EXAMPLE OF DIFFICULTIES ENCOUNTERED DURING MERGING DECKS.

YESSAD K. (METEO-FRANCE/CNRM/GMAP/ALGO)

August 10, 2010

∗ Introduction:

We give some examples of source merging exercices, showing potential difficulties. Some cases will be easy to solve, some other ones more difficult. Exercices generally use routine TRACARELA (excerp of routine TRACARE, where we have kept only the calculation of PSLAR).
Content of TRACARELA is:

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                       C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE               C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

∗ Example 1:

This is a case where the merging will be easy without any conflict.

```
Contributor 1 renames array ZA into ZZA.

SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                        C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE     C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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


IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO

ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO

ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

```
Contributor 2 adds some printings: print PSLAR(1).

SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                        C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE     C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

USE PARKIND1  ,ONLY : JPIM     ,JPRB
USE YOMHOOK   ,ONLY : LHOOK,   DR_HOOK
USE YOMLUN    ,ONLY : NULOUT

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

An automatic source merging will give the following result, without any conflict.

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                        C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE     C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

USE PARKIND1  ,ONLY : JPIM     ,JPRB
USE YOMHOOK   ,ONLY : LHOOK,   DR_HOOK
USE YOMLUN    ,ONLY : NULOUT

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

∗ Example 2:

This is a case where an automatic merging will be possible but not giving the right result: an additional manual modification is necessary.

```
Contributor 1 renames array PSLAC into PSLACE.

SUBROUTINE TRACARELA(PSLACE,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                        C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE                C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLACE(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLACE(JL)*PSLACE(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLACE(JL))
    ZB=ZC2P1*PSLACE(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLACE(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLACE(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO




ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

```
Contributor 2 adds a new case KTYP=3

SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                        C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE                C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 3) THEN
  DO JL=1,KLEN
    PSLAR(JL)=PSLAC(JL)
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

An automatic source merging will give the following result, without any conflict.

```
SUBROUTINE TRACARELA(PSLACE,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                       C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE               C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLACE(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLACE(JL)*PSLACE(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLACE(JL))
    ZB=ZC2P1*PSLACE(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLACE(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLACE(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 3) THEN
  DO JL=1,KLEN
    PSLAR(JL)=PSLAC(JL)
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

But we can see that this result is not correct, because under case KTYP=3 there is PSLAC instead of PSLACE. Modification of PSLAC into PSLACE must be done by hand. The final right result is:

```
SUBROUTINE TRACARELA(PSLACE,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                       C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE               C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLACE(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLACE(JL)*PSLACE(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLACE(JL))
    ZB=ZC2P1*PSLACE(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLACE(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLACE(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 3) THEN
  DO JL=1,KLEN
    PSLAR(JL)=PSLACE(JL)
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

This example shows that:
Even if an automatic source merging performs well without any conflict, we cannot always guarantee that the result is good. In some cases additional manual modifications may be required!

∗ Example 3:

This is a case where an automatic merging will be impossible to do, and where a manual merging becomes very easy once well understood the different contributions.

Contributor 1 renames PSLAR into PSLARE and ZA into ZZA.

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLARE,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                       C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE               C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLARE(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLARE(JL)=ZC*ZA
    PSLARE(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLARE(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLARE(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLARE(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLARE(JL)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

Contributor 2 exchanges KTYP=2 code with KTYP=1 code.

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                       C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE               C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

The way to merge the two contributions is either to start from the first one, then to introduce manually the second one, or to start from the second one, then to introduce manually the first one. Both ways have a similar difficulty in our case. The final right result is:

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLARE,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                       C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE               C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLARE(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLARE(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLARE(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLARE(JL)))
  ENDDO
ELSEIF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLARE(JL)=ZC*ZA
    PSLARE(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLARE(JL)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

∗ Example 4:

This is a case where additional modifications must be reported in another subroutine.

```
Contributor 1 changes the dimensions of array PSLAR.

SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                        C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE                C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(2,KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"


IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(1,JL)=ZC*ZA
    PSLAR(1,JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(1,JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(1,JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(1,JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(1,JL)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

```
Contributor 2 puts the KTYP=1 code in a subroutine.

SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                        C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE                C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"
#include "tracarela1.intfb.h

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  CALL TRACARELA1(KLEN,ZC2P1,ZC2M1,PSLAC,PSLAR)



ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

Content of the new routine TRACARELA1 is:

```
SUBROUTINE TRACARELA1(KLEN,PC2P1,PC2M1,PSLAC,PSLAR)

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PC2P1
REAL(KIND=JPRB)    ,INTENT(IN)    :: PC2M1
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN)

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZHOOK_HANDLE

IF (LHOOK) CALL DR_HOOK('TRACARELA1',0,ZHOOK_HANDLE)

DO JL=1,KLEN
  PSLAR(JL)=(PC2P1*PSLAC(JL)+PC2M1)/(PC2P1+PC2M1*PSLAC(JL))
  PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
ENDDO

IF (LHOOK) CALL DR_HOOK('TRACARELA1',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA1
```

An automatic source merging for routine TRACARELA will lead to minor conflicts under the KTYP=1 code, which can be easily solved. That leads to the following result.

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                        C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE                C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(2,KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"
#include "tracarela1.intfb.h

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(1,JL)=ZC*ZA
    PSLAR(1,JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(1,JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  CALL TRACARELA1(KLEN,ZC2P1,ZC2M1,PSLAC,PSLAR)
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

But the dimension change for array PSLAR should be reported in the new routine TRACARELA1. The final content of the new routine TRACARELA1 is:

```
SUBROUTINE TRACARELA1(KLEN,PC2P1,PC2M1,PSLAC,PSLAR)

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PC2P1
REAL(KIND=JPRB)    ,INTENT(IN)    :: PC2M1
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(2,KLEN)

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZHOOK_HANDLE

IF (LHOOK) CALL DR_HOOK('TRACARELA1',0,ZHOOK_HANDLE)

DO JL=1,KLEN
  PSLAR(1,JL)=(PC2P1*PSLAC(JL)+PC2M1)/(PC2P1+PC2M1*PSLAC(JL))
  PSLAR(1,JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(1,JL)))
ENDDO

IF (LHOOK) CALL DR_HOOK('TRACARELA1',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA1
```

This example shows that:
Side effects on another routines may be encountered when merging (need to report some modifications in some other routines).

∗ Example 5:

This is a case where one of the contributions must be ignored.

```
Contributor 1 renames variable ZA into ZZA

SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                      C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE              C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

```
Contributor 2 removes case KTYP=2.

SUBROUTINE TRACARELA(PSLAC,PSLAR,KLEN,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                      C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE              C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSLAC(KLEN)

REAL(KIND=JPRB)   ,INTENT(OUT)   :: PSLAR(KLEN)


REAL(KIND=JPRB)   ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZC2M1, ZC2P1
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)


ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

We can notice that array ZA has disappeared in the contribution given by contributor 2. In this case, modification given by contributor 1 can be completely ignored. The final result for TRACARELA is identical to what contributor 2 has given.

∗ Example 6:

This is a case where contributor 1 removes a useless dummy argument, and where contributor 2 does modifications, the consequence of which the useless dummy argument becomes used.

We start from a slightly modified version of TRACARELA containing an additional (useless) dummy argument KD.

```
SUBROUTINE TRACARELA(KD,PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                       C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE               C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM)               :: KD         ! Argument NOT used
INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)   :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)   :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)  :: PSLAR(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)   :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)   :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)   :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

Modifications proposed:

Contributor 1 removes the useless dummy argument KD.

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                       C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE               C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE


INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)   :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)   :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)  :: PSLAR(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)   :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)   :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)   :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

Contributor 2 changes dimension of array PSLAR, by using KD.

```
SUBROUTINE TRACARELA(KD,PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                       C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE               C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KD
INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)   :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)   :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)  :: PSLAR(KLEN,KD)
REAL(KIND=JPRB)    ,INTENT(IN)   :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)   :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)   :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL,1)=ZC*ZA
    PSLAR(JL,1)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL,1)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL,1)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL,1)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL,1)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

An automatic source merging for routine TRACARELA does not give any conflict, and leads to the following result:

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                      C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE              C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN,KD)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL,1)=ZC*ZA
    PSLAR(JL,1)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL,1)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL,1)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL,1)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL,1)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

But this result is not correct because variable KD is now used to dimension PSLAR. Passing KD as dummy argument, and declaration of KD (with the proper intent attribute) should be restored. The proper result is:

```
SUBROUTINE TRACARELA(KD,PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                      C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE              C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KD
INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN,KD)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL,1)=ZC*ZA
    PSLAR(JL,1)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL,1)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL,1)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL,1)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL,1)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

∗ Example 7:

This is a case where a modification is provided similarly on both sides.

We start from a slightly modified version of TRACARELA containing an additional (useless) dummy argument KD (see example 6).

Modifications proposed:

```
Contributor 1 removes the useless dummy argument KD.

SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                      C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE              C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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


IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO

ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO

ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

```
Contributor 2 removes the useless dummy argument KD and adds printings.

SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                      C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE              C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

USE PARKIND1  ,ONLY : JPIM     ,JPRB
USE YOMHOOK   ,ONLY : LHOOK,   DR_HOOK
USE YOMLUN    ,ONLY : NULOUT


IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

An automatic source merging for routine TRACARELA gives minor conflicts easy to solve, but we can notice that the final result is identical to contribution brought by contributor 2 in our case, because contribution number 2 contains all the modifications of contribution number 1.

∗ Example 8:

This is a case where the merging leads to a duplicated line.

Contributor 1 adds printings for case KTYP=1.

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                      C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE              C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

USE YOMLUN    ,ONLY : NULOUT
USE PARKIND1  ,ONLY : JPIM     ,JPRB
USE YOMHOOK   ,ONLY : LHOOK,   DR_HOOK


IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO

ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

Contributor 2 adds printings for case KTYP=2.

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                      C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE              C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


USE PARKIND1  ,ONLY : JPIM     ,JPRB
USE YOMHOOK   ,ONLY : LHOOK,   DR_HOOK
USE YOMLUN    ,ONLY : NULOUT

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO

ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

An automatic source merging for routine TRACARELA does not give any conflict, and leads to the following result:

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                          C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE                  C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

USE YOMLUN    ,ONLY : NULOUT
USE PARKIND1  ,ONLY : JPIM     ,JPRB
USE YOMHOOK   ,ONLY : LHOOK,   DR_HOOK
USE YOMLUN    ,ONLY : NULOUT

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

But we can see that this is not correct because NULOUT is declared twice (even if compilators often do not say anything in this case). The proper final result is:

```
SUBROUTINE TRACARELA(PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                          C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE                  C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

USE PARKIND1  ,ONLY : JPIM     ,JPRB
USE YOMHOOK   ,ONLY : LHOOK,   DR_HOOK
USE YOMLUN    ,ONLY : NULOUT

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL)=ZC*ZA
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL)))
  ENDDO
  WRITE(NULOUT,'(1X,E20.14)') PSLAR(1)
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

∗ Example 9:

This is a case where the merging does not give any conflict, but where additional modifications should be reported on some callers.

We assume that initially, TRACARELA is called by one routine: CALLER1.

- Contributor 1 adds a new dummy argument KD and modifies dimension of PSLAR in TRACARELA. Caller CALLER1 is adapted.
- Contributor 2 does not modify TRACARELA, but introduces calls to TRACARELA in routines CALLER2 and CALLER3, which formerly did not call TRACARELA.

```
Contributor 1 adds a new dummy argument KD and modifies dimension of PSLAR.

SUBROUTINE TRACARELA(KD,PSLAC,PCLOC,PSLAR,KLEN,PSLAP,PCLAP,PFACDI,KTYP)

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!  LATITUDE:                                                        C
!  PASSAGE DE LA SPHERE DE CALCUL A LA SPHERE REELLE                C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

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

IMPLICIT NONE

INTEGER(KIND=JPIM),INTENT(IN)    :: KD
INTEGER(KIND=JPIM),INTENT(IN)    :: KLEN
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAC(KLEN)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLOC(KLEN)
REAL(KIND=JPRB)    ,INTENT(OUT)   :: PSLAR(KLEN,KD)
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PCLAP
REAL(KIND=JPRB)    ,INTENT(IN)    :: PFACDI
INTEGER(KIND=JPIM),INTENT(IN)    :: KTYP

INTEGER(KIND=JPIM) :: JL
REAL(KIND=JPRB) :: ZA, ZB, ZC, ZC2M1, ZC2P1, ZCLAC, ZDBLC
REAL(KIND=JPRB) :: ZHOOK_HANDLE

#include "abor1.intfb.h"

IF (LHOOK) CALL DR_HOOK('TRACARELA',0,ZHOOK_HANDLE)

ZDBLC=2.0_JPRB*PFACDI
ZC2P1=PFACDI*PFACDI+1.0_JPRB
ZC2M1=PFACDI*PFACDI-1.0_JPRB

IF (KTYP == 2) THEN
  DO JL=1,KLEN
    ZCLAC=SQRT(1.0_JPRB-PSLAC(JL)*PSLAC(JL))
    ZA=1.0_JPRB/(ZC2P1+ZC2M1*PSLAC(JL))
    ZB=ZC2P1*PSLAC(JL)+ZC2M1
    ZC=ZDBLC*PCLAP*ZCLAC*PCLOC(JL)+ZB*PSLAP
    PSLAR(JL,1)=ZC*ZA
    PSLAR(JL,1)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL,1)))
  ENDDO
ELSEIF (KTYP == 1) THEN
  DO JL=1,KLEN
    PSLAR(JL,1)=(ZC2P1*PSLAC(JL)+ZC2M1)/(ZC2P1+ZC2M1*PSLAC(JL))
    PSLAR(JL,1)=MAX(-1.0_JPRB, MIN(1.0_JPRB,PSLAR(JL,1)))
  ENDDO
ELSE
  CALL ABOR1('TRACARELA: ABOR1 CALLED')
ENDIF

IF (LHOOK) CALL DR_HOOK('TRACARELA',1,ZHOOK_HANDLE)
END SUBROUTINE TRACARELA
```

