Regional Cooperation for
Limited Area Modelling in Central Europe

LACE
nwp central europe

Code Refactoring and cleaning for ALARO physics

Martina Tudor

ZAMG DHMZ Czech Hydrometeorological Institute OMSZ IMGW METEO ROMANIA SHMU ARSO METEO Slovenia

Overiview

Many reasons:
- new coding structures introduced by ECMWF,
- GPU adaptation,
- long subroutines hardly understandable

Work for ALARO to be based on the ARPEGE work of Philippe Marginaud

TO DO:
- have a separate aplpar for each physics package -> aplpar_alaro
- introduce new coding structures
- remove computations from mf_phys and aplpar_alaro
- adapt the code to go through automated GPU adaptation procedure

Code cleaning and APLPAR splitting

Physics calls are organized in MF_PHYS and APLPAR subroutines

- physics is one OpenMP loop
- too complex for for ttransformation to GPU code
- hard to use/modify/learn
- contains computations, allocations, variable definitions and many subroutine calls

MF_PHYS

– local computations moved to subroutines

APL_ARPEGE

- only calls to different subroutines
- all computations moved to subroutines
- uses encapsulated data (no modules)
- new data structures do not go below this level (EMCWF did differently)

New coding stuctures

- only one subroutine per file (acraneb2)

- only NPROMA arrays (no ZVARH(0:KLEV) or ZVARF(KLEV))
- no module variables (no more use YOMPHY, ONLY : ...)
  - moved them to a specific data structure
- no ALLOCATABLE arrays
  - everything is allocated at a single place before phy (memory)
- all output arguments should be NPROMA arrays
- new notations

APLPAR split done for ARPEGE

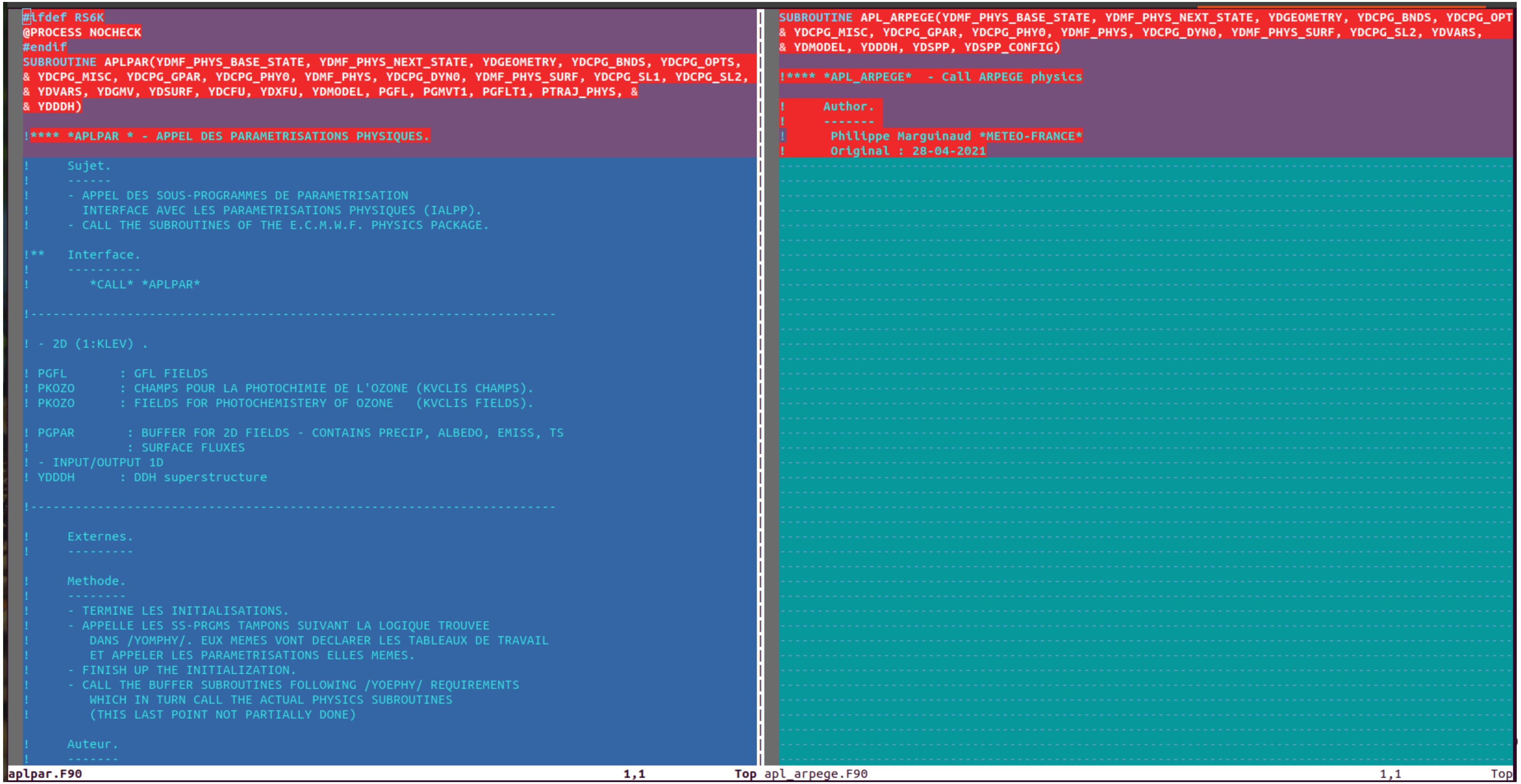
apl_arpege_aerosols_for_radiation.F90
apl_arpege_albedo_computation.F90
apl_arpege_atmosphere_update.F90
apl_arpege_cloudiness.F90
apl_arpege_deep_convection.F90
apl_arpege_dprecips.F90
apl_arpege.F90
apl_arpege_hydro_budget.F90
apl_arpege_init.F90
apl_arpege_init_surfex.F90
apl_arpege_oceanic_fluxes.F90
apl_arpege_precipitation.F90
apl_arpege_radiation.F90
apl_arpege_shallow_convection_and_turbulence.F90
apl_arpege_soil_hydro.F90
apl_arpege_surface.F90
apl_arpege_surface_update.F90

APL_ARPEGE calls

```
CALL CPPHINP
CALL MF_PHYS_FPL_PART1
  CALL MF_PHYS_SAVE_PHSURF_PART1
CALL APLPAR_INIT
  CALL CHECKMV
CALL APL_ARPEGE_INIT
CALL ACTQSAT
    CALL ACSOL               &
CALL APL_ARPEGE_INIT_SURFEX
  CALL ACHMTLS
  CALL ACHMT
CALL ACCLPH
CALL APL_ARPEGE_OCEANIC_FLUXES
CALL APL_WIND_GUST
CALL APL_ARPEGE_SHALLOW_CONVECTION_AND_TURBULENCE
CALL APL_ARPEGE_ALBEDO_COMPUTATION
CALL APL_ARPEGE_AEROSOLS_FOR_RADIATION
CALL APL_ARPEGE_CLOUDINESS
CALL APL_ARPEGE_RADIATION
CALL APL_ARPEGE_SOIL_HYDRO
CALL APL_ARPEGE_SURFACE
  CALL ACDNSHF          &
CALL ACDRAG
  CALL ACPLUIS (
```

```
CALL APL_ARPEGE_DEEP_CONVECTION
CALL APL_ARPEGE_PRECIPITATION       &
CALL QNGCOR
CALL APL_ARPEGE_HYDRO_BUDGET
CALL ACDRME
  CALL APLPAR_FLEXDIA
  CALL ACEVADCAPE
  CALL ACCLDIA
CALL ACVISIH
CALL PPWETPOINT                     &
CALL APL_ARPEGE_DPRECIPS
CALL MF_PHYS_MOCON
  CALL MF_PHYS_CORWAT
  CALL CPQSOL
CALL APL_ARPEGE_ATMOSPHERE_UPDATE
CALL MF_PHYS_FPL_PART2
CALL MF_PHYS_TRANSFER
CALL APL_ARPEGE_SURFACE_UPDATE
  CALL MF_PHYS_SAVE_PHSURF_PART2
CALL MF_PHYS_BAYRAD
CALL MF_PHYS_PRECIPS
```

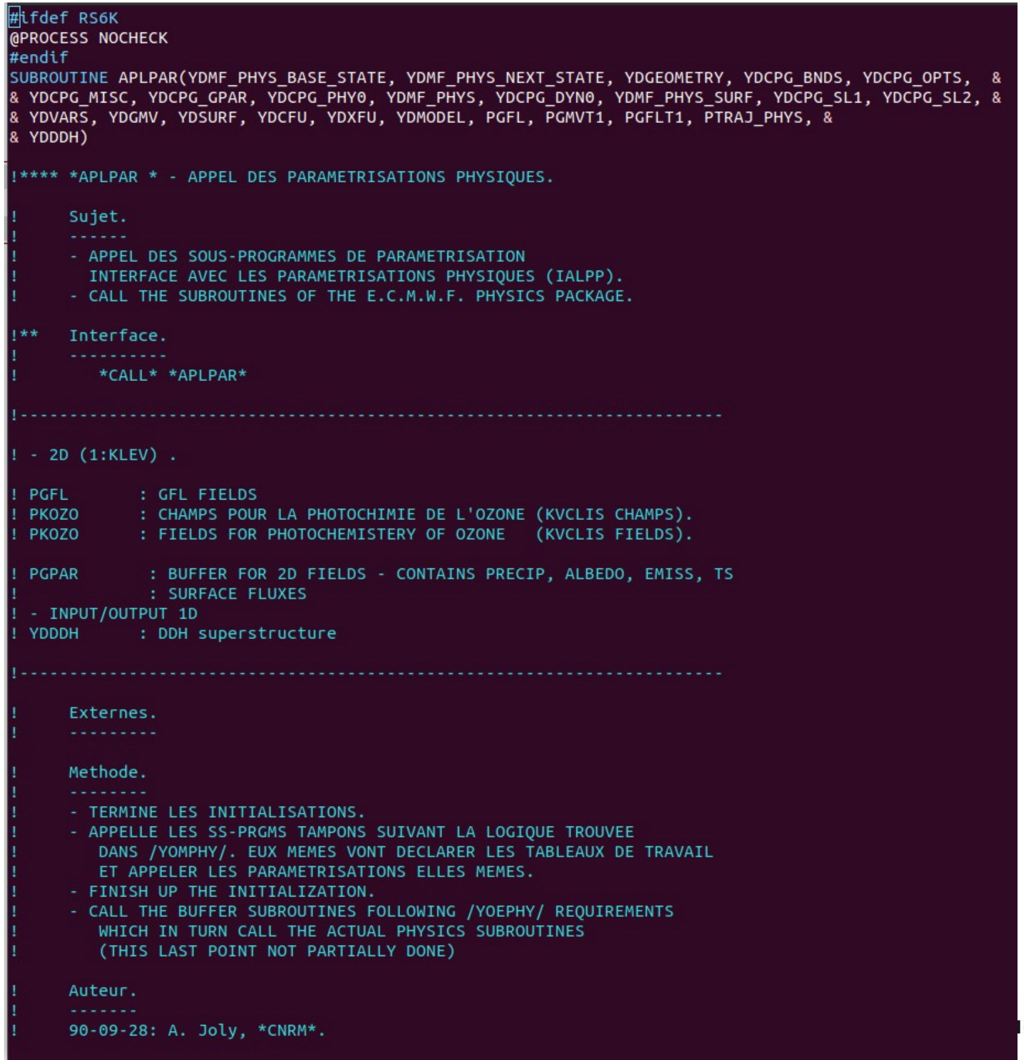
New version of APLPAR and APL_ARPEGE



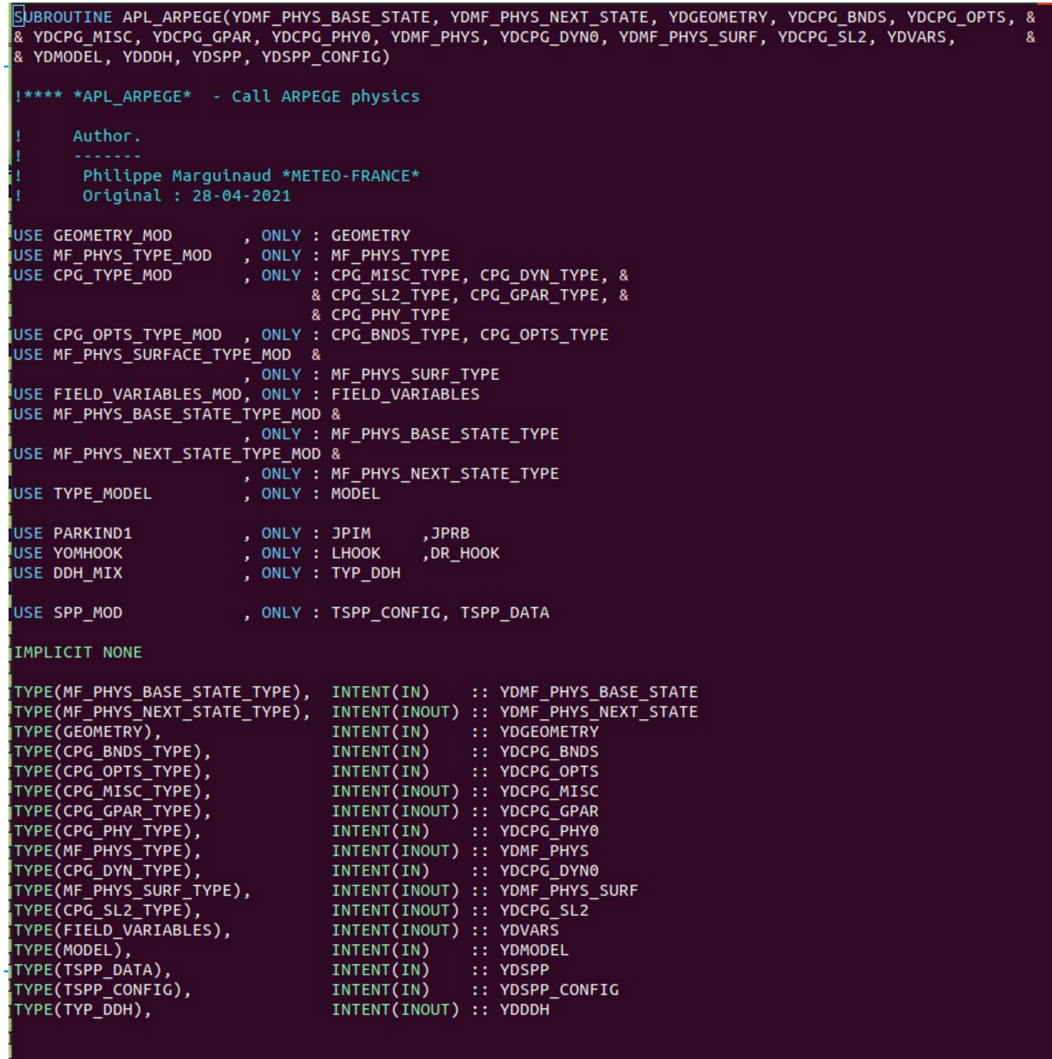
```
#ifdef RS6K
@PROCESS NOCHECK
#endif
SUBROUTINE APLPAR(YDMF_PHYS_BASE_STATE, YDMF_PHYS_NEXT_STATE, YDGEOMETRY, YDCPG_BNDS, YDCPG_OPTS,
& YDCPG_MISC, YDCPG_GPAR, YDCPG_PHY0, YDMF_PHYS, YDCPG_DYN0, YDMF_PHYS_SURF, YDCPG_SL1, YDCPG_SL2,
& YDVARS, YDGMV, YDSURF, YDCFU, YDXFU, YDMODEL, PGFL, PGMVT1, PGFLT1, PTRAJ_PHYS, &
& YDDDH)

!**** *APLPAR * - APPEL DES PARAMETRISATIONS PHYSIQUES.

!     Sujet.
!     ------
!     - APPEL DES SOUS-PROGRAMMES DE PARAMETRISATION
!       INTERFACE AVEC LES PARAMETRISATIONS PHYSIQUES (IALPP).
!     - CALL THE SUBROUTINES OF THE E.C.M.W.F. PHYSICS PACKAGE.

!**   Interface.
!     ----------
!        *CALL* *APLPAR*

!-----------------------------------------------------------------------

! - 2D (1:KLEV) .

! PGFL       : GFL FIELDS
! PKOZO      : CHAMPS POUR LA PHOTOCHIMIE DE L'OZONE (KVCLIS CHAMPS).
! PKOZO      : FIELDS FOR PHOTOCHEMISTERY OF OZONE   (KVCLIS FIELDS).

! PGPAR       : BUFFER FOR 2D FIELDS - CONTAINS PRECIP, ALBEDO, EMISS, TS
!             : SURFACE FLUXES
! - INPUT/OUTPUT 1D
! YDDDH      : DDH superstructure

!-----------------------------------------------------------------------

!     Externes.
!     ---------

!     Methode.
!     --------
!     - TERMINE LES INITIALISATIONS.
!     - APPELLE LES SS-PRGMS TAMPONS SUIVANT LA LOGIQUE TROUVEE
!        DANS /YOMPHY/. EUX MEMES VONT DECLARER LES TABLEAUX DE TRAVAIL
!        ET APPELER LES PARAMETRISATIONS ELLES MEMES.
!     - FINISH UP THE INITIALIZATION.
!     - CALL THE BUFFER SUBROUTINES FOLLOWING /YOEPHY/ REQUIREMENTS
!        WHICH IN TURN CALL THE ACTUAL PHYSICS SUBROUTINES
!        (THIS LAST POINT NOT PARTIALLY DONE)

!     Auteur.
!     -------
aplpar.F90                                                   1,1            Top
```

```
SUBROUTINE APL_ARPEGE(YDMF_PHYS_BASE_STATE, YDMF_PHYS_NEXT_STATE, YDGEOMETRY, YDCPG_BNDS, YDCPG_OPT
& YDCPG_MISC, YDCPG_GPAR, YDCPG_PHY0, YDMF_PHYS, YDCPG_DYN0, YDMF_PHYS_SURF, YDCPG_SL2, YDVARS,
& YDMODEL, YDDDH, YDSPP, YDSPP_CONFIG)

!**** *APL_ARPEGE*  - Call ARPEGE physics

!     Author.
!     -------
!      Philippe Marguinaud *METEO-FRANCE*
!      Original : 28-04-2021
apl_arpege.F90                                               1,1            Top
```

