en météorologie. Note de travail ARPEGE numéro **30**, 58pp.

Undén, P., 1995: IFS post-processing changes proposed by METEO-FRANCE and an improved formulation (internal note).

Yessad, K., 1997: Semi-Lagrangian computations in the cycles 17 and 18 of ARPEGE/IFS (internal note).

Yessad, K., 1997: Semi-implicit spectral computations in the cycles 17 and 18 of ARPEGE/IFS (internal note).

Yessad, K., 1997: Horizontal diffusion in the cycles 17 and 18 of ARPEGE/IFS (internal note).

Yessad, K., 1997: Spectral transforms in the cycles 17 and 18 of ARPEGE/IFS (internal note).

Yessad, K., 1997: Sphere to sphere transforms in spectral space in the cycles 17 and 18 of ARPEGE/IFS: configurations 911 and 912, TRAGEO. (internal note).