New version of APLPAR and APL_ARPEGE

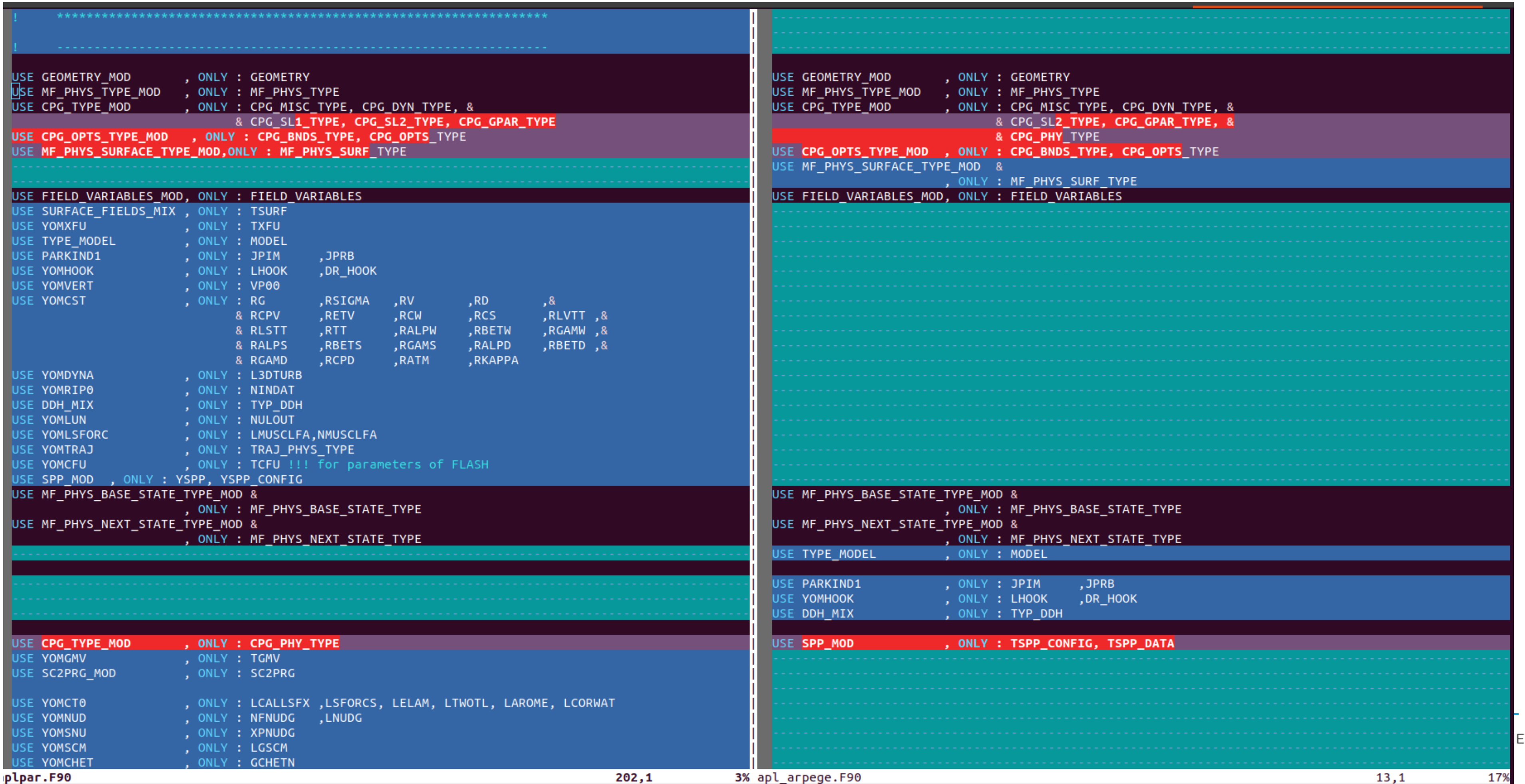


```
#ifdef RS6K
@PROCESS NOCHECK
#endif
SUBROUTINE APLPAR(YDMF_PHYS_BASE_STATE, YDMF_PHYS_NEXT_STATE, YDGEOMETRY, YDCPG_BNDS, YDCPG_OPTS,  &
& YDCPG_MISC, YDCPG_GPAR, YDCPG_PHY0, YDMF_PHYS, YDCPG_DYN0, YDMF_PHYS_SURF, YDCPG_SL1, YDCPG_SL2, &
& YDVARS, YDGMV, YDSURF, YDCFU, YDXFU, YDMODEL, PGFL, PGMVT1, PGFLT1, PTRAJ_PHYS, &
& YDDDH)

!**** *APLPAR * - APPEL DES PARAMETRISATIONS PHYSIQUES.

!     Sujet.
!     ------
!     - APPEL DES SOUS-PROGRAMMES DE PARAMETRISATION
!       INTERFACE AVEC LES PARAMETRISATIONS PHYSIQUES (IALPP).
!     - CALL THE SUBROUTINES OF THE E.C.M.W.F. PHYSICS PACKAGE.

!**   Interface.
!     ----------
!        *CALL* *APLPAR*

!-----------------------------------------------------------------------

! - 2D (1:KLEV) .

! PGFL       : GFL FIELDS
! PKOZO      : CHAMPS POUR LA PHOTOCHIMIE DE L'OZONE (KVCLIS CHAMPS).
! PKOZO      : FIELDS FOR PHOTOCHEMISTERY OF OZONE   (KVCLIS FIELDS).

! PGPAR       : BUFFER FOR 2D FIELDS - CONTAINS PRECIP, ALBEDO, EMISS, TS
!             : SURFACE FLUXES
! - INPUT/OUTPUT 1D
! YDDDH      : DDH superstructure

!-----------------------------------------------------------------------

!     Externes.
!     ---------

!     Methode.
!     --------
!     - TERMINE LES INITIALISATIONS.
!     - APPELLE LES SS-PRGMS TAMPONS SUIVANT LA LOGIQUE TROUVEE
!        DANS /YOMPHY/. EUX MEMES VONT DECLARER LES TABLEAUX DE TRAVAIL
!        ET APPELER LES PARAMETRISATIONS ELLES MEMES.
!     - FINISH UP THE INITIALIZATION.
!     - CALL THE BUFFER SUBROUTINES FOLLOWING /YOEPHY/ REQUIREMENTS
!        WHICH IN TURN CALL THE ACTUAL PHYSICS SUBROUTINES
!        (THIS LAST POINT NOT PARTIALLY DONE)

!     Auteur.
!     -------
!     90-09-28: A. Joly, *CNRM*.
```



```
SUBROUTINE APL_ARPEGE(YDMF_PHYS_BASE_STATE, YDMF_PHYS_NEXT_STATE, YDGEOMETRY, YDCPG_BNDS, YDCPG_OPTS, &
& YDCPG_MISC, YDCPG_GPAR, YDCPG_PHY0, YDMF_PHYS, YDCPG_DYN0, YDMF_PHYS_SURF, YDCPG_SL2, YDVARS,      &
& YDMODEL, YDDDH, YDSPP, YDSPP_CONFIG)

!**** *APL_ARPEGE*  - Call ARPEGE physics

!     Author.
!     -------
!      Philippe Marguinaud *METEO-FRANCE*
!      Original : 28-04-2021

USE GEOMETRY_MOD        , ONLY : GEOMETRY
USE MF_PHYS_TYPE_MOD    , ONLY : MF_PHYS_TYPE
USE CPG_TYPE_MOD        , ONLY : CPG_MISC_TYPE, CPG_DYN_TYPE, &
                               & CPG_SL2_TYPE, CPG_GPAR_TYPE, &
                               & CPG_PHY_TYPE
USE CPG_OPTS_TYPE_MOD   , ONLY : CPG_BNDS_TYPE, CPG_OPTS_TYPE
USE MF_PHYS_SURFACE_TYPE_MOD  &
                        , ONLY : MF_PHYS_SURF_TYPE
USE FIELD_VARIABLES_MOD, ONLY : FIELD_VARIABLES
USE MF_PHYS_BASE_STATE_TYPE_MOD &
                        , ONLY : MF_PHYS_BASE_STATE_TYPE
USE MF_PHYS_NEXT_STATE_TYPE_MOD &
                        , ONLY : MF_PHYS_NEXT_STATE_TYPE
USE TYPE_MODEL          , ONLY : MODEL

USE PARKIND1            , ONLY : JPIM     ,JPRB
USE YOMHOOK             , ONLY : LHOOK    ,DR_HOOK
USE DDH_MIX             , ONLY : TYP_DDH

USE SPP_MOD             , ONLY : TSPP_CONFIG, TSPP_DATA

IMPLICIT NONE

TYPE(MF_PHYS_BASE_STATE_TYPE),  INTENT(IN)    :: YDMF_PHYS_BASE_STATE
TYPE(MF_PHYS_NEXT_STATE_TYPE),  INTENT(INOUT) :: YDMF_PHYS_NEXT_STATE
TYPE(GEOMETRY),                 INTENT(IN)    :: YDGEOMETRY
TYPE(CPG_BNDS_TYPE),            INTENT(IN)    :: YDCPG_BNDS
TYPE(CPG_OPTS_TYPE),            INTENT(IN)    :: YDCPG_OPTS
TYPE(CPG_MISC_TYPE),            INTENT(INOUT) :: YDCPG_MISC
TYPE(CPG_GPAR_TYPE),            INTENT(INOUT) :: YDCPG_GPAR
TYPE(CPG_PHY_TYPE),             INTENT(IN)    :: YDCPG_PHY0
TYPE(MF_PHYS_TYPE),             INTENT(INOUT) :: YDMF_PHYS
TYPE(CPG_DYN_TYPE),             INTENT(IN)    :: YDCPG_DYN0
TYPE(MF_PHYS_SURF_TYPE),        INTENT(INOUT) :: YDMF_PHYS_SURF
TYPE(CPG_SL2_TYPE),             INTENT(INOUT) :: YDCPG_SL2
TYPE(FIELD_VARIABLES),          INTENT(INOUT) :: YDVARS
TYPE(MODEL),                    INTENT(IN)    :: YDMODEL
TYPE(TSPP_DATA),                INTENT(IN)    :: YDSPP
TYPE(TSPP_CONFIG),              INTENT(IN)    :: YDSPP_CONFIG
TYPE(TYP_DDH),                  INTENT(INOUT) :: YDDDH
```

New version of APLPAR and APL_ARPEGE



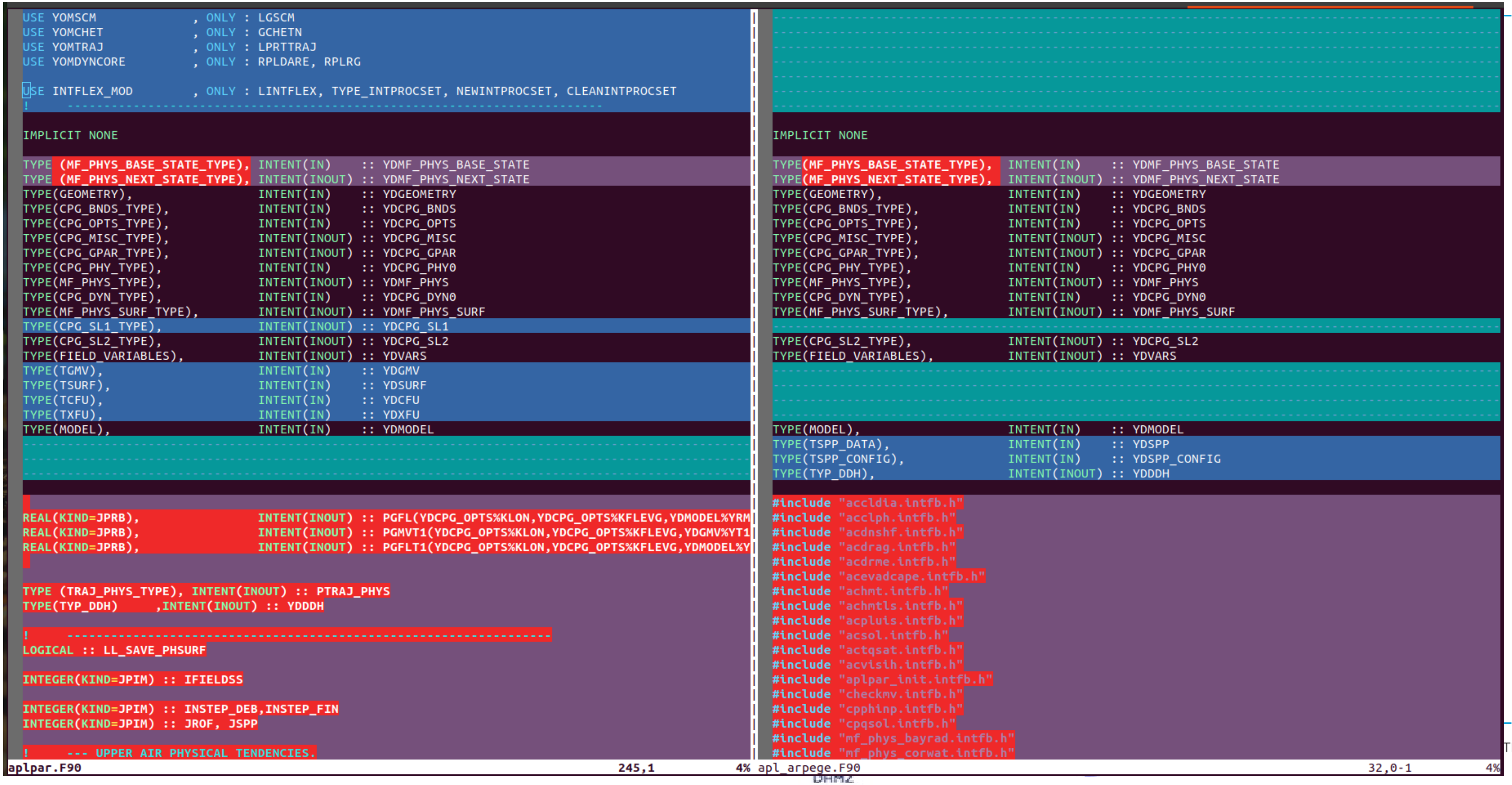
```
!      ******************************************************************

!      ------------------------------------------------------------------

USE GEOMETRY_MOD        , ONLY : GEOMETRY
USE MF_PHYS_TYPE_MOD    , ONLY : MF_PHYS_TYPE
USE CPG_TYPE_MOD        , ONLY : CPG_MISC_TYPE, CPG_DYN_TYPE, &
                               & CPG_SL1_TYPE, CPG_SL2_TYPE, CPG_GPAR_TYPE
USE CPG_OPTS_TYPE_MOD    , ONLY : CPG_BNDS_TYPE, CPG_OPTS_TYPE
USE MF_PHYS_SURFACE_TYPE_MOD,ONLY : MF_PHYS_SURF_TYPE
USE FIELD_VARIABLES_MOD, ONLY : FIELD_VARIABLES
USE SURFACE_FIELDS_MIX , ONLY : TSURF
USE YOMXFU              , ONLY : TXFU
USE TYPE_MODEL          , ONLY : MODEL
USE PARKIND1            , ONLY : JPIM     ,JPRB
USE YOMHOOK             , ONLY : LHOOK    ,DR_HOOK
USE YOMVERT             , ONLY : VP00
USE YOMCST              , ONLY : RG       ,RSIGMA   ,RV       ,RD       ,&
                               & RCPV     ,RETV     ,RCW      ,RCS      ,RLVTT ,&
                               & RLSTT    ,RTT      ,RALPW    ,RBETW    ,RGAMW ,&
                               & RALPS    ,RBETS    ,RGAMS    ,RALPD    ,RBETD ,&
                               & RGAMD    ,RCPD     ,RATM     ,RKAPPA
USE YOMDYNA             , ONLY : L3DTURB
USE YOMRIP0             , ONLY : NINDAT
USE DDH_MIX             , ONLY : TYP_DDH
USE YOMLUN              , ONLY : NULOUT
USE YOMLSFORC           , ONLY : LMUSCLFA,NMUSCLFA
USE YOMTRAJ             , ONLY : TRAJ_PHYS_TYPE
USE YOMCFU              , ONLY : TCFU !!! for parameters of FLASH
USE SPP_MOD   , ONLY : YSPP, YSPP_CONFIG
USE MF_PHYS_BASE_STATE_TYPE_MOD &
                        , ONLY : MF_PHYS_BASE_STATE_TYPE
USE MF_PHYS_NEXT_STATE_TYPE_MOD &
                        , ONLY : MF_PHYS_NEXT_STATE_TYPE

USE CPG_TYPE_MOD        , ONLY : CPG_PHY_TYPE
USE YOMGMV              , ONLY : TGMV
USE SC2PRG_MOD          , ONLY : SC2PRG

USE YOMCT0              , ONLY : LCALLSFX ,LSFORCS, LELAM, LTWOTL, LAROME, LCORWAT
USE YOMNUD              , ONLY : NFNUDG   ,LNUDG
USE YOMSNU              , ONLY : XPNUDG
USE YOMSCM              , ONLY : LGSCM
USE YOMCHET             , ONLY : GCHETN
```

```
plpar.F90                                                        202,1          3%
```

```
USE GEOMETRY_MOD        , ONLY : GEOMETRY
USE MF_PHYS_TYPE_MOD    , ONLY : MF_PHYS_TYPE
USE CPG_TYPE_MOD        , ONLY : CPG_MISC_TYPE, CPG_DYN_TYPE, &
                               & CPG_SL2_TYPE, CPG_GPAR_TYPE, &
                               & CPG_PHY_TYPE
USE CPG_OPTS_TYPE_MOD   , ONLY : CPG_BNDS_TYPE, CPG_OPTS_TYPE
USE MF_PHYS_SURFACE_TYPE_MOD  &
                        , ONLY : MF_PHYS_SURF_TYPE
USE FIELD_VARIABLES_MOD, ONLY : FIELD_VARIABLES
USE MF_PHYS_BASE_STATE_TYPE_MOD &
                        , ONLY : MF_PHYS_BASE_STATE_TYPE
USE MF_PHYS_NEXT_STATE_TYPE_MOD &
                        , ONLY : MF_PHYS_NEXT_STATE_TYPE
USE TYPE_MODEL          , ONLY : MODEL

USE PARKIND1            , ONLY : JPIM     ,JPRB
USE YOMHOOK             , ONLY : LHOOK    ,DR_HOOK
USE DDH_MIX             , ONLY : TYP_DDH

USE SPP_MOD             , ONLY : TSPP_CONFIG, TSPP_DATA
```

```
apl_arpege.F90                                                   13,1          17%
```

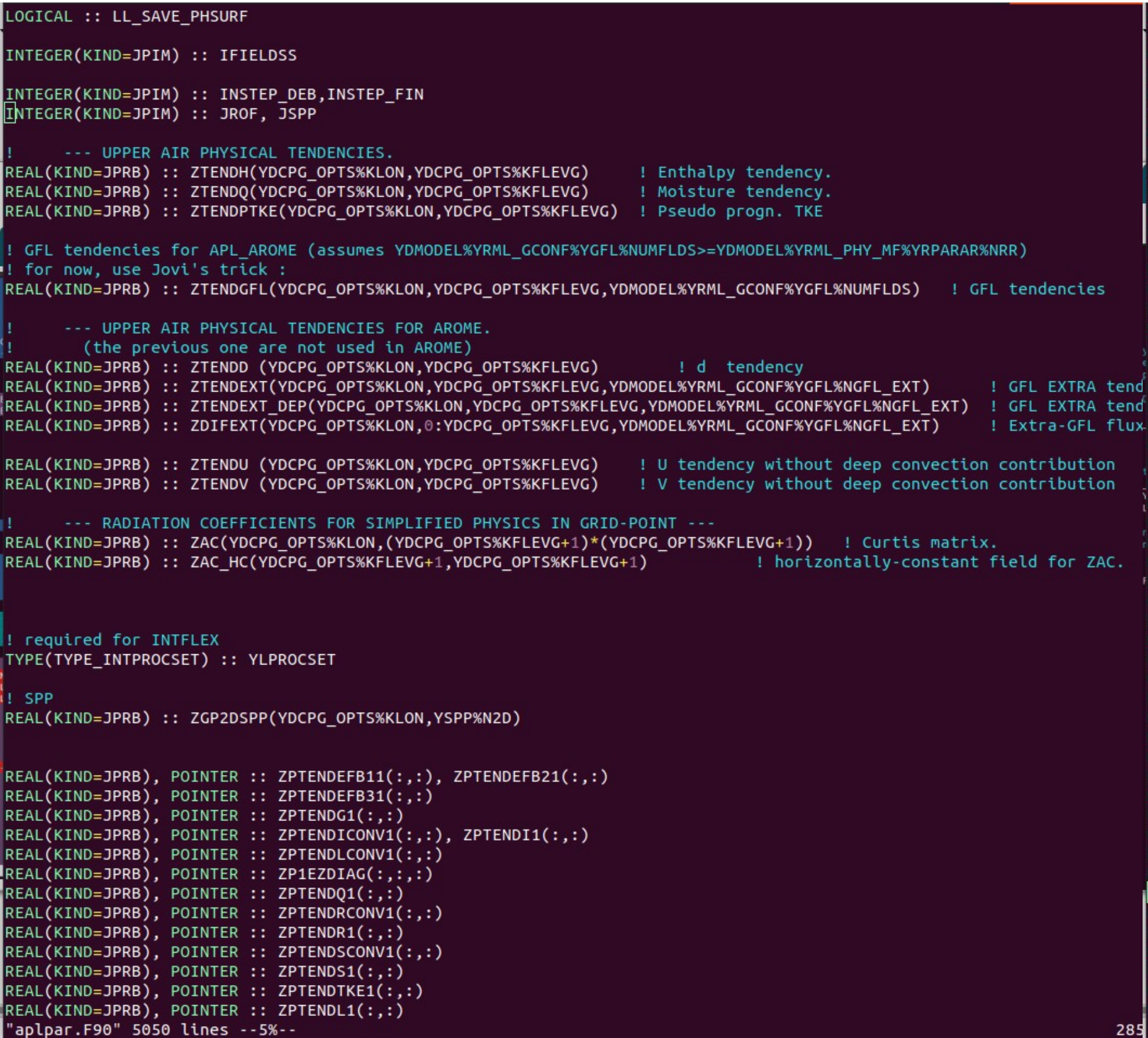
New version of APLPAR and APL_ARPEGE



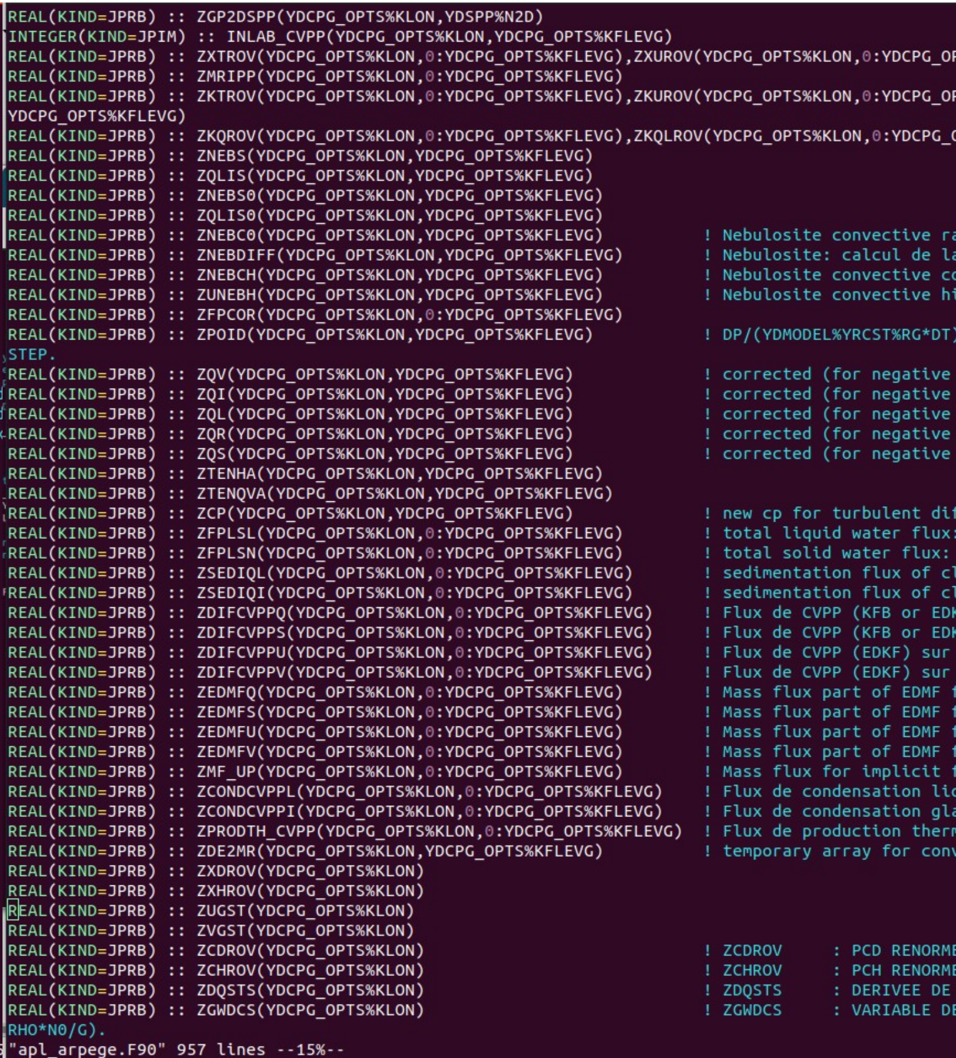
```
USE YOMSCM              , ONLY : LGSCM
USE YOMCHET             , ONLY : GCHETN
USE YOMTRAJ             , ONLY : LPRTTRAJ
USE YOMDYNCORE          , ONLY : RPLDARE, RPLRG

USE INTFLEX_MOD         , ONLY : LINTFLEX, TYPE_INTPROCSET, NEWINTPROCSET, CLEANINTPROCSET
!     -----------------------------------------------------------------------

IMPLICIT NONE

TYPE (MF_PHYS_BASE_STATE_TYPE), INTENT(IN)    :: YDMF_PHYS_BASE_STATE
TYPE (MF_PHYS_NEXT_STATE_TYPE), INTENT(INOUT) :: YDMF_PHYS_NEXT_STATE
TYPE(GEOMETRY),                 INTENT(IN)    :: YDGEOMETRY
TYPE(CPG_BNDS_TYPE),            INTENT(IN)    :: YDCPG_BNDS
TYPE(CPG_OPTS_TYPE),            INTENT(IN)    :: YDCPG_OPTS
TYPE(CPG_MISC_TYPE),            INTENT(INOUT) :: YDCPG_MISC
TYPE(CPG_GPAR_TYPE),            INTENT(INOUT) :: YDCPG_GPAR
TYPE(CPG_PHY_TYPE),             INTENT(IN)    :: YDCPG_PHY0
TYPE(MF_PHYS_TYPE),             INTENT(INOUT) :: YDMF_PHYS
TYPE(CPG_DYN_TYPE),             INTENT(IN)    :: YDCPG_DYN0
TYPE(MF_PHYS_SURF_TYPE),        INTENT(INOUT) :: YDMF_PHYS_SURF
TYPE(CPG_SL1_TYPE),             INTENT(INOUT) :: YDCPG_SL1
TYPE(CPG_SL2_TYPE),             INTENT(INOUT) :: YDCPG_SL2
TYPE(FIELD_VARIABLES),          INTENT(INOUT) :: YDVARS
TYPE(TGMV),                     INTENT(IN)    :: YDGMV
TYPE(TSURF),                    INTENT(IN)    :: YDSURF
TYPE(TCFU),                     INTENT(IN)    :: YDCFU
TYPE(TXFU),                     INTENT(IN)    :: YDXFU
TYPE(MODEL),                    INTENT(IN)    :: YDMODEL



REAL(KIND=JPRB),                INTENT(INOUT) :: PGFL(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG,YDMODEL%YRM
REAL(KIND=JPRB),                INTENT(INOUT) :: PGMVT1(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG,YDGMV%YT1
REAL(KIND=JPRB),                INTENT(INOUT) :: PGFLT1(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG,YDMODEL%Y


TYPE (TRAJ_PHYS_TYPE), INTENT(INOUT) :: PTRAJ_PHYS
TYPE(TYP_DDH)      ,INTENT(INOUT) :: YDDDH

!     -----------------------------------------------------------------------
LOGICAL :: LL_SAVE_PHSURF

INTEGER(KIND=JPIM) :: IFIELDSS

INTEGER(KIND=JPIM) :: INSTEP_DEB,INSTEP_FIN
INTEGER(KIND=JPIM) :: JROF, JSPP

!     --- UPPER AIR PHYSICAL TENDENCIES.
aplpar.F90                                                                  245,1          4%
```

```
IMPLICIT NONE

TYPE(MF_PHYS_BASE_STATE_TYPE),  INTENT(IN)    :: YDMF_PHYS_BASE_STATE
TYPE(MF_PHYS_NEXT_STATE_TYPE),  INTENT(INOUT) :: YDMF_PHYS_NEXT_STATE
TYPE(GEOMETRY),                 INTENT(IN)    :: YDGEOMETRY
TYPE(CPG_BNDS_TYPE),            INTENT(IN)    :: YDCPG_BNDS
TYPE(CPG_OPTS_TYPE),            INTENT(IN)    :: YDCPG_OPTS
TYPE(CPG_MISC_TYPE),            INTENT(INOUT) :: YDCPG_MISC
TYPE(CPG_GPAR_TYPE),            INTENT(INOUT) :: YDCPG_GPAR
TYPE(CPG_PHY_TYPE),             INTENT(IN)    :: YDCPG_PHY0
TYPE(MF_PHYS_TYPE),             INTENT(INOUT) :: YDMF_PHYS
TYPE(CPG_DYN_TYPE),             INTENT(IN)    :: YDCPG_DYN0
TYPE(MF_PHYS_SURF_TYPE),        INTENT(INOUT) :: YDMF_PHYS_SURF

TYPE(CPG_SL2_TYPE),             INTENT(INOUT) :: YDCPG_SL2
TYPE(FIELD_VARIABLES),          INTENT(INOUT) :: YDVARS




TYPE(MODEL),                    INTENT(IN)    :: YDMODEL
TYPE(TSPP_DATA),                INTENT(IN)    :: YDSPP
TYPE(TSPP_CONFIG),              INTENT(IN)    :: YDSPP_CONFIG
TYPE(TYP_DDH),                  INTENT(INOUT) :: YDDDH

#include "accldia.intfb.h"
#include "acclph.intfb.h"
#include "acdnshf.intfb.h"
#include "acdrag.intfb.h"
#include "acdrme.intfb.h"
#include "acevadcape.intfb.h"
#include "achmt.intfb.h"
#include "achmtls.intfb.h"
#include "acpluis.intfb.h"
#include "acsol.intfb.h"
#include "actqsat.intfb.h"
#include "acvisih.intfb.h"
#include "aplpar_init.intfb.h"
#include "checkmv.intfb.h"
#include "cpphinp.intfb.h"
#include "cpqsol.intfb.h"
#include "mf_phys_bayrad.intfb.h"
#include "mf_phys_corwat.intfb.h"
4% apl_arpege.F90                                                           32,0-1         4%
```

New version of APLPAR and APL_ARPEGE

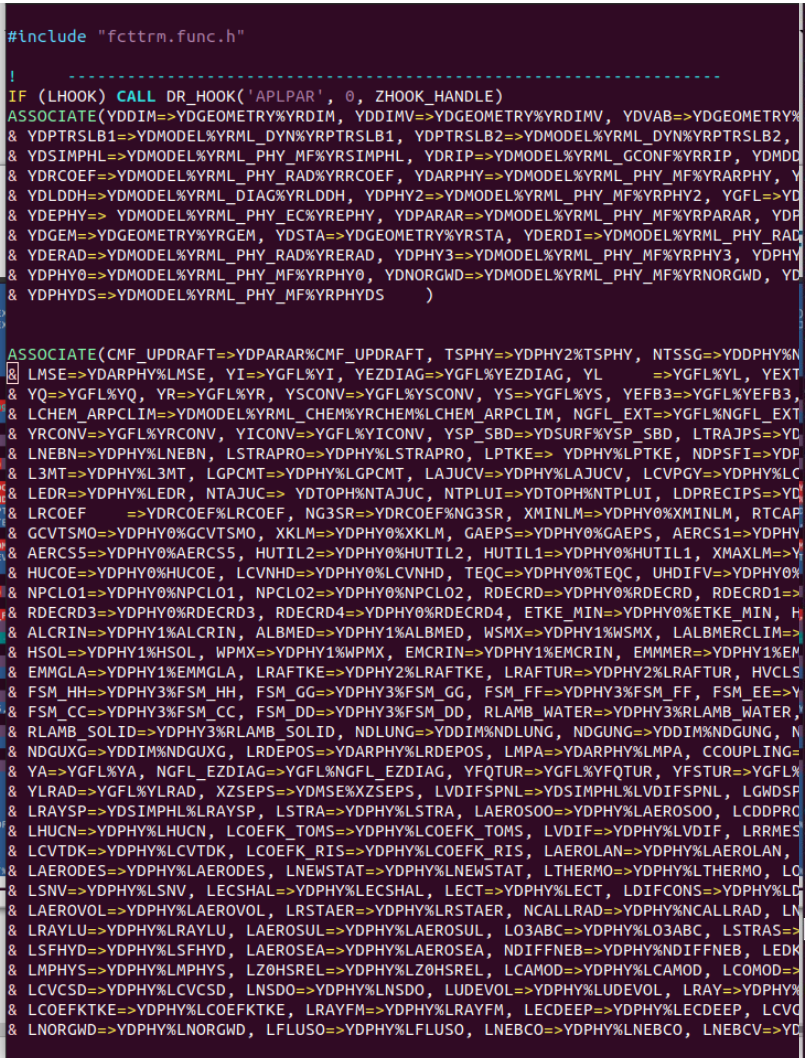


```
LOGICAL :: LL_SAVE_PHSURF

INTEGER(KIND=JPIM) :: IFIELDSS

INTEGER(KIND=JPIM) :: INSTEP_DEB,INSTEP_FIN
INTEGER(KIND=JPIM) :: JROF, JSPP

!     --- UPPER AIR PHYSICAL TENDENCIES.
REAL(KIND=JPRB) :: ZTENDH(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)      ! Enthalpy tendency.
REAL(KIND=JPRB) :: ZTENDQ(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)      ! Moisture tendency.
REAL(KIND=JPRB) :: ZTENDPTKE(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)   ! Pseudo progn. TKE

! GFL tendencies for APL_AROME (assumes YDMODEL%YRML_GCONF%YGFL%NUMFLDS>=YDMODEL%YRML_PHY_MF%YRPARAR%NRR)
! for now, use Jovi's trick :
REAL(KIND=JPRB) :: ZTENDGFL(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG,YDMODEL%YRML_GCONF%YGFL%NUMFLDS)   ! GFL tendencies

!     --- UPPER AIR PHYSICAL TENDENCIES FOR AROME.
!       (the previous one are not used in AROME)
REAL(KIND=JPRB) :: ZTENDD (YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)         ! d  tendency
REAL(KIND=JPRB) :: ZTENDEXT(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG,YDMODEL%YRML_GCONF%YGFL%NGFL_EXT)      ! GFL EXTRA tend
REAL(KIND=JPRB) :: ZTENDEXT_DEP(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG,YDMODEL%YRML_GCONF%YGFL%NGFL_EXT)  ! GFL EXTRA tend
REAL(KIND=JPRB) :: ZDIFEXT(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG,YDMODEL%YRML_GCONF%YGFL%NGFL_EXT)     ! Extra-GFL flux

REAL(KIND=JPRB) :: ZTENDU (YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)     ! U tendency without deep convection contribution
REAL(KIND=JPRB) :: ZTENDV (YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)     ! V tendency without deep convection contribution

!     --- RADIATION COEFFICIENTS FOR SIMPLIFIED PHYSICS IN GRID-POINT ---
REAL(KIND=JPRB) :: ZAC(YDCPG_OPTS%KLON,(YDCPG_OPTS%KFLEVG+1)*(YDCPG_OPTS%KFLEVG+1))    ! Curtis matrix.
REAL(KIND=JPRB) :: ZAC_HC(YDCPG_OPTS%KFLEVG+1,YDCPG_OPTS%KFLEVG+1)            ! horizontally-constant field for ZAC.



! required for INTFLEX
TYPE(TYPE_INTPROCSET) :: YLPROCSET

! SPP
REAL(KIND=JPRB) :: ZGP2DSPP(YDCPG_OPTS%KLON,YSPP%N2D)


REAL(KIND=JPRB), POINTER :: ZPTENDEFB11(:,:), ZPTENDEFB21(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDEFB31(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDG1(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDICONV1(:,:), ZPTENDI1(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDLCONV1(:,:)
REAL(KIND=JPRB), POINTER :: ZP1EZDIAG(:,:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDQ1(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDRCONV1(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDR1(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDSCONV1(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDS1(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDTKE1(:,:)
REAL(KIND=JPRB), POINTER :: ZPTENDL1(:,:)
"aplpar.F90" 5050 lines --5%--                                                                      285
```

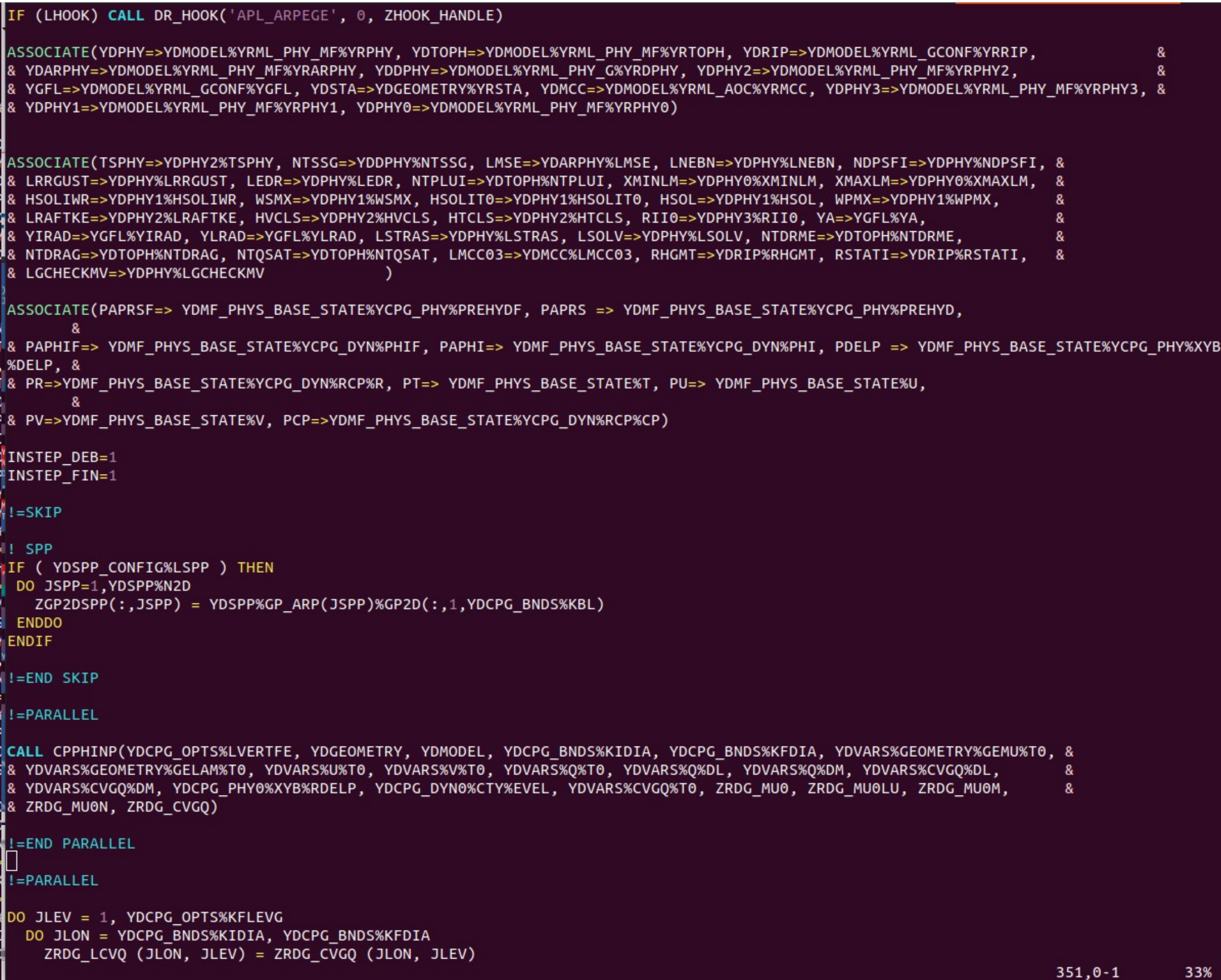


```
REAL(KIND=JPRB) :: ZGP2DSPP(YDCPG_OPTS%KLON,YDSPP%N2D)
INTEGER(KIND=JPIM) :: INLAB_CVPP(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZXTROV(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG),ZXUROV(YDCPG_OPTS%KLON,0:YDCPG_OP
REAL(KIND=JPRB) :: ZMRIPP(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZKTROV(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG),ZKUROV(YDCPG_OPTS%KLON,0:YDCPG_OP
YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZKQROV(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG),ZKQLROV(YDCPG_OPTS%KLON,0:YDCPG_O
REAL(KIND=JPRB) :: ZNEBS(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZQLIS(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZNEBS0(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZQLIS0(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZNEBC0(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)             ! Nebulosite convective ra
REAL(KIND=JPRB) :: ZNEBDIFF(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)           ! Nebulosite: calcul de la
REAL(KIND=JPRB) :: ZNEBCH(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)             ! Nebulosite convective co
REAL(KIND=JPRB) :: ZUNEBH(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)             ! Nebulosite convective hi
REAL(KIND=JPRB) :: ZFPCOR(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZPOID(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)              ! DP/(YDMODEL%YRCST%RG*DT)
STEP.
REAL(KIND=JPRB) :: ZQV(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)                ! corrected (for negative
REAL(KIND=JPRB) :: ZQI(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)                ! corrected (for negative
REAL(KIND=JPRB) :: ZQL(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)                ! corrected (for negative
REAL(KIND=JPRB) :: ZQR(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)                ! corrected (for negative
REAL(KIND=JPRB) :: ZQS(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)                ! corrected (for negative
REAL(KIND=JPRB) :: ZTENHA(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZTENQVA(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)
REAL(KIND=JPRB) :: ZCP(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)                ! new cp for turbulent dif
REAL(KIND=JPRB) :: ZFPLSL(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)           ! total liquid water flux:
REAL(KIND=JPRB) :: ZFPLSN(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)           ! total solid water flux:
REAL(KIND=JPRB) :: ZSEDIQL(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)          ! sedimentation flux of cl
REAL(KIND=JPRB) :: ZSEDIQI(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)          ! sedimentation flux of cl
REAL(KIND=JPRB) :: ZDIFCVPPQ(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)        ! Flux de CVPP (KFB or EDK
REAL(KIND=JPRB) :: ZDIFCVPPS(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)        ! Flux de CVPP (KFB or EDK
REAL(KIND=JPRB) :: ZDIFCVPPU(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)        ! Flux de CVPP (EDKF) sur
REAL(KIND=JPRB) :: ZDIFCVPPV(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)        ! Flux de CVPP (EDKF) sur
REAL(KIND=JPRB) :: ZEDMFQ(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)           ! Mass flux part of EDMF f
REAL(KIND=JPRB) :: ZEDMFS(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)           ! Mass flux part of EDMF f
REAL(KIND=JPRB) :: ZEDMFU(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)           ! Mass flux part of EDMF f
REAL(KIND=JPRB) :: ZEDMFV(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)           ! Mass flux part of EDMF f
REAL(KIND=JPRB) :: ZMF_UP(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)           ! Mass flux for implicit f
REAL(KIND=JPRB) :: ZCONDCVPPL(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)       ! Flux de condensation lic
REAL(KIND=JPRB) :: ZCONDCVPPI(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)       ! Flux de condensation gla
REAL(KIND=JPRB) :: ZPRODTH_CVPP(YDCPG_OPTS%KLON,0:YDCPG_OPTS%KFLEVG)     ! Flux de production therr
REAL(KIND=JPRB) :: ZDE2MR(YDCPG_OPTS%KLON,YDCPG_OPTS%KFLEVG)             ! temporary array for conv
REAL(KIND=JPRB) :: ZXDROV(YDCPG_OPTS%KLON)
REAL(KIND=JPRB) :: ZXHROV(YDCPG_OPTS%KLON)
REAL(KIND=JPRB) :: ZUGST(YDCPG_OPTS%KLON)
REAL(KIND=JPRB) :: ZVGST(YDCPG_OPTS%KLON)
REAL(KIND=JPRB) :: ZCDROV(YDCPG_OPTS%KLON)                               ! ZCDROV     : PCD RENORME
REAL(KIND=JPRB) :: ZCHROV(YDCPG_OPTS%KLON)                               ! ZCHROV     : PCH RENORME
REAL(KIND=JPRB) :: ZDQSTS(YDCPG_OPTS%KLON)                               ! ZDQSTS     : DERIVEE DE
REAL(KIND=JPRB) :: ZGWDCS(YDCPG_OPTS%KLON)                               ! ZGWDCS     : VARIABLE DE
RHO*N0/G).
"apl_arpege.F90" 957 lines --15%--
```

New version of APLPAR and APL_ARPEGE

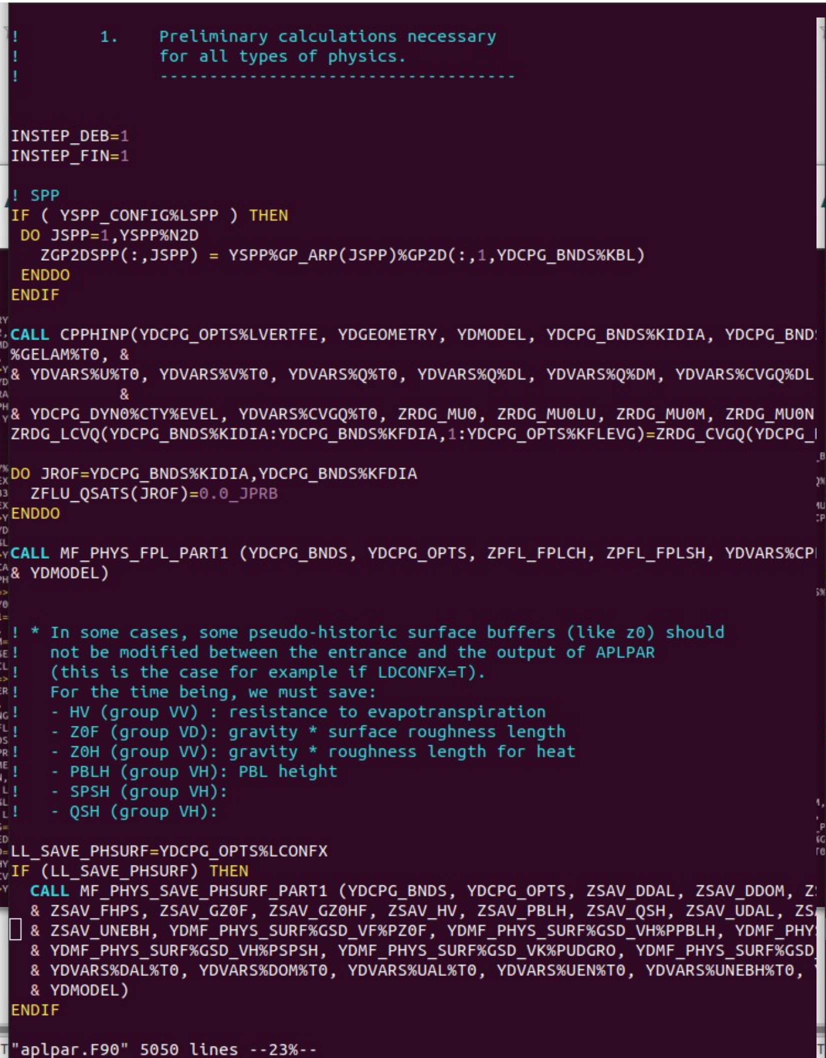


```
#include "fcttrm.func.h"

!     ----------------------------------------------------------------------
IF (LHOOK) CALL DR_HOOK('APLPAR', 0, ZHOOK_HANDLE)
ASSOCIATE(YDDIM=>YDGEOMETRY%YRDIM, YDDIMV=>YDGEOMETRY%YRDIMV, YDVAB=>YDGEOMETRY%
& YDPTRSLB1=>YDMODEL%YRML_DYN%YRPTRSLB1, YDPTRSLB2=>YDMODEL%YRML_DYN%YRPTRSLB2,
& YDSIMPHL=>YDMODEL%YRML_PHY_MF%YRSIMPHL, YDRIP=>YDMODEL%YRML_GCONF%YRRIP, YDMDD
& YDRCOEF=>YDMODEL%YRML_PHY_RAD%YRRCOEF, YDARPHY=>YDMODEL%YRML_PHY_MF%YRARPHY, Y
& YDLDDH=>YDMODEL%YRML_DIAG%YRLDDH, YDPHY2=>YDMODEL%YRML_PHY_MF%YRPHY2, YGFL=>YD
& YDEPHY=> YDMODEL%YRML_PHY_EC%YREPHY, YDPARAR=>YDMODEL%YRML_PHY_MF%YRPARAR, YDP
& YDGEM=>YDGEOMETRY%YRGEM, YDSTA=>YDGEOMETRY%YRSTA, YDERDI=>YDMODEL%YRML_PHY_RAD
& YDERAD=>YDMODEL%YRML_PHY_RAD%YRERAD, YDPHY3=>YDMODEL%YRML_PHY_MF%YRPHY3, YDPHY
& YDPHY0=>YDMODEL%YRML_PHY_MF%YRPHY0, YDNORGWD=>YDMODEL%YRML_PHY_MF%YRNORGWD, YD
& YDPHYDS=>YDMODEL%YRML_PHY_MF%YRPHYDS     )


ASSOCIATE(CMF_UPDRAFT=>YDPARAR%CMF_UPDRAFT, TSPHY=>YDPHY2%TSPHY, NTSSG=>YDDPHY%N
& LMSE=>YDARPHY%LMSE, YI=>YGFL%YI, YEZDIAG=>YGFL%YEZDIAG, YL      =>YGFL%YL, YEXT
& YQ=>YGFL%YQ, YR=>YGFL%YR, YSCONV=>YGFL%YSCONV, YS=>YGFL%YS, YEFB3=>YGFL%YEFB3,
& LCHEM_ARPCLIM=>YDMODEL%YRML_CHEM%YRCHEM%LCHEM_ARPCLIM, NGFL_EXT=>YGFL%NGFL_EXT
& YRCONV=>YGFL%YRCONV, YICONV=>YGFL%YICONV, YSP_SBD=>YDSURF%YSP_SBD, LTRAJPS=>YD
& LNEBN=>YDPHY%LNEBN, LSTRAPRO=>YDPHY%LSTRAPRO, LPTKE=> YDPHY%LPTKE, NDPSFI=>YDP
& L3MT=>YDPHY%L3MT, LGPCMT=>YDPHY%LGPCMT, LAJUCV=>YDPHY%LAJUCV, LCVPGY=>YDPHY%LC
& LEDR=>YDPHY%LEDR, NTAJUC=> YDTOPH%NTAJUC, NTPLUI=>YDTOPH%NTPLUI, LDPRECIPS=>YD
& LRCOEF    =>YDRCOEF%LRCOEF, NG3SR=>YDRCOEF%NG3SR, XMINLM=>YDPHY0%XMINLM, RTCAP
& GCVTSMO=>YDPHY0%GCVTSMO, XKLM=>YDPHY0%XKLM, GAEPS=>YDPHY0%GAEPS, AERCS1=>YDPHY
& AERCS5=>YDPHY0%AERCS5, HUTIL2=>YDPHY0%HUTIL2, HUTIL1=>YDPHY0%HUTIL1, XMAXLM=>Y
& HUCOE=>YDPHY0%HUCOE, LCVNHD=>YDPHY0%LCVNHD, TEQC=>YDPHY0%TEQC, UHDIFV=>YDPHY0%
& NPCLO1=>YDPHY0%NPCLO1, NPCLO2=>YDPHY0%NPCLO2, RDECRD=>YDPHY0%RDECRD, RDECRD1=>
& RDECRD3=>YDPHY0%RDECRD3, RDECRD4=>YDPHY0%RDECRD4, ETKE_MIN=>YDPHY0%ETKE_MIN, H
& ALCRIN=>YDPHY1%ALCRIN, ALBMED=>YDPHY1%ALBMED, WSMX=>YDPHY1%WSMX, LALBMERCLIM=>
& HSOL=>YDPHY1%HSOL, WPMX=>YDPHY1%WPMX, EMCRIN=>YDPHY1%EMCRIN, EMMMER=>YDPHY1%EM
& EMMGLA=>YDPHY1%EMMGLA, LRAFTKE=>YDPHY2%LRAFTKE, LRAFTUR=>YDPHY2%LRAFTUR, HVCLS
& FSM_HH=>YDPHY3%FSM_HH, FSM_GG=>YDPHY3%FSM_GG, FSM_FF=>YDPHY3%FSM_FF, FSM_EE=>Y
& FSM_CC=>YDPHY3%FSM_CC, FSM_DD=>YDPHY3%FSM_DD, RLAMB_WATER=>YDPHY3%RLAMB_WATER,
& RLAMB_SOLID=>YDPHY3%RLAMB_SOLID, NDLUNG=>YDDIM%NDLUNG, NDGUNG=>YDDIM%NDGUNG, N
& NDGUXG=>YDDIM%NDGUXG, LRDEPOS=>YDARPHY%LRDEPOS, LMPA=>YDARPHY%LMPA, CCOUPLING=
& YA=>YGFL%YA, NGFL_EZDIAG=>YGFL%NGFL_EZDIAG, YFQTUR=>YGFL%YFQTUR, YFSTUR=>YGFL%
& YLRAD=>YGFL%YLRAD, XZSEPS=>YDMSE%XZSEPS, LVDIFSPNL=>YDSIMPHL%LVDIFSPNL, LGWDSP
& LRAYSP=>YDSIMPHL%LRAYSP, LSTRA=>YDPHY%LSTRA, LAEROSOO=>YDPHY%LAEROSOO, LCDDPRO
& LHUCN=>YDPHY%LHUCN, LCOEFK_TOMS=>YDPHY%LCOEFK_TOMS, LVDIF=>YDPHY%LVDIF, LRRMES
& LCVTDK=>YDPHY%LCVTDK, LCOEFK_RIS=>YDPHY%LCOEFK_RIS, LAEROLAN=>YDPHY%LAEROLAN,
& LAERODES=>YDPHY%LAERODES, LNEWSTAT=>YDPHY%LNEWSTAT, LTHERMO=>YDPHY%LTHERMO, LO
& LSNV=>YDPHY%LSNV, LECSHAL=>YDPHY%LECSHAL, LECT=>YDPHY%LECT, LDIFCONS=>YDPHY%LD
& LAEROVOL=>YDPHY%LAEROVOL, LRSTAER=>YDPHY%LRSTAER, NCALLRAD=>YDPHY%NCALLRAD, LN
& LRAYLU=>YDPHY%LRAYLU, LAEROSUL=>YDPHY%LAEROSUL, LO3ABC=>YDPHY%LO3ABC, LSTRAS=>
& LSFHYD=>YDPHY%LSFHYD, LAEROSEA=>YDPHY%LAEROSEA, NDIFFNEB=>YDPHY%NDIFFNEB, LEDK
& LMPHYS=>YDPHY%LMPHYS, LZ0HSREL=>YDPHY%LZ0HSREL, LCAMOD=>YDPHY%LCAMOD, LCOMOD=>
& LCVCSD=>YDPHY%LCVCSD, LNSDO=>YDPHY%LNSDO, LUDEVOL=>YDPHY%LUDEVOL, LRAY=>YDPHY%
& LCOEFKTKE=>YDPHY%LCOEFKTKE, LRAYFM=>YDPHY%LRAYFM, LECDEEP=>YDPHY%LECDEEP, LCVC
& LNORGWD=>YDPHY%LNORGWD, LFLUSO=>YDPHY%LFLUSO, LNEBCO=>YDPHY%LNEBCO, LNEBCV=>YD
```

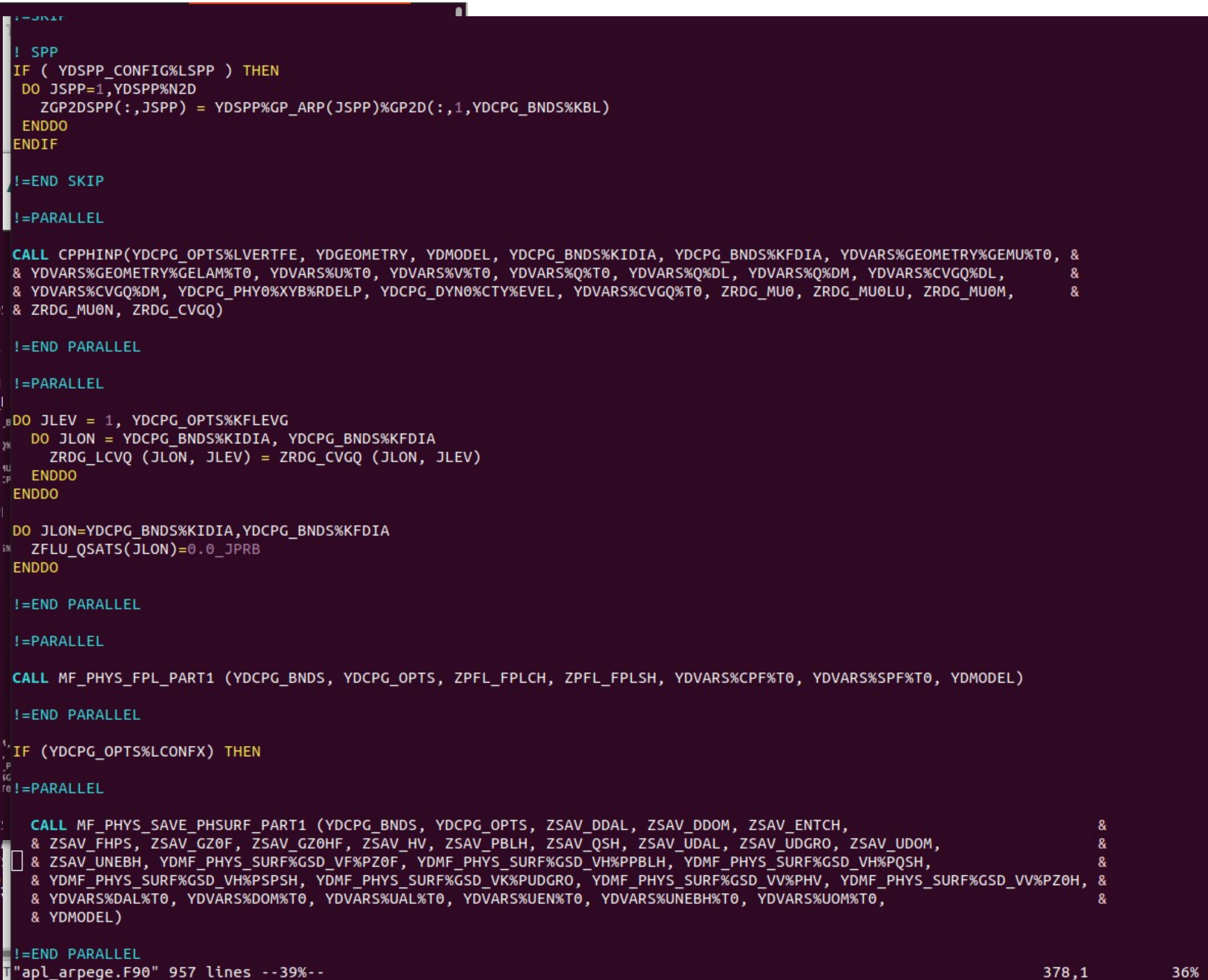


```
IF (LHOOK) CALL DR_HOOK('APL_ARPEGE', 0, ZHOOK_HANDLE)

ASSOCIATE(YDPHY=>YDMODEL%YRML_PHY_MF%YRPHY, YDTOPH=>YDMODEL%YRML_PHY_MF%YRTOPH, YDRIP=>YDMODEL%YRML_GCONF%YRRIP,          &
& YDARPHY=>YDMODEL%YRML_PHY_MF%YRARPHY, YDDPHY=>YDMODEL%YRML_PHY_G%YRDPHY, YDPHY2=>YDMODEL%YRML_PHY_MF%YRPHY2,            &
& YGFL=>YDMODEL%YRML_GCONF%YGFL, YDSTA=>YDGEOMETRY%YRSTA, YDMCC=>YDMODEL%YRML_AOC%YRMCC, YDPHY3=>YDMODEL%YRML_PHY_MF%YRPHY3, &
& YDPHY1=>YDMODEL%YRML_PHY_MF%YRPHY1, YDPHY0=>YDMODEL%YRML_PHY_MF%YRPHY0)


ASSOCIATE(TSPHY=>YDPHY2%TSPHY, NTSSG=>YDDPHY%NTSSG, LMSE=>YDARPHY%LMSE, LNEBN=>YDPHY%LNEBN, NDPSFI=>YDPHY%NDPSFI, &
& LRRGUST=>YDPHY%LRRGUST, LEDR=>YDPHY%LEDR, NTPLUI=>YDTOPH%NTPLUI, XMINLM=>YDPHY0%XMINLM, XMAXLM=>YDPHY0%XMAXLM,  &
& HSOLIWR=>YDPHY1%HSOLIWR, WSMX=>YDPHY1%WSMX, HSOLIT0=>YDPHY1%HSOLIT0, HSOL=>YDPHY1%HSOL, WPMX=>YDPHY1%WPMX,       &
& LRAFTKE=>YDPHY2%LRAFTKE, HVCLS=>YDPHY2%HVCLS, HTCLS=>YDPHY2%HTCLS, RII0=>YDPHY3%RII0, YA=>YGFL%YA,             &
& YIRAD=>YGFL%YIRAD, YLRAD=>YGFL%YLRAD, LSTRAS=>YDPHY%LSTRAS, LSOLV=>YDPHY%LSOLV, NTDRME=>YDTOPH%NTDRME,          &
& NTDRAG=>YDTOPH%NTDRAG, NTQSAT=>YDTOPH%NTQSAT, LMCC03=>YDMCC%LMCC03, RHGMT=>YDRIP%RHGMT, RSTATI=>YDRIP%RSTATI,   &
& LGCHECKMV=>YDPHY%LGCHECKMV              )

ASSOCIATE(PAPRSF=> YDMF_PHYS_BASE_STATE%YCPG_PHY%PREHYDF, PAPRS => YDMF_PHYS_BASE_STATE%YCPG_PHY%PREHYD,
       &
& PAPHIF=> YDMF_PHYS_BASE_STATE%YCPG_DYN%PHIF, PAPHI=> YDMF_PHYS_BASE_STATE%YCPG_DYN%PHI, PDELP => YDMF_PHYS_BASE_STATE%YCPG_PHY%XYB
%DELP, &
& PR=>YDMF_PHYS_BASE_STATE%YCPG_DYN%RCP%R, PT=> YDMF_PHYS_BASE_STATE%T, PU=> YDMF_PHYS_BASE_STATE%U,
       &
& PV=>YDMF_PHYS_BASE_STATE%V, PCP=>YDMF_PHYS_BASE_STATE%YCPG_DYN%RCP%CP)

INSTEP_DEB=1
INSTEP_FIN=1

!=SKIP

! SPP
IF ( YDSPP_CONFIG%LSPP ) THEN
 DO JSPP=1,YDSPP%N2D
   ZGP2DSPP(:,JSPP) = YDSPP%GP_ARP(JSPP)%GP2D(:,1,YDCPG_BNDS%KBL)
 ENDDO
ENDIF

!=END SKIP

!=PARALLEL

CALL CPPHINP(YDCPG_OPTS%LVERTFE, YDGEOMETRY, YDMODEL, YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA, YDVARS%GEOMETRY%GEMU%T0, &
& YDVARS%GEOMETRY%GELAM%T0, YDVARS%U%T0, YDVARS%V%T0, YDVARS%Q%T0, YDVARS%Q%DL, YDVARS%Q%DM, YDVARS%CVGQ%DL,       &
& YDVARS%CVGQ%DM, YDCPG_PHY0%XYB%RDELP, YDCPG_DYN0%CTY%EVEL, YDVARS%CVGQ%T0, ZRDG_MU0, ZRDG_MU0LU, ZRDG_MU0M,      &
& ZRDG_MU0N, ZRDG_CVGQ)

!=END PARALLEL

!=PARALLEL

DO JLEV = 1, YDCPG_OPTS%KFLEVG
  DO JLON = YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA
    ZRDG_LCVQ (JLON, JLEV) = ZRDG_CVGQ (JLON, JLEV)
                                                                                                       351,0-1       33%
```

New version of APLPAR and APL_ARPEGE



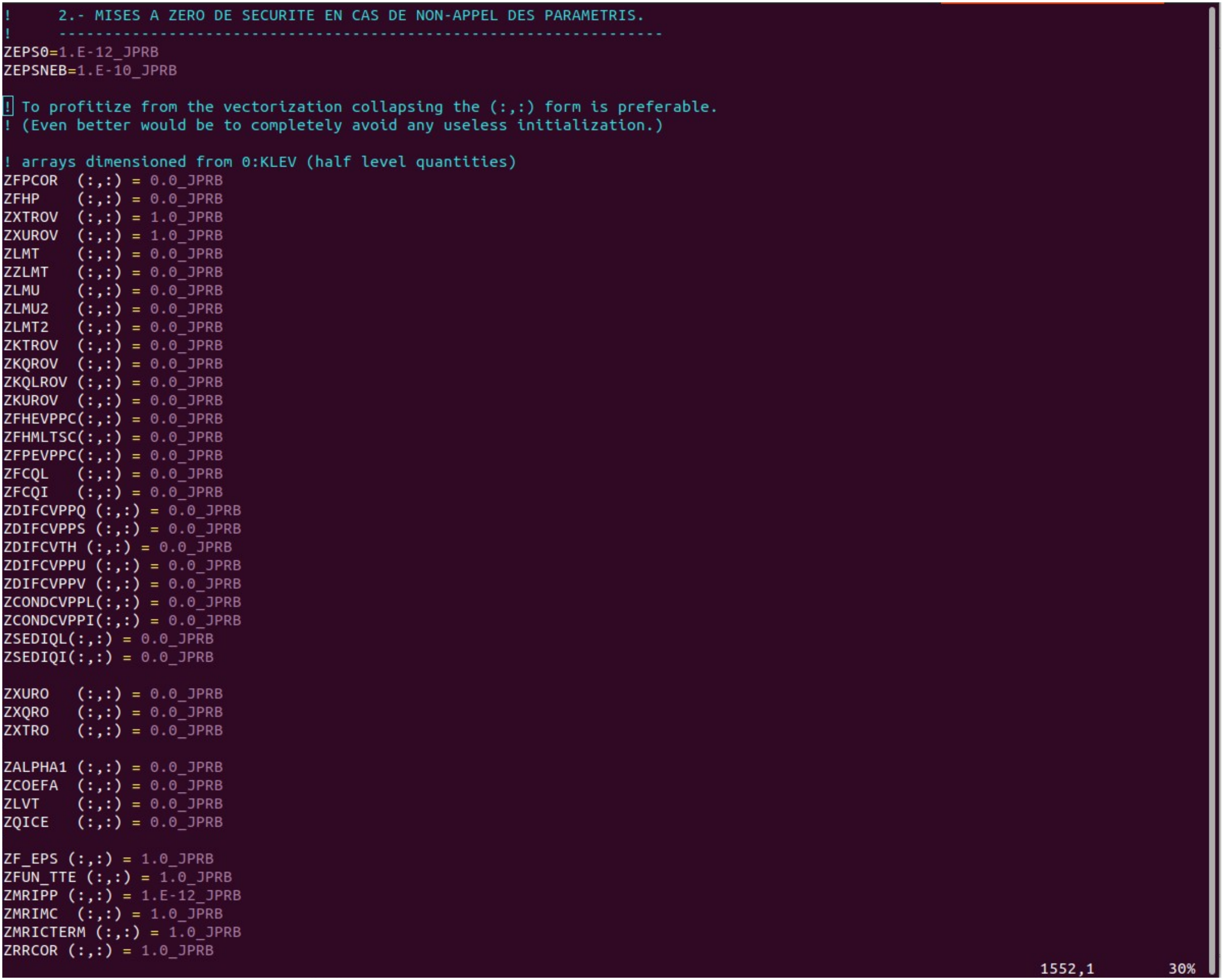
```
!        1.    Preliminary calculations necessary
!              for all types of physics.
!              ------------------------------------


INSTEP_DEB=1
INSTEP_FIN=1

! SPP
IF ( YSPP_CONFIG%LSPP ) THEN
 DO JSPP=1,YSPP%N2D
   ZGP2DSPP(:,JSPP) = YSPP%GP_ARP(JSPP)%GP2D(:,1,YDCPG_BNDS%KBL)
 ENDDO
ENDIF

CALL CPPHINP(YDCPG_OPTS%LVERTFE, YDGEOMETRY, YDMODEL, YDCPG_BNDS%KIDIA, YDCPG_BND
%GELAM%T0, &
& YDVARS%U%T0, YDVARS%V%T0, YDVARS%Q%T0, YDVARS%Q%DL, YDVARS%Q%DM, YDVARS%CVGQ%DL
           &
& YDCPG_DYN0%CTY%EVEL, YDVARS%CVGQ%T0, ZRDG_MU0, ZRDG_MU0LU, ZRDG_MU0M, ZRDG_MU0N
ZRDG_LCVQ(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA,1:YDCPG_OPTS%KFLEVG)=ZRDG_CVGQ(YDCPG_

DO JROF=YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
  ZFLU_QSATS(JROF)=0.0_JPRB
ENDDO

CALL MF_PHYS_FPL_PART1 (YDCPG_BNDS, YDCPG_OPTS, ZPFL_FPLCH, ZPFL_FPLSH, YDVARS%CP
& YDMODEL)


! * In some cases, some pseudo-historic surface buffers (like z0) should
!   not be modified between the entrance and the output of APLPAR
!   (this is the case for example if LDCONFX=T).
!   For the time being, we must save:
!   - HV (group VV) : resistance to evapotranspiration
!   - Z0F (group VD): gravity * surface roughness length
!   - Z0H (group VV): gravity * roughness length for heat
!   - PBLH (group VH): PBL height
!   - SPSH (group VH):
!   - QSH (group VH):

LL_SAVE_PHSURF=YDCPG_OPTS%LCONFX
IF (LL_SAVE_PHSURF) THEN
  CALL MF_PHYS_SAVE_PHSURF_PART1 (YDCPG_BNDS, YDCPG_OPTS, ZSAV_DDAL, ZSAV_DDOM, Z
  & ZSAV_FHPS, ZSAV_GZ0F, ZSAV_GZ0HF, ZSAV_HV, ZSAV_PBLH, ZSAV_QSH, ZSAV_UDAL, ZS
  & ZSAV_UNEBH, YDMF_PHYS_SURF%GSD_VF%PZ0F, YDMF_PHYS_SURF%GSD_VH%PPBLH, YDMF_PHY
  & YDMF_PHYS_SURF%GSD_VH%PSPSH, YDMF_PHYS_SURF%GSD_VK%PUDGRO, YDMF_PHYS_SURF%GSD
  & YDVARS%DAL%T0, YDVARS%DOM%T0, YDVARS%UAL%T0, YDVARS%UEN%T0, YDVARS%UNEBH%T0,
  & YDMODEL)
ENDIF

"aplpar.F90" 5050 lines --23%--
```



```
!=SKIP

! SPP
IF ( YDSPP_CONFIG%LSPP ) THEN
 DO JSPP=1,YDSPP%N2D
   ZGP2DSPP(:,JSPP) = YDSPP%GP_ARP(JSPP)%GP2D(:,1,YDCPG_BNDS%KBL)
 ENDDO
ENDIF

!=END SKIP

!=PARALLEL

CALL CPPHINP(YDCPG_OPTS%LVERTFE, YDGEOMETRY, YDMODEL, YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA, YDVARS%GEOMETRY%GEMU%T0, &
& YDVARS%GEOMETRY%GELAM%T0, YDVARS%U%T0, YDVARS%V%T0, YDVARS%Q%T0, YDVARS%Q%DL, YDVARS%Q%DM, YDVARS%CVGQ%DL,       &
& YDVARS%CVGQ%DM, YDCPG_PHY0%XYB%RDELP, YDCPG_DYN0%CTY%EVEL, YDVARS%CVGQ%T0, ZRDG_MU0, ZRDG_MU0LU, ZRDG_MU0M,      &
& ZRDG_MU0N, ZRDG_CVGQ)

!=END PARALLEL

!=PARALLEL

DO JLEV = 1, YDCPG_OPTS%KFLEVG
  DO JLON = YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA
    ZRDG_LCVQ (JLON, JLEV) = ZRDG_CVGQ (JLON, JLEV)
  ENDDO
ENDDO

DO JLON=YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
  ZFLU_QSATS(JLON)=0.0_JPRB
ENDDO

!=END PARALLEL

!=PARALLEL

CALL MF_PHYS_FPL_PART1 (YDCPG_BNDS, YDCPG_OPTS, ZPFL_FPLCH, ZPFL_FPLSH, YDVARS%CPF%T0, YDVARS%SPF%T0, YDMODEL)

!=END PARALLEL

IF (YDCPG_OPTS%LCONFX) THEN

!=PARALLEL

  CALL MF_PHYS_SAVE_PHSURF_PART1 (YDCPG_BNDS, YDCPG_OPTS, ZSAV_DDAL, ZSAV_DDOM, ZSAV_ENTCH,                                &
  & ZSAV_FHPS, ZSAV_GZ0F, ZSAV_GZ0HF, ZSAV_HV, ZSAV_PBLH, ZSAV_QSH, ZSAV_UDAL, ZSAV_UDGRO, ZSAV_UDOM,                     &
  & ZSAV_UNEBH, YDMF_PHYS_SURF%GSD_VF%PZ0F, YDMF_PHYS_SURF%GSD_VH%PPBLH, YDMF_PHYS_SURF%GSD_VH%PQSH,                    &
  & YDMF_PHYS_SURF%GSD_VH%PSPSH, YDMF_PHYS_SURF%GSD_VK%PUDGRO, YDMF_PHYS_SURF%GSD_VV%PHV, YDMF_PHYS_SURF%GSD_VV%PZ0H, &
  & YDVARS%DAL%T0, YDVARS%DOM%T0, YDVARS%UAL%T0, YDVARS%UEN%T0, YDVARS%UNEBH%T0, YDVARS%UOM%T0,                       &
  & YDMODEL)

!=END PARALLEL
"apl_arpege.F90" 957 lines --39%--                                                                          378,1         36%
```

New version of APLPAR and APL_ARPEGE

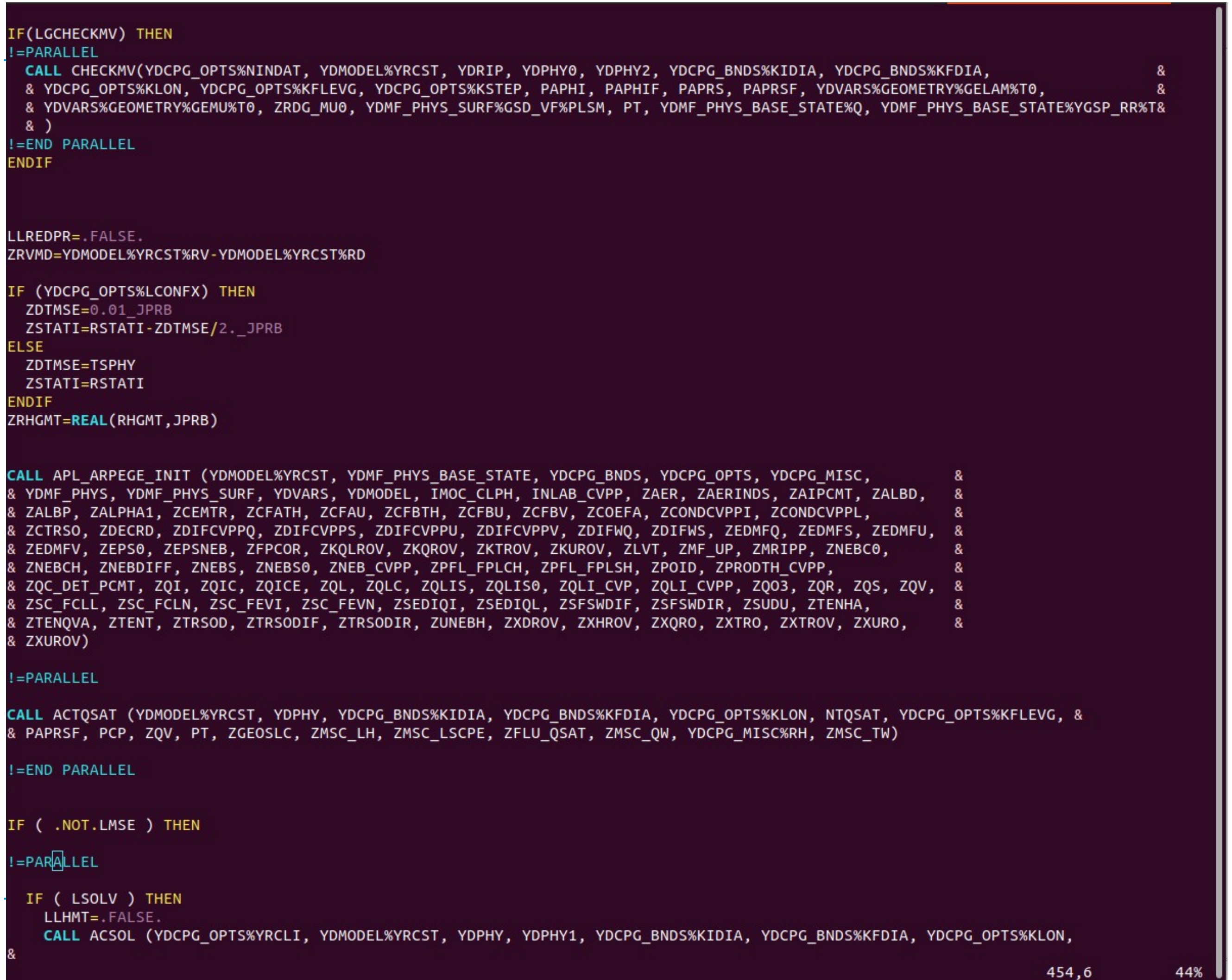
This part of aplpar
Now in
apl_arpege_init



```
!     2.- MISES A ZERO DE SECURITE EN CAS DE NON-APPEL DES PARAMETRIS.
!     -----------------------------------------------------------------
ZEPS0=1.E-12_JPRB
ZEPSNEB=1.E-10_JPRB

! To profitize from the vectorization collapsing the (:,:) form is preferable.
! (Even better would be to completely avoid any useless initialization.)

! arrays dimensioned from 0:KLEV (half level quantities)
ZFPCOR   (:,:) = 0.0_JPRB
ZFHP     (:,:) = 0.0_JPRB
ZXTROV   (:,:) = 1.0_JPRB
ZXUROV   (:,:) = 1.0_JPRB
ZLMT     (:,:) = 0.0_JPRB
ZZLMT    (:,:) = 0.0_JPRB
ZLMU     (:,:) = 0.0_JPRB
ZLMU2    (:,:) = 0.0_JPRB
ZLMT2    (:,:) = 0.0_JPRB
ZKTROV   (:,:) = 0.0_JPRB
ZKQROV   (:,:) = 0.0_JPRB
ZKQLROV  (:,:) = 0.0_JPRB
ZKUROV   (:,:) = 0.0_JPRB
ZFHEVPPC(:,:) = 0.0_JPRB
ZFHMLTSC(:,:) = 0.0_JPRB
ZFPEVPPC(:,:) = 0.0_JPRB
ZFCQL    (:,:) = 0.0_JPRB
ZFCQI    (:,:) = 0.0_JPRB
ZDIFCVPPQ (:,:) = 0.0_JPRB
ZDIFCVPPS (:,:) = 0.0_JPRB
ZDIFCVTH (:,:) = 0.0_JPRB
ZDIFCVPPU (:,:) = 0.0_JPRB
ZDIFCVPPV (:,:) = 0.0_JPRB
ZCONDCVPPL(:,:) = 0.0_JPRB
ZCONDCVPPI(:,:) = 0.0_JPRB
ZSEDIQL(:,:) = 0.0_JPRB
ZSEDIQI(:,:) = 0.0_JPRB

ZXURO    (:,:) = 0.0_JPRB
ZXQRO    (:,:) = 0.0_JPRB
ZXTRO    (:,:) = 0.0_JPRB

ZALPHA1 (:,:) = 0.0_JPRB
ZCOEFA   (:,:) = 0.0_JPRB
ZLVT     (:,:) = 0.0_JPRB
ZQICE    (:,:) = 0.0_JPRB

ZF_EPS (:,:) = 1.0_JPRB
ZFUN_TTE (:,:) = 1.0_JPRB
ZMRIPP (:,:) = 1.E-12_JPRB
ZMRIMC  (:,:) = 1.0_JPRB
ZMRICTERM (:,:) = 1.0_JPRB
ZRRCOR (:,:) = 1.0_JPRB
                                                                   1552,1        30%
```

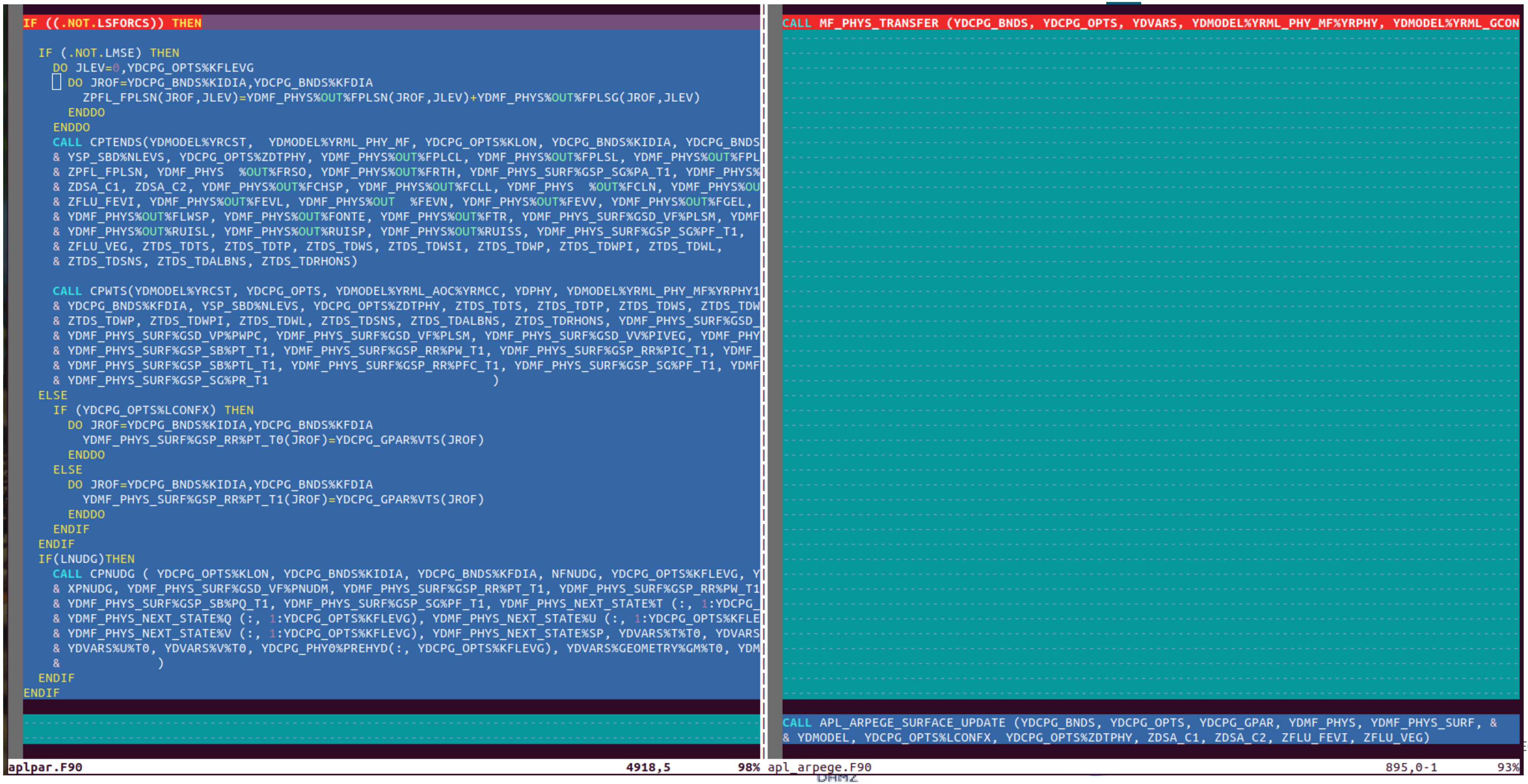
New version of APLPAR and APL_ARPEGE

apl_arpege



```
IF(LGCHECKMV) THEN
!=PARALLEL
  CALL CHECKMV(YDCPG_OPTS%NINDAT, YDMODEL%YRCST, YDRIP, YDPHY0, YDPHY2, YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA,               &
  & YDCPG_OPTS%KLON, YDCPG_OPTS%KFLEVG, YDCPG_OPTS%KSTEP, PAPHI, PAPHIF, PAPRS, PAPRSF, YDVARS%GEOMETRY%GELAM%T0,        &
  & YDVARS%GEOMETRY%GEMU%T0, ZRDG_MU0, YDMF_PHYS_SURF%GSD_VF%PLSM, PT, YDMF_PHYS_BASE_STATE%Q, YDMF_PHYS_BASE_STATE%YGSP_RR%T&
  & )
!=END PARALLEL
ENDIF



LLREDPR=.FALSE.
ZRVMD=YDMODEL%YRCST%RV-YDMODEL%YRCST%RD

IF (YDCPG_OPTS%LCONFX) THEN
  ZDTMSE=0.01_JPRB
  ZSTATI=RSTATI-ZDTMSE/2._JPRB
ELSE
  ZDTMSE=TSPHY
  ZSTATI=RSTATI
ENDIF
ZRHGMT=REAL(RHGMT,JPRB)


CALL APL_ARPEGE_INIT (YDMODEL%YRCST, YDMF_PHYS_BASE_STATE, YDCPG_BNDS, YDCPG_OPTS, YDCPG_MISC,          &
& YDMF_PHYS, YDMF_PHYS_SURF, YDVARS, YDMODEL, IMOC_CLPH, INLAB_CVPP, ZAER, ZAERINDS, ZAIPCMT, ZALBD,   &
& ZALBP, ZALPHA1, ZCEMTR, ZCFATH, ZCFAU, ZCFBTH, ZCFBU, ZCFBV, ZCOEFA, ZCONDCVPPI, ZCONDCVPPL,          &
& ZCTRSO, ZDECRD, ZDIFCVPPQ, ZDIFCVPPS, ZDIFCVPPU, ZDIFCVPPV, ZDIFWQ, ZDIFWS, ZEDMFQ, ZEDMFS, ZEDMFU,  &
& ZEDMFV, ZEPS0, ZEPSNEB, ZFPCOR, ZKQLROV, ZKQROV, ZKTROV, ZKUROV, ZLVT, ZMF_UP, ZMRIPP, ZNEBC0,       &
& ZNEBCH, ZNEBDIFF, ZNEBS, ZNEBS0, ZNEB_CVPP, ZPFL_FPLCH, ZPFL_FPLSH, ZPOID, ZPRODTH_CVPP,              &
& ZQC_DET_PCMT, ZQI, ZQIC, ZQICE, ZQL, ZQLC, ZQLIS, ZQLIS0, ZQLI_CVP, ZQLI_CVPP, ZQO3, ZQR, ZQS, ZQV,  &
& ZSC_FCLL, ZSC_FCLN, ZSC_FEVI, ZSC_FEVN, ZSEDIQI, ZSEDIQL, ZSFSWDIF, ZSFSWDIR, ZSUDU, ZTENHA,          &
& ZTENQVA, ZTENT, ZTRSOD, ZTRSODIF, ZTRSODIR, ZUNEBH, ZXDROV, ZXHROV, ZXQRO, ZXTRO, ZXTROV, ZXURO,     &
& ZXUROV)

!=PARALLEL

CALL ACTQSAT (YDMODEL%YRCST, YDPHY, YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA, YDCPG_OPTS%KLON, NTQSAT, YDCPG_OPTS%KFLEVG, &
& PAPRSF, PCP, ZQV, PT, ZGEOSLC, ZMSC_LH, ZMSC_LSCPE, ZFLU_QSAT, ZMSC_QW, YDCPG_MISC%RH, ZMSC_TW)

!=END PARALLEL


IF ( .NOT.LMSE ) THEN

!=PARALLEL

  IF ( LSOLV ) THEN
    LLHMT=.FALSE.
    CALL ACSOL (YDCPG_OPTS%YRCLI, YDMODEL%YRCST, YDPHY, YDPHY1, YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA, YDCPG_OPTS%KLON,
&
                                                                                                          454,6          44%
```

New version of APLPAR and APL_ARPEGE



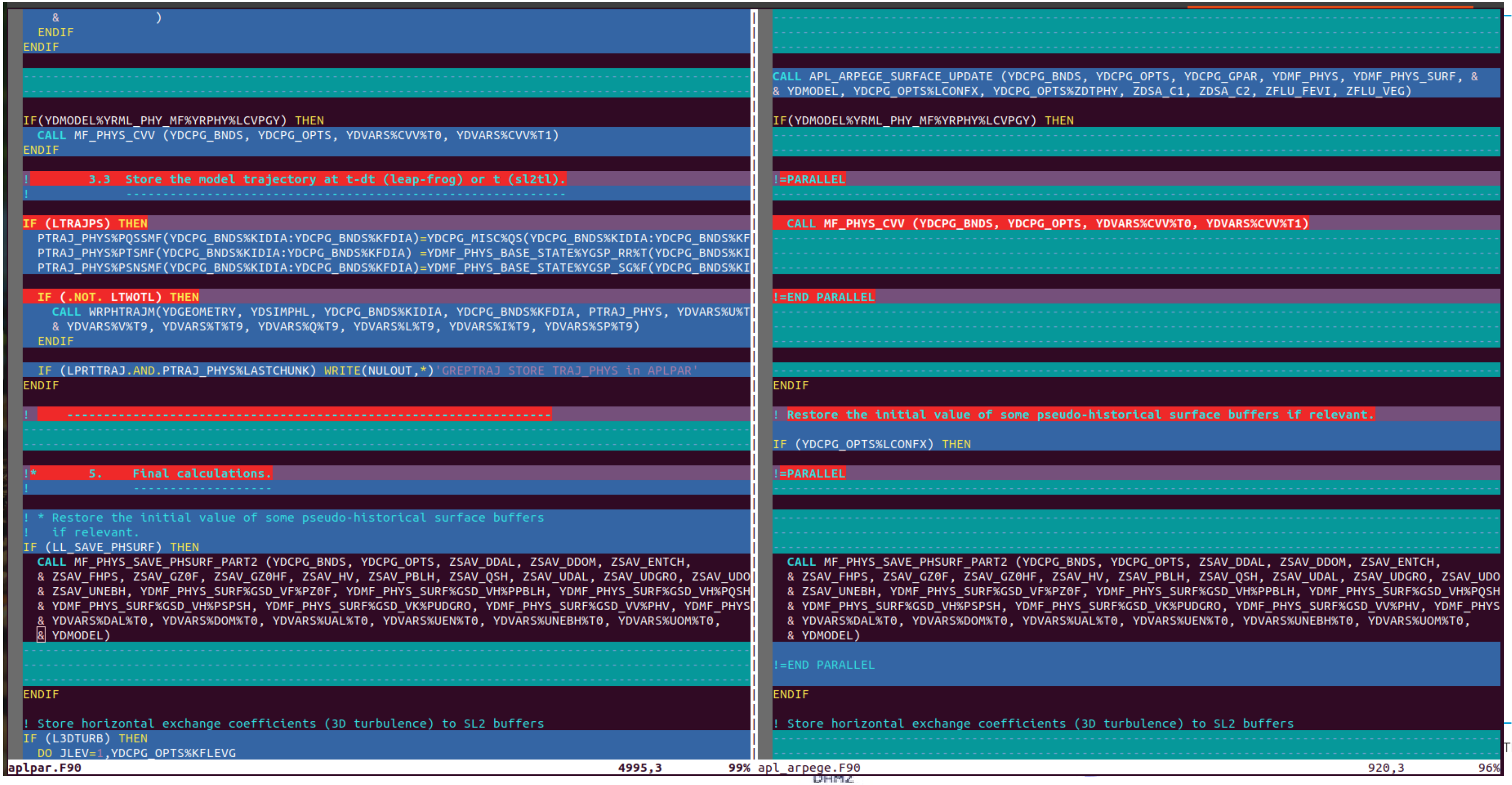
```
IF ((.NOT.LSFORCS)) THEN

  IF (.NOT.LMSE) THEN
    DO JLEV=0,YDCPG_OPTS%KFLEVG
      DO JROF=YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
        ZPFL_FPLSN(JROF,JLEV)=YDMF_PHYS%OUT%FPLSN(JROF,JLEV)+YDMF_PHYS%OUT%FPLSG(JROF,JLEV)
      ENDDO
    ENDDO
    CALL CPTENDS(YDMODEL%YRCST,  YDMODEL%YRML_PHY_MF, YDCPG_OPTS%KLON, YDCPG_BNDS%KIDIA, YDCPG_BNDS
    & YSP_SBD%NLEVS, YDCPG_OPTS%ZDTPHY, YDMF_PHYS%OUT%FPLCL, YDMF_PHYS%OUT%FPLSL, YDMF_PHYS%OUT%FPL
    & ZPFL_FPLSN, YDMF_PHYS  %OUT%FRSO, YDMF_PHYS%OUT%FRTH, YDMF_PHYS_SURF%GSP_SG%PA_T1, YDMF_PHYS%
    & ZDSA_C1, ZDSA_C2, YDMF_PHYS%OUT%FCHSP, YDMF_PHYS%OUT%FCLL, YDMF_PHYS  %OUT%FCLN, YDMF_PHYS%OU
    & ZFLU_FEVI, YDMF_PHYS%OUT%FEVL, YDMF_PHYS%OUT  %FEVN, YDMF_PHYS%OUT%FEVV, YDMF_PHYS%OUT%FGEL,
    & YDMF_PHYS%OUT%FLWSP, YDMF_PHYS%OUT%FONTE, YDMF_PHYS%OUT%FTR, YDMF_PHYS_SURF%GSD_VF%PLSM, YDMF
    & YDMF_PHYS%OUT%RUISL, YDMF_PHYS%OUT%RUISP, YDMF_PHYS%OUT%RUISS, YDMF_PHYS_SURF%GSP_SG%PF_T1,
    & ZFLU_VEG, ZTDS_TDTS, ZTDS_TDTP, ZTDS_TDWS, ZTDS_TDWSI, ZTDS_TDWP, ZTDS_TDWPI, ZTDS_TDWL,
    & ZTDS_TDSNS, ZTDS_TDALBNS, ZTDS_TDRHONS)

    CALL CPWTS(YDMODEL%YRCST, YDCPG_OPTS, YDMODEL%YRML_AOC%YRMCC, YDPHY, YDMODEL%YRML_PHY_MF%YRPHY1
    & YDCPG_BNDS%KFDIA, YSP_SBD%NLEVS, YDCPG_OPTS%ZDTPHY, ZTDS_TDTS, ZTDS_TDTP, ZTDS_TDWS, ZTDS_TDW
    & ZTDS_TDWP, ZTDS_TDWPI, ZTDS_TDWL, ZTDS_TDSNS, ZTDS_TDALBNS, ZTDS_TDRHONS, YDMF_PHYS_SURF%GSD_
    & YDMF_PHYS_SURF%GSD_VP%PWPC, YDMF_PHYS_SURF%GSD_VF%PLSM, YDMF_PHYS_SURF%GSD_VV%PIVEG, YDMF_PHY
    & YDMF_PHYS_SURF%GSP_SB%PT_T1, YDMF_PHYS_SURF%GSP_RR%PW_T1, YDMF_PHYS_SURF%GSP_RR%PIC_T1, YDMF_
    & YDMF_PHYS_SURF%GSP_SB%PTL_T1, YDMF_PHYS_SURF%GSP_RR%PFC_T1, YDMF_PHYS_SURF%GSP_SG%PF_T1, YDMF
    & YDMF_PHYS_SURF%GSP_SG%PR_T1                              )
  ELSE
    IF (YDCPG_OPTS%LCONFX) THEN
      DO JROF=YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
        YDMF_PHYS_SURF%GSP_RR%PT_T0(JROF)=YDCPG_GPAR%VTS(JROF)
      ENDDO
    ELSE
      DO JROF=YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
        YDMF_PHYS_SURF%GSP_RR%PT_T1(JROF)=YDCPG_GPAR%VTS(JROF)
      ENDDO
    ENDIF
  ENDIF
  IF(LNUDG)THEN
    CALL CPNUDG ( YDCPG_OPTS%KLON, YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA, NFNUDG, YDCPG_OPTS%KFLEVG, Y
    & XPNUDG, YDMF_PHYS_SURF%GSD_VF%PNUDM, YDMF_PHYS_SURF%GSP_RR%PT_T1, YDMF_PHYS_SURF%GSP_RR%PW_T1
    & YDMF_PHYS_SURF%GSP_SB%PQ_T1, YDMF_PHYS_SURF%GSP_SG%PF_T1, YDMF_PHYS_NEXT_STATE%T (:, 1:YDCPG_
    & YDMF_PHYS_NEXT_STATE%Q (:, 1:YDCPG_OPTS%KFLEVG), YDMF_PHYS_NEXT_STATE%U (:, 1:YDCPG_OPTS%KFLE
    & YDMF_PHYS_NEXT_STATE%V (:, 1:YDCPG_OPTS%KFLEVG), YDMF_PHYS_NEXT_STATE%SP, YDVARS%T%T0, YDVARS
    & YDVARS%U%T0, YDVARS%V%T0, YDCPG_PHY0%PREHYD(:, YDCPG_OPTS%KFLEVG), YDVARS%GEOMETRY%GM%T0, YDM
    &             )
  ENDIF
ENDIF
```

```
aplpar.F90                                                        4918,5         98%
```

```
CALL MF_PHYS_TRANSFER (YDCPG_BNDS, YDCPG_OPTS, YDVARS, YDMODEL%YRML_PHY_MF%YRPHY, YDMODEL%YRML_GCON

CALL APL_ARPEGE_SURFACE_UPDATE (YDCPG_BNDS, YDCPG_OPTS, YDCPG_GPAR, YDMF_PHYS, YDMF_PHYS_SURF, &
& YDMODEL, YDCPG_OPTS%LCONFX, YDCPG_OPTS%ZDTPHY, ZDSA_C1, ZDSA_C2, ZFLU_FEVI, ZFLU_VEG)
```

```
apl_arpege.F90                                                    895,0-1        93%
```

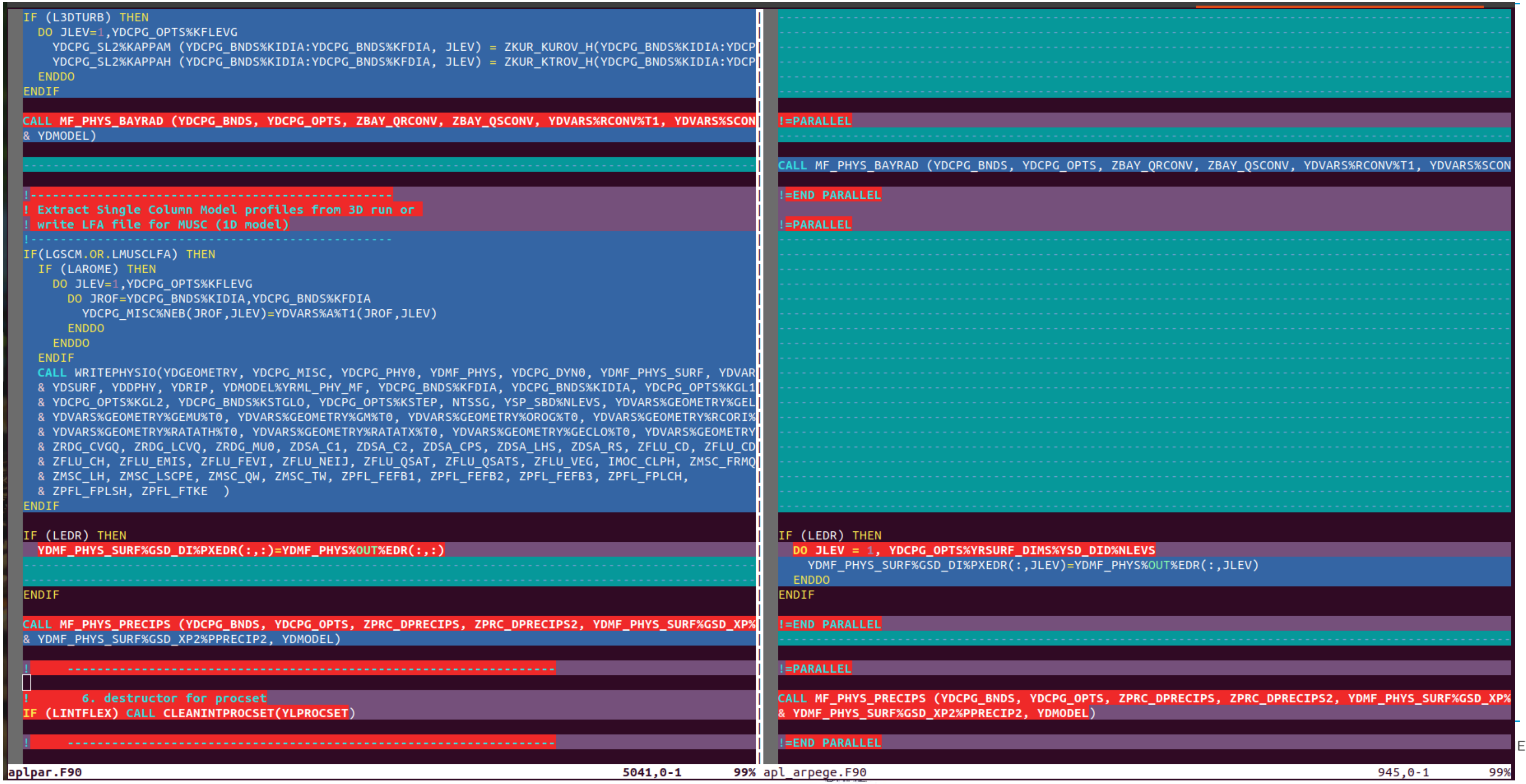
New version of APLPAR and APL_ARPEGE



```
    &                )
  ENDIF
ENDIF


IF(YDMODEL%YRML_PHY_MF%YRPHY%LCVPGY) THEN
  CALL MF_PHYS_CVV (YDCPG_BNDS, YDCPG_OPTS, YDVARS%CVV%T0, YDVARS%CVV%T1)
ENDIF

!        3.3  Store the model trajectory at t-dt (leap-frog) or t (sl2tl).
!             ------------------------------------------------------------

IF (LTRAJPS) THEN
  PTRAJ_PHYS%PQSSMF(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA)=YDCPG_MISC%QS(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KF
  PTRAJ_PHYS%PTSMF(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA) =YDMF_PHYS_BASE_STATE%YGSP_RR%T(YDCPG_BNDS%KI
  PTRAJ_PHYS%PSNSMF(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA)=YDMF_PHYS_BASE_STATE%YGSP_SG%F(YDCPG_BNDS%KI

  IF (.NOT. LTWOTL) THEN
    CALL WRPHTRAJM(YDGEOMETRY, YDSIMPHL, YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA, PTRAJ_PHYS, YDVARS%U%T
    & YDVARS%V%T9, YDVARS%T%T9, YDVARS%Q%T9, YDVARS%L%T9, YDVARS%I%T9, YDVARS%SP%T9)
  ENDIF

  IF (LPRTTRAJ.AND.PTRAJ_PHYS%LASTCHUNK) WRITE(NULOUT,*)'GREPTRAJ STORE TRAJ_PHYS in APLPAR'
ENDIF

!     -----------------------------------------------------------------


!*        5.    Final calculations.
!               -------------------

! * Restore the initial value of some pseudo-historical surface buffers
!   if relevant.
IF (LL_SAVE_PHSURF) THEN
  CALL MF_PHYS_SAVE_PHSURF_PART2 (YDCPG_BNDS, YDCPG_OPTS, ZSAV_DDAL, ZSAV_DDOM, ZSAV_ENTCH,
  & ZSAV_FHPS, ZSAV_GZ0F, ZSAV_GZ0HF, ZSAV_HV, ZSAV_PBLH, ZSAV_QSH, ZSAV_UDAL, ZSAV_UDGRO, ZSAV_UDO
  & ZSAV_UNEBH, YDMF_PHYS_SURF%GSD_VF%PZ0F, YDMF_PHYS_SURF%GSD_VH%PPBLH, YDMF_PHYS_SURF%GSD_VH%PQSH
  & YDMF_PHYS_SURF%GSD_VH%PSPSH, YDMF_PHYS_SURF%GSD_VK%PUDGRO, YDMF_PHYS_SURF%GSD_VV%PHV, YDMF_PHYS
  & YDVARS%DAL%T0, YDVARS%DOM%T0, YDVARS%UAL%T0, YDVARS%UEN%T0, YDVARS%UNEBH%T0, YDVARS%UOM%T0,
  & YDMODEL)

ENDIF

! Store horizontal exchange coefficients (3D turbulence) to SL2 buffers
IF (L3DTURB) THEN
  DO JLEV=1,YDCPG_OPTS%KFLEVG
aplpar.F90                                                  4995,3         99%
```

```

CALL APL_ARPEGE_SURFACE_UPDATE (YDCPG_BNDS, YDCPG_OPTS, YDCPG_GPAR, YDMF_PHYS, YDMF_PHYS_SURF, &
& YDMODEL, YDCPG_OPTS%LCONFX, YDCPG_OPTS%ZDTPHY, ZDSA_C1, ZDSA_C2, ZFLU_FEVI, ZFLU_VEG)

IF(YDMODEL%YRML_PHY_MF%YRPHY%LCVPGY) THEN

!=PARALLEL

  CALL MF_PHYS_CVV (YDCPG_BNDS, YDCPG_OPTS, YDVARS%CVV%T0, YDVARS%CVV%T1)

!=END PARALLEL

ENDIF

! Restore the initial value of some pseudo-historical surface buffers if relevant.

IF (YDCPG_OPTS%LCONFX) THEN

!=PARALLEL

  CALL MF_PHYS_SAVE_PHSURF_PART2 (YDCPG_BNDS, YDCPG_OPTS, ZSAV_DDAL, ZSAV_DDOM, ZSAV_ENTCH,
  & ZSAV_FHPS, ZSAV_GZ0F, ZSAV_GZ0HF, ZSAV_HV, ZSAV_PBLH, ZSAV_QSH, ZSAV_UDAL, ZSAV_UDGRO, ZSAV_UDO
  & ZSAV_UNEBH, YDMF_PHYS_SURF%GSD_VF%PZ0F, YDMF_PHYS_SURF%GSD_VH%PPBLH, YDMF_PHYS_SURF%GSD_VH%PQSH
  & YDMF_PHYS_SURF%GSD_VH%PSPSH, YDMF_PHYS_SURF%GSD_VK%PUDGRO, YDMF_PHYS_SURF%GSD_VV%PHV, YDMF_PHYS
  & YDVARS%DAL%T0, YDVARS%DOM%T0, YDVARS%UAL%T0, YDVARS%UEN%T0, YDVARS%UNEBH%T0, YDVARS%UOM%T0,
  & YDMODEL)

!=END PARALLEL

ENDIF

! Store horizontal exchange coefficients (3D turbulence) to SL2 buffers

apl_arpege.F90                                                  920,3         96%
```

New version of APLPAR and APL_ARPEGE

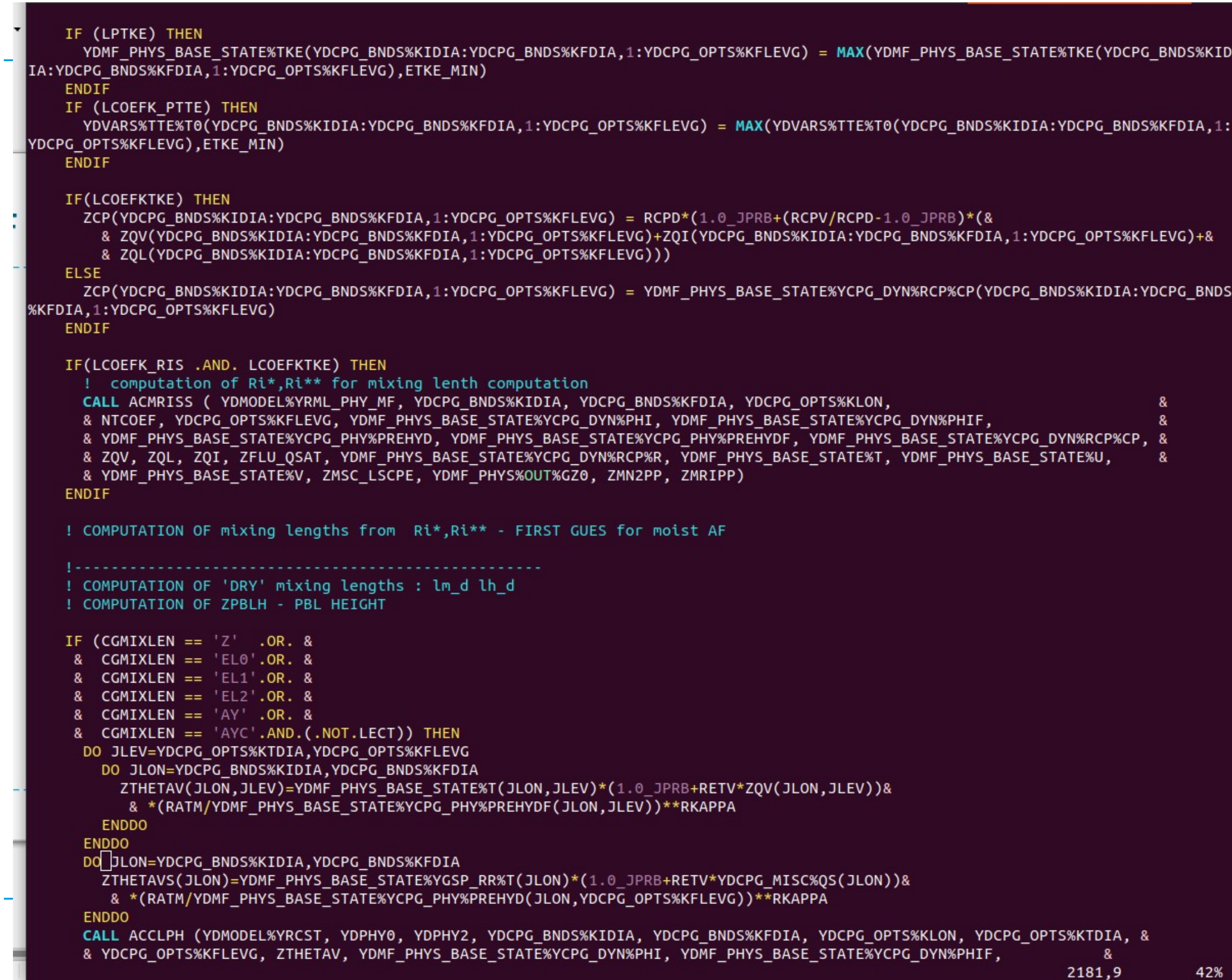


```
IF (L3DTURB) THEN
  DO JLEV=1,YDCPG_OPTS%KFLEVG
    YDCPG_SL2%KAPPAM (YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA, JLEV) = ZKUR_KUROV_H(YDCPG_BNDS%KIDIA:YDCP
    YDCPG_SL2%KAPPAH (YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA, JLEV) = ZKUR_KTROV_H(YDCPG_BNDS%KIDIA:YDCP
  ENDDO
ENDIF

CALL MF_PHYS_BAYRAD (YDCPG_BNDS, YDCPG_OPTS, ZBAY_QRCONV, ZBAY_QSCONV, YDVARS%RCONV%T1, YDVARS%SCON
& YDMODEL)


!-------------------------------------------------
! Extract Single Column Model profiles from 3D run or
! write LFA file for MUSC (1D model)
!-------------------------------------------------
IF(LGSCM.OR.LMUSCLFA) THEN
  IF (LAROME) THEN
    DO JLEV=1,YDCPG_OPTS%KFLEVG
      DO JROF=YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
        YDCPG_MISC%NEB(JROF,JLEV)=YDVARS%A%T1(JROF,JLEV)
      ENDDO
    ENDDO
  ENDIF
  CALL WRITEPHYSIO(YDGEOMETRY, YDCPG_MISC, YDCPG_PHY0, YDMF_PHYS, YDCPG_DYN0, YDMF_PHYS_SURF, YDVAR
  & YDSURF, YDDPHY, YDRIP, YDMODEL%YRML_PHY_MF, YDCPG_BNDS%KFDIA, YDCPG_BNDS%KIDIA, YDCPG_OPTS%KGL1
  & YDCPG_OPTS%KGL2, YDCPG_BNDS%KSTGLO, YDCPG_OPTS%KSTEP, NTSSG, YSP_SBD%NLEVS, YDVARS%GEOMETRY%GEL
  & YDVARS%GEOMETRY%GEMU%T0, YDVARS%GEOMETRY%GM%T0, YDVARS%GEOMETRY%OROG%T0, YDVARS%GEOMETRY%RCORI%
  & YDVARS%GEOMETRY%RATATH%T0, YDVARS%GEOMETRY%RATATX%T0, YDVARS%GEOMETRY%GECLO%T0, YDVARS%GEOMETRY
  & ZRDG_CVGQ, ZRDG_LCVQ, ZRDG_MU0, ZDSA_C1, ZDSA_C2, ZDSA_CPS, ZDSA_LHS, ZDSA_RS, ZFLU_CD, ZFLU_CD
  & ZFLU_CH, ZFLU_EMIS, ZFLU_FEVI, ZFLU_NEIJ, ZFLU_QSAT, ZFLU_QSATS, ZFLU_VEG, IMOC_CLPH, ZMSC_FRMQ
  & ZMSC_LH, ZMSC_LSCPE, ZMSC_QW, ZMSC_TW, ZPFL_FEFB1, ZPFL_FEFB2, ZPFL_FEFB3, ZPFL_FPLCH,
  & ZPFL_FPLSH, ZPFL_FTKE  )
ENDIF

IF (LEDR) THEN
  YDMF_PHYS_SURF%GSD_DI%PXEDR(:,:)=YDMF_PHYS%OUT%EDR(:,:)

ENDIF

CALL MF_PHYS_PRECIPS (YDCPG_BNDS, YDCPG_OPTS, ZPRC_DPRECIPS, ZPRC_DPRECIPS2, YDMF_PHYS_SURF%GSD_XP%
& YDMF_PHYS_SURF%GSD_XP2%PPRECIP2, YDMODEL)

!     ------------------------------------------------------------------


!       6. destructor for procset
IF (LINTFLEX) CALL CLEANINTPROCSET(YLPROCSET)

!     ------------------------------------------------------------------

aplpar.F90                                                     5041,0-1       99%
```

```
!=PARALLEL

CALL MF_PHYS_BAYRAD (YDCPG_BNDS, YDCPG_OPTS, ZBAY_QRCONV, ZBAY_QSCONV, YDVARS%RCONV%T1, YDVARS%SCON

!=END PARALLEL

!=PARALLEL

IF (LEDR) THEN
  DO JLEV = 1, YDCPG_OPTS%YRSURF_DIMS%YSD_DID%NLEVS
    YDMF_PHYS_SURF%GSD_DI%PXEDR(:,JLEV)=YDMF_PHYS%OUT%EDR(:,JLEV)
  ENDDO
ENDIF

!=END PARALLEL

!=PARALLEL

CALL MF_PHYS_PRECIPS (YDCPG_BNDS, YDCPG_OPTS, ZPRC_DPRECIPS, ZPRC_DPRECIPS2, YDMF_PHYS_SURF%GSD_XP%
& YDMF_PHYS_SURF%GSD_XP2%PPRECIP2, YDMODEL)

!=END PARALLEL

apl_arpege.F90                                                  945,0-1        99%
```

New version of APLPAR

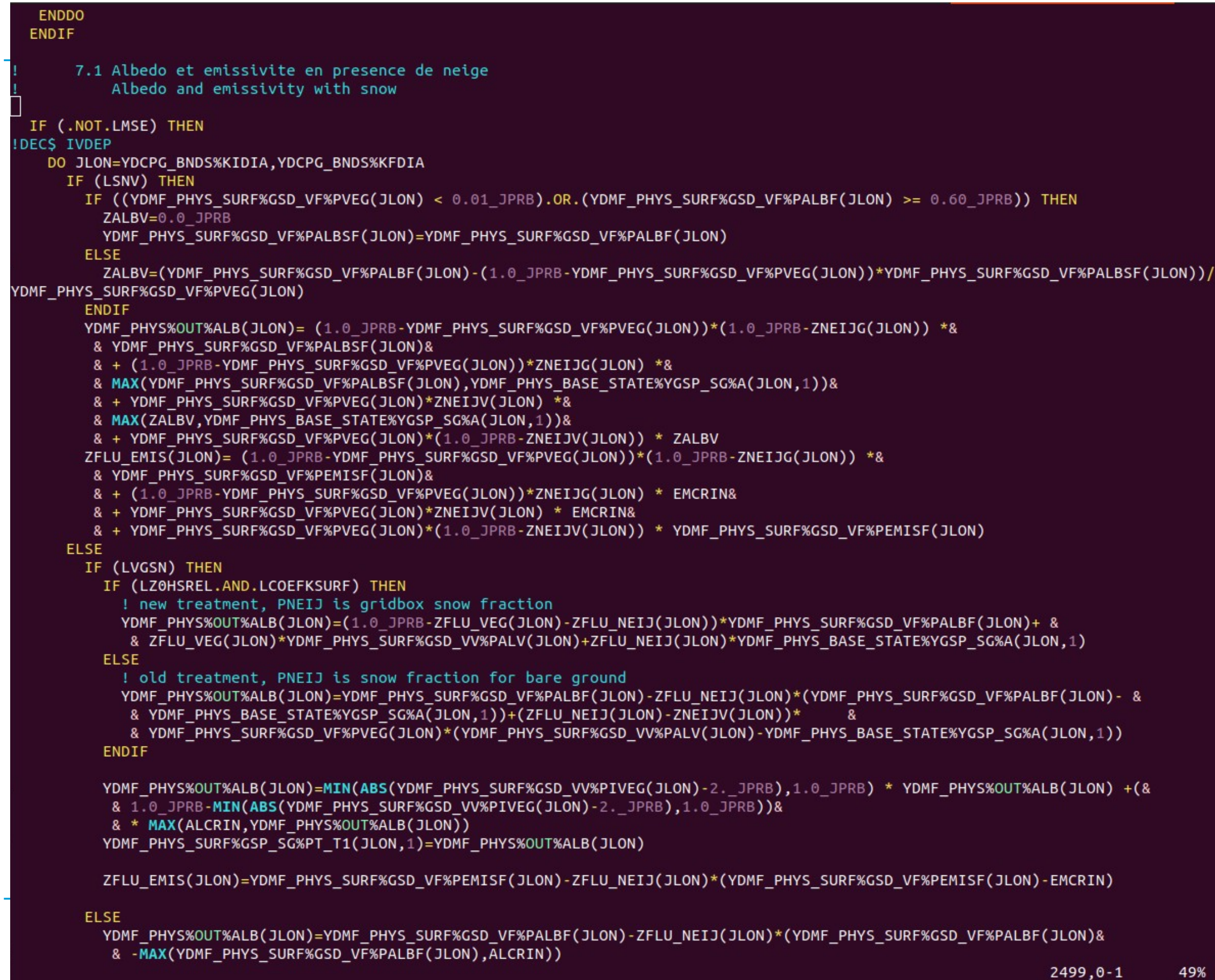
Put this part of aplpar
Into a subroutine ...



```
    IF (LPTKE) THEN
      YDMF_PHYS_BASE_STATE%TKE(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA,1:YDCPG_OPTS%KFLEVG) = MAX(YDMF_PHYS_BASE_STATE%TKE(YDCPG_BNDS%KID
IA:YDCPG_BNDS%KFDIA,1:YDCPG_OPTS%KFLEVG),ETKE_MIN)
    ENDIF
    IF (LCOEFK_PTTE) THEN
      YDVARS%TTE%T0(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA,1:YDCPG_OPTS%KFLEVG) = MAX(YDVARS%TTE%T0(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA,1:
YDCPG_OPTS%KFLEVG),ETKE_MIN)
    ENDIF

    IF(LCOEFKTKE) THEN
      ZCP(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA,1:YDCPG_OPTS%KFLEVG) = RCPD*(1.0_JPRB+(RCPV/RCPD-1.0_JPRB)*(&
        & ZQV(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA,1:YDCPG_OPTS%KFLEVG)+ZQI(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA,1:YDCPG_OPTS%KFLEVG)+&
        & ZQL(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA,1:YDCPG_OPTS%KFLEVG)))
    ELSE
      ZCP(YDCPG_BNDS%KIDIA:YDCPG_BNDS%KFDIA,1:YDCPG_OPTS%KFLEVG) = YDMF_PHYS_BASE_STATE%YCPG_DYN%RCP%CP(YDCPG_BNDS%KIDIA:YDCPG_BNDS
%KFDIA,1:YDCPG_OPTS%KFLEVG)
    ENDIF

    IF(LCOEFK_RIS .AND. LCOEFKTKE) THEN
      !  computation of Ri*,Ri** for mixing lenth computation
      CALL ACMRISS ( YDMODEL%YRML_PHY_MF, YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA, YDCPG_OPTS%KLON,                              &
      & NTCOEF, YDCPG_OPTS%KFLEVG, YDMF_PHYS_BASE_STATE%YCPG_DYN%PHI, YDMF_PHYS_BASE_STATE%YCPG_DYN%PHIF,                    &
      & YDMF_PHYS_BASE_STATE%YCPG_PHY%PREHYD, YDMF_PHYS_BASE_STATE%YCPG_PHY%PREHYDF, YDMF_PHYS_BASE_STATE%YCPG_DYN%RCP%CP, &
      & ZQV, ZQL, ZQI, ZFLU_QSAT, YDMF_PHYS_BASE_STATE%YCPG_DYN%RCP%R, YDMF_PHYS_BASE_STATE%T, YDMF_PHYS_BASE_STATE%U,     &
      & YDMF_PHYS_BASE_STATE%V, ZMSC_LSCPE, YDMF_PHYS%OUT%GZ0, ZMN2PP, ZMRIPP)
    ENDIF

    ! COMPUTATION OF mixing lengths from  Ri*,Ri** - FIRST GUES for moist AF

    !----------------------------------------------------
    ! COMPUTATION OF 'DRY' mixing lengths : lm_d lh_d
    ! COMPUTATION OF ZPBLH - PBL HEIGHT

    IF (CGMIXLEN == 'Z'  .OR. &
     &  CGMIXLEN == 'EL0'.OR. &
     &  CGMIXLEN == 'EL1'.OR. &
     &  CGMIXLEN == 'EL2'.OR. &
     &  CGMIXLEN == 'AY' .OR. &
     &  CGMIXLEN == 'AYC'.AND.(.NOT.LECT)) THEN
      DO JLEV=YDCPG_OPTS%KTDIA,YDCPG_OPTS%KFLEVG
        DO JLON=YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
          ZTHETAV(JLON,JLEV)=YDMF_PHYS_BASE_STATE%T(JLON,JLEV)*(1.0_JPRB+RETV*ZQV(JLON,JLEV))&
           & *(RATM/YDMF_PHYS_BASE_STATE%YCPG_PHY%PREHYDF(JLON,JLEV))**RKAPPA
        ENDDO
      ENDDO
      DO JLON=YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
        ZTHETAVS(JLON)=YDMF_PHYS_BASE_STATE%YGSP_RR%T(JLON)*(1.0_JPRB+RETV*YDCPG_MISC%QS(JLON))&
         & *(RATM/YDMF_PHYS_BASE_STATE%YCPG_PHY%PREHYD(JLON,YDCPG_OPTS%KFLEVG))**RKAPPA
      ENDDO
      CALL ACCLPH (YDMODEL%YRCST, YDPHY0, YDPHY2, YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA, YDCPG_OPTS%KLON, YDCPG_OPTS%KTDIA, &
      & YDCPG_OPTS%KFLEVG, ZTHETAV, YDMF_PHYS_BASE_STATE%YCPG_DYN%PHI, YDMF_PHYS_BASE_STATE%YCPG_DYN%PHIF,            &
                                                                                                        2181,9        42%
```

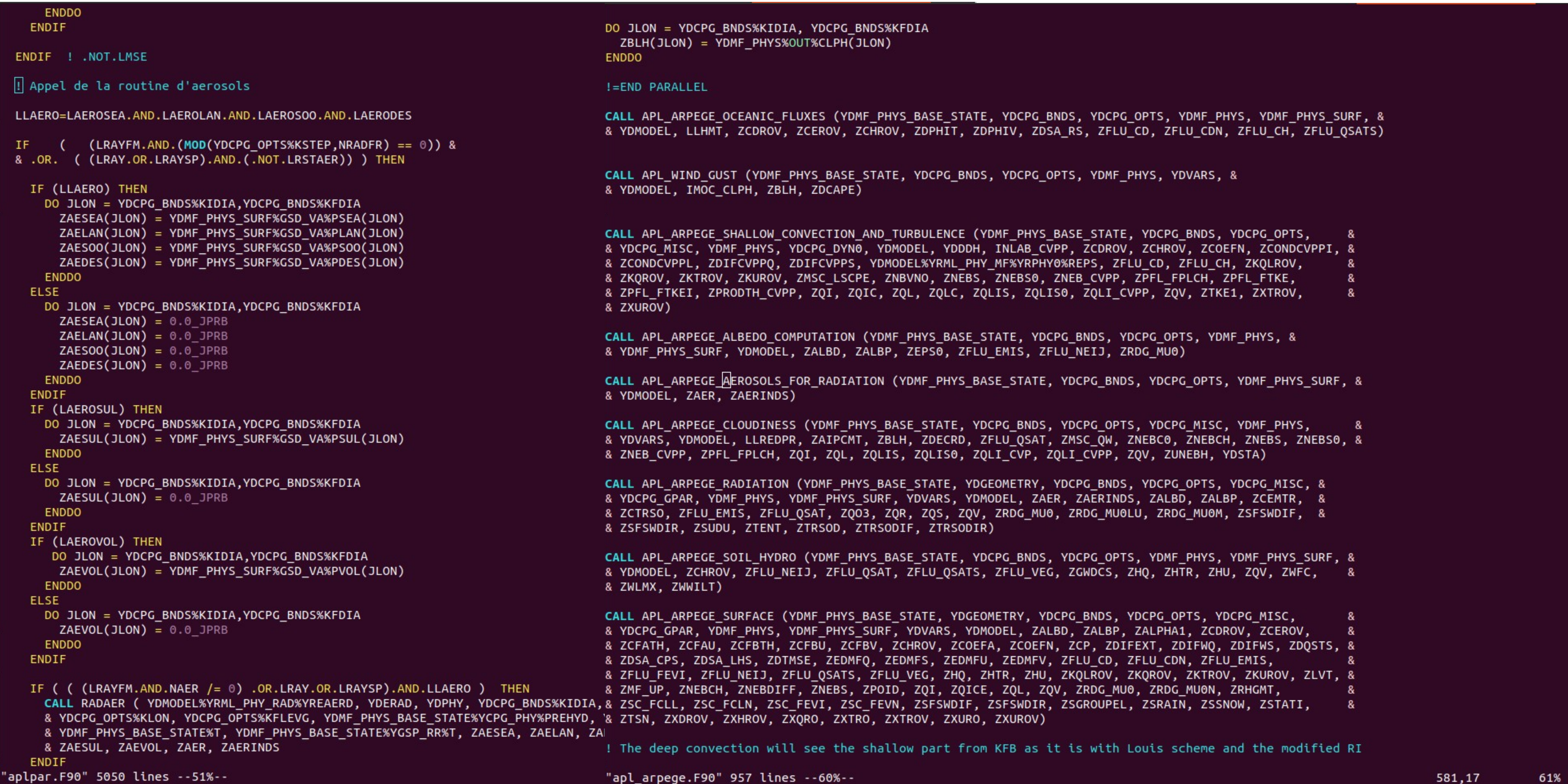
New version of APLPAR

And this one too



```
   ENDDO
  ENDIF

!      7.1 Albedo et emissivite en presence de neige
!          Albedo and emissivity with snow

  IF (.NOT.LMSE) THEN
!DEC$ IVDEP
    DO JLON=YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
      IF (LSNV) THEN
        IF ((YDMF_PHYS_SURF%GSD_VF%PVEG(JLON) < 0.01_JPRB).OR.(YDMF_PHYS_SURF%GSD_VF%PALBF(JLON) >= 0.60_JPRB)) THEN
          ZALBV=0.0_JPRB
          YDMF_PHYS_SURF%GSD_VF%PALBSF(JLON)=YDMF_PHYS_SURF%GSD_VF%PALBF(JLON)
        ELSE
          ZALBV=(YDMF_PHYS_SURF%GSD_VF%PALBF(JLON)-(1.0_JPRB-YDMF_PHYS_SURF%GSD_VF%PVEG(JLON))*YDMF_PHYS_SURF%GSD_VF%PALBSF(JLON))/
YDMF_PHYS_SURF%GSD_VF%PVEG(JLON)
        ENDIF
        YDMF_PHYS%OUT%ALB(JLON)= (1.0_JPRB-YDMF_PHYS_SURF%GSD_VF%PVEG(JLON))*(1.0_JPRB-ZNEIJG(JLON)) *&
         & YDMF_PHYS_SURF%GSD_VF%PALBSF(JLON)&
         & + (1.0_JPRB-YDMF_PHYS_SURF%GSD_VF%PVEG(JLON))*ZNEIJG(JLON) *&
         & MAX(YDMF_PHYS_SURF%GSD_VF%PALBSF(JLON),YDMF_PHYS_BASE_STATE%YGSP_SG%A(JLON,1))&
         & + YDMF_PHYS_SURF%GSD_VF%PVEG(JLON)*ZNEIJV(JLON) *&
         & MAX(ZALBV,YDMF_PHYS_BASE_STATE%YGSP_SG%A(JLON,1))&
         & + YDMF_PHYS_SURF%GSD_VF%PVEG(JLON)*(1.0_JPRB-ZNEIJV(JLON)) * ZALBV
        ZFLU_EMIS(JLON)= (1.0_JPRB-YDMF_PHYS_SURF%GSD_VF%PVEG(JLON))*(1.0_JPRB-ZNEIJG(JLON)) *&
         & YDMF_PHYS_SURF%GSD_VF%PEMISF(JLON)&
         & + (1.0_JPRB-YDMF_PHYS_SURF%GSD_VF%PVEG(JLON))*ZNEIJG(JLON) * EMCRIN&
         & + YDMF_PHYS_SURF%GSD_VF%PVEG(JLON)*ZNEIJV(JLON) * EMCRIN&
         & + YDMF_PHYS_SURF%GSD_VF%PVEG(JLON)*(1.0_JPRB-ZNEIJV(JLON)) * YDMF_PHYS_SURF%GSD_VF%PEMISF(JLON)
      ELSE
        IF (LVGSN) THEN
          IF (LZ0HSREL.AND.LCOEFKSURF) THEN
            ! new treatment, PNEIJ is gridbox snow fraction
            YDMF_PHYS%OUT%ALB(JLON)=(1.0_JPRB-ZFLU_VEG(JLON)-ZFLU_NEIJ(JLON))*YDMF_PHYS_SURF%GSD_VF%PALBF(JLON)+ &
             & ZFLU_VEG(JLON)*YDMF_PHYS_SURF%GSD_VV%PALV(JLON)+ZFLU_NEIJ(JLON)*YDMF_PHYS_BASE_STATE%YGSP_SG%A(JLON,1)
          ELSE
            ! old treatment, PNEIJ is snow fraction for bare ground
            YDMF_PHYS%OUT%ALB(JLON)=YDMF_PHYS_SURF%GSD_VF%PALBF(JLON)-ZFLU_NEIJ(JLON)*(YDMF_PHYS_SURF%GSD_VF%PALBF(JLON)- &
             & YDMF_PHYS_BASE_STATE%YGSP_SG%A(JLON,1))+(ZFLU_NEIJ(JLON)-ZNEIJV(JLON))*     &
             & YDMF_PHYS_SURF%GSD_VF%PVEG(JLON)*(YDMF_PHYS_SURF%GSD_VV%PALV(JLON)-YDMF_PHYS_BASE_STATE%YGSP_SG%A(JLON,1))
          ENDIF

          YDMF_PHYS%OUT%ALB(JLON)=MIN(ABS(YDMF_PHYS_SURF%GSD_VV%PIVEG(JLON)-2._JPRB),1.0_JPRB) * YDMF_PHYS%OUT%ALB(JLON) +(&
           & 1.0_JPRB-MIN(ABS(YDMF_PHYS_SURF%GSD_VV%PIVEG(JLON)-2._JPRB),1.0_JPRB))&
           & * MAX(ALCRIN,YDMF_PHYS%OUT%ALB(JLON))
          YDMF_PHYS_SURF%GSP_SG%PT_T1(JLON,1)=YDMF_PHYS%OUT%ALB(JLON)

          ZFLU_EMIS(JLON)=YDMF_PHYS_SURF%GSD_VF%PEMISF(JLON)-ZFLU_NEIJ(JLON)*(YDMF_PHYS_SURF%GSD_VF%PEMISF(JLON)-EMCRIN)

        ELSE
          YDMF_PHYS%OUT%ALB(JLON)=YDMF_PHYS_SURF%GSD_VF%PALBF(JLON)-ZFLU_NEIJ(JLON)*(YDMF_PHYS_SURF%GSD_VF%PALBF(JLON)&
           & -MAX(YDMF_PHYS_SURF%GSD_VF%PALBF(JLON),ALCRIN))
                                                                                                   2499,0-1        49%
```

New version of APLPAR



```
    ENDDO
  ENDIF

ENDIF  ! .NOT.LMSE

! Appel de la routine d'aerosols

LLAERO=LAEROSEA.AND.LAEROLAN.AND.LAEROSOO.AND.LAERODES

IF    (   (LRAYFM.AND.(MOD(YDCPG_OPTS%KSTEP,NRADFR) == 0)) &
& .OR.  ( (LRAY.OR.LRAYSP).AND.(.NOT.LRSTAER)) ) THEN

  IF (LLAERO) THEN
    DO JLON = YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
      ZAESEA(JLON) = YDMF_PHYS_SURF%GSD_VA%PSEA(JLON)
      ZAELAN(JLON) = YDMF_PHYS_SURF%GSD_VA%PLAN(JLON)
      ZAESOO(JLON) = YDMF_PHYS_SURF%GSD_VA%PSOO(JLON)
      ZAEDES(JLON) = YDMF_PHYS_SURF%GSD_VA%PDES(JLON)
    ENDDO
  ELSE
    DO JLON = YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
      ZAESEA(JLON) = 0.0_JPRB
      ZAELAN(JLON) = 0.0_JPRB
      ZAESOO(JLON) = 0.0_JPRB
      ZAEDES(JLON) = 0.0_JPRB
    ENDDO
  ENDIF
  IF (LAEROSUL) THEN
    DO JLON = YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
      ZAESUL(JLON) = YDMF_PHYS_SURF%GSD_VA%PSUL(JLON)
    ENDDO
  ELSE
    DO JLON = YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
      ZAESUL(JLON) = 0.0_JPRB
    ENDDO
  ENDIF
  IF (LAEROVOL) THEN
     DO JLON = YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
      ZAEVOL(JLON) = YDMF_PHYS_SURF%GSD_VA%PVOL(JLON)
    ENDDO
  ELSE
    DO JLON = YDCPG_BNDS%KIDIA,YDCPG_BNDS%KFDIA
      ZAEVOL(JLON) = 0.0_JPRB
    ENDDO
  ENDIF

  IF ( ( (LRAYFM.AND.NAER /= 0) .OR.LRAY.OR.LRAYSP).AND.LLAERO )  THEN
    CALL RADAER ( YDMODEL%YRML_PHY_RAD%YREAERD, YDERAD, YDPHY, YDCPG_BNDS%KIDIA,
    & YDCPG_OPTS%KLON, YDCPG_OPTS%KFLEVG, YDMF_PHYS_BASE_STATE%YCPG_PHY%PREHYD, '
    & YDMF_PHYS_BASE_STATE%T, YDMF_PHYS_BASE_STATE%YGSP_RR%T, ZAESEA, ZAELAN, ZA
    & ZAESUL, ZAEVOL, ZAER, ZAERINDS
  ENDIF
"aplpar.F90" 5050 lines --51%--
```

```
DO JLON = YDCPG_BNDS%KIDIA, YDCPG_BNDS%KFDIA
  ZBLH(JLON) = YDMF_PHYS%OUT%CLPH(JLON)
ENDDO

!=END PARALLEL

CALL APL_ARPEGE_OCEANIC_FLUXES (YDMF_PHYS_BASE_STATE, YDCPG_BNDS, YDCPG_OPTS, YDMF_PHYS, YDMF_PHYS_SURF, &
& YDMODEL, LLHMT, ZCDROV, ZCEROV, ZCHROV, ZDPHIT, ZDPHIV, ZDSA_RS, ZFLU_CD, ZFLU_CDN, ZFLU_CH, ZFLU_QSATS)


CALL APL_WIND_GUST (YDMF_PHYS_BASE_STATE, YDCPG_BNDS, YDCPG_OPTS, YDMF_PHYS, YDVARS, &
& YDMODEL, IMOC_CLPH, ZBLH, ZDCAPE)


CALL APL_ARPEGE_SHALLOW_CONVECTION_AND_TURBULENCE (YDMF_PHYS_BASE_STATE, YDCPG_BNDS, YDCPG_OPTS,     &
& YDCPG_MISC, YDMF_PHYS, YDCPG_DYN0, YDMODEL, YDDDH, INLAB_CVPP, ZCDROV, ZCHROV, ZCOEFN, ZCONDCVPPI, &
& ZCONDCVPPL, ZDIFCVPPQ, ZDIFCVPPS, YDMODEL%YRML_PHY_MF%YRPHY0%REPS, ZFLU_CD, ZFLU_CH, ZKQLROV,      &
& ZKQROV, ZKTROV, ZKUROV, ZMSC_LSCPE, ZNBVNO, ZNEBS, ZNEBS0, ZNEB_CVPP, ZPFL_FPLCH, ZPFL_FTKE,       &
& ZPFL_FTKEI, ZPRODTH_CVPP, ZQI, ZQIC, ZQL, ZQLC, ZQLIS, ZQLIS0, ZQLI_CVPP, ZQV, ZTKE1, ZXTROV,      &
& ZXUROV)

CALL APL_ARPEGE_ALBEDO_COMPUTATION (YDMF_PHYS_BASE_STATE, YDCPG_BNDS, YDCPG_OPTS, YDMF_PHYS, &
& YDMF_PHYS_SURF, YDMODEL, ZALBD, ZALBP, ZEPS0, ZFLU_EMIS, ZFLU_NEIJ, ZRDG_MU0)

CALL APL_ARPEGE_AEROSOLS_FOR_RADIATION (YDMF_PHYS_BASE_STATE, YDCPG_BNDS, YDCPG_OPTS, YDMF_PHYS_SURF, &
& YDMODEL, ZAER, ZAERINDS)

CALL APL_ARPEGE_CLOUDINESS (YDMF_PHYS_BASE_STATE, YDCPG_BNDS, YDCPG_OPTS, YDCPG_MISC, YDMF_PHYS,      &
& YDVARS, YDMODEL, LLREDPR, ZAIPCMT, ZBLH, ZDECRD, ZFLU_QSAT, ZMSC_QW, ZNEBC0, ZNEBCH, ZNEBS, ZNEBS0, &
& ZNEB_CVPP, ZPFL_FPLCH, ZQI, ZQL, ZQLIS, ZQLIS0, ZQLI_CVP, ZQLI_CVPP, ZQV, ZUNEBH, YDSTA)

CALL APL_ARPEGE_RADIATION (YDMF_PHYS_BASE_STATE, YDGEOMETRY, YDCPG_BNDS, YDCPG_OPTS, YDCPG_MISC, &
& YDCPG_GPAR, YDMF_PHYS, YDMF_PHYS_SURF, YDVARS, YDMODEL, ZAER, ZAERINDS, ZALBD, ZALBP, ZCEMTR,  &
& ZCTRSO, ZFLU_EMIS, ZFLU_QSAT, ZQO3, ZQR, ZQS, ZQV, ZRDG_MU0, ZRDG_MU0LU, ZRDG_MU0M, ZSFSWDIF,  &
& ZSFSWDIR, ZSUDU, ZTENT, ZTRSOD, ZTRSODIF, ZTRSODIR)

CALL APL_ARPEGE_SOIL_HYDRO (YDMF_PHYS_BASE_STATE, YDCPG_BNDS, YDCPG_OPTS, YDMF_PHYS, YDMF_PHYS_SURF, &
& YDMODEL, ZCHROV, ZFLU_NEIJ, ZFLU_QSAT, ZFLU_QSATS, ZFLU_VEG, ZGWDCS, ZHQ, ZHTR, ZHU, ZQV, ZWFC,    &
& ZWLMX, ZWWILT)

CALL APL_ARPEGE_SURFACE (YDMF_PHYS_BASE_STATE, YDGEOMETRY, YDCPG_BNDS, YDCPG_OPTS, YDCPG_MISC,       &
& YDCPG_GPAR, YDMF_PHYS, YDMF_PHYS_SURF, YDVARS, YDMODEL, ZALBD, ZALBP, ZALPHA1, ZCDROV, ZCEROV,     &
& ZCFATH, ZCFAU, ZCFBTH, ZCFBU, ZCFBV, ZCHROV, ZCOEFA, ZCOEFN, ZCP, ZDIFEXT, ZDIFWQ, ZDIFWS, ZDQSTS, &
& ZDSA_CPS, ZDSA_LHS, ZDTMSE, ZEDMFQ, ZEDMFS, ZEDMFU, ZEDMFV, ZFLU_CD, ZFLU_CDN, ZFLU_EMIS,          &
& ZFLU_FEVI, ZFLU_NEIJ, ZFLU_QSATS, ZFLU_VEG, ZHQ, ZHTR, ZHU, ZKQLROV, ZKQROV, ZKTROV, ZKUROV, ZLVT, &
& ZMF_UP, ZNEBCH, ZNEBDIFF, ZNEBS, ZPOID, ZQI, ZQICE, ZQL, ZQV, ZRDG_MU0, ZRDG_MU0N, ZRHGMT,         &
& ZSC_FCLL, ZSC_FCLN, ZSC_FEVI, ZSC_FEVN, ZSFSWDIF, ZSFSWDIR, ZSGROUPEL, ZSRAIN, ZSSNOW, ZSTATI,     &
& ZTSN, ZXDROV, ZXHROV, ZXQRO, ZXTRO, ZXTROV, ZXURO, ZXUROV)

! The deep convection will see the shallow part from KFB as it is with Louis scheme and the modified RI

"apl_arpege.F90" 957 lines --60%--                                                  581,17          61%
```

APLPAR split to do for ALARO

APLPAR is still there!
Initial step can be done automatically with a namelist provided
BUT
- we use multiple physics options operationally (A-LAEF)
- we want to leave some options (pTKE)
- can a ‘namelist’ with all usefull switches on (that would never work for running) be used?
- after the automatic step, still lot of work to do manually

We also need an ‘init’ routine (and other helper type routines)

Do we want to create apl_alaro_turb, apl_alaro_deep_cnv …
- at this point?
- maybe later?
- different answer for different parts

CPG and CPG_DRV refactoring

- allows different parts of CPG to be called in different OpenMP loops. Currently:

```
!$OMP PARALLEL DO
DO JBLK=1,NBLK
CALL CPG_GP
CALL MF_PHYS
CALL CPG_DIA
CALL CPG_DYN
CALL CPG_END
ENDDO
!$OMP PARALLEL DO
```

```
!$OMP PARALLEL DO
DO JBLK=1,NBLK
CALL CPG_GP
ENDDO
!$OMP PARALLEL DO
DO JBLK=1,NBLK
CALL MF_PHYS
ENDDO
!$OMP PARALLEL DO
DO JBLK=1,NBLK
CALL CPG_DIA
CALL CPG_DYN
CALL CPG_END
ENDDO
```

CPG and CPG_DRV refactoring

CPG gets an argument that defines the configuration.
This argument defines
 - if the different parts are executed in a single call to cpg
 - or in separate subsequient calls to CPG.
This allows the decsion on which loop structure to use at runtime.

```
!$OMP PARALLEL DO
DO JKGLO = 1, NGPTOT, NPROMA
! cpg_gp.F90
CALL CPG (..., CDPART=”X00”)
ENDDO

!$OMP PARALLEL DO
DO JKGLO = 1, NGPTOT, NPROMA
! mf_phys.F90
CALL CPG (..., CDPART=”0X0”)
ENDDO

!$OMP PARALLEL DO
DO JKGLO = 1, NGPTOT, NPROMA
! cpg_dia/dyn/end.F90
CALL CPG (..., CDPART=”00X”)
ENDDO
```

CPG and CPG_DRV refactoring

CPG gets an argument that defines the configuration.

This argument defines

- if the different parts are executed in a single call to cpg
- or in separate subsequient calls to CPG.

This allows the decsion on which loop structure to use at runtime.

```
!$OMP PARALLEL DO
DO JKGLO = 1, NGPTOT, NPROMA
! cpg_gp.F90
CALL CPG (..., CDPART=”X00”)
ENDDO

!$OMP PARALLEL DO
DO JKGLO = 1, NGPTOT, NPROMA
! mf_phys.F90
CALL CPG (..., CDPART=”0X0”)
ENDDO

!$OMP PARALLEL DO
DO JKGLO = 1, NGPTOT, NPROMA
! cpg_dia/dyn/end.F90
CALL CPG (..., CDPART=”00X”)
ENDDO
```

Memory consumption increase

When splitting a loop any variable that is passed between the different parts must be allocated with an extra dimension. In the example below, when a loop is split, it is necessary to make X an array. This increases memory consumption.

```
REAL :: X
REAL :: Y(NBLK)
DO JBLK=1,NBLK
X=JBLK
Y(JBLK)=X
ENDDO
```

```
REAL :: X(NBLK)
REAL :: Y(NBLK)
DO JBLK=1,NBLK
X(JBLK)=JBLK
ENDDO
DO IBLK=1,NBLK
Y(JBLK)=X(JBLK)
ENDDO
```

Discussion

mf_phys and apl_alaro

ECMWF moved