An automatic source merging does not provide any conflict, and leads to the following result:

- TRACARELA and CALLER1 is identical to version provided by contributor 1.
- CALLER2 and CALLER3 call TRACARELA but with the old list of dummy arguments (missing KD, misdimensioned array matching with PSLAR).

So an additional modification should be provided to routines CALLER2 and CALLER3 in order to have the proper call to TRACARELA and the proper dimension for arrays entering TRACARELA.

∗ Example 10:

We now abandon the simple example of routine TRACARELA, and consider a longer routine present in ARP/IFS. We give an example of difficult source merging between CY36R4 and CY36T2: spcsi.F90. If we try to do a source merging via CLEARCASE, we can notice that this is very difficult. There are a lot of conflicts and the proper result will be probably impossible to obtain with this method. In this case the only way to solve the source merging is:

- Understand the different contributions which have been introduced in CY36R4.
- Understand the different contributions which have been introduced in CY36T2.
- Find the easier method among the two following ones: reintroduce the CY36R4 modifications on the top of CY36T2, or reintroduce the CY36T2 modifications on the top of CY36R4.

We now show the code source of SPCSI for the three following cycles: CY36, CY36T2 and CY36R4.

CY36:

```
SUBROUTINE SPCSI(&
 ! --- INPUT ----------------------------------------------------------------
 & CDCONF,KM,KMLOC,KSTA,KEND,LDONEM, &
 ! --- INOUT ----------------------------------------------------------------
 & PSPVORG,PSPDIVG,PSPTG,PSPSPG)

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

USE YOMMP     , ONLY : MYSETV   ,MYSETW   ,NPROCM   ,NPTRSV   ,NPTRSVF
USE YOMDIM    , ONLY : NFLEVG   ,NSMAX
USE YOMCST    , ONLY : ROMEGA
USE YOMCTO    , ONLY : NCONF    ,LSLAG    ,LTWOTL   ,LSITRIC
USE YOMDYN    , ONLY : SCGMAP   ,SIRDEL   ,SIMO     ,SIMI     ,&
 & SIVP     ,SIHEG    ,SIHEG2   ,SITRICA  ,SITRICB  ,&
 & SITRICC  ,TDT      ,BETADT   ,VESL     ,LSIDG    ,&
 & XIDT     ,LIMPF    ,LVERFLT  ,REPSVFVO ,REPSVFDI ,&
 & NLEVVF
USE YOMLAP    , ONLY : NVALUE   ,NSEOL    ,RLAPDI   ,RLAPIN
USE YOMGEM    , ONLY : RSTRET

!**** *SPCSI* - SPECTRAL SPACE SEMI-IMPLICIT COMPUTATIONS FOR HYD MODEL.

!     Purpose.
!     --------

!**   Interface.
!     ----------
!        *CALL* *SPCSI(..)

!        Explicit arguments :  CDCONF  - configuration (see doc.)
!        --------------------  KM      - Zonal wavenumber
!                              KMLOC   - Zonal wavenumber (DM-local numbering)
!                              KSTA    - First column processed
!                              KEND    - Last column processed
!                              LDONEM  - T if only one m if processed
!                              PSPVORG - Vorticity columns
!                              PSPDIVG - Divergence columns
!                              PSPTG   - Temperature columns
!                              PSPSPG  - Surface Pressure

!     Method.
!     -------

!     Externals.
!     ----------

!     Reference.
!     ----------
!        ECMWF Research Department documentation of the IFS

!     Author.
!     -------
!        Mats Hamrud and Philippe Courtier  *ECMWF*

!     Modifications.
!     --------------
!        Original : 87-11-24 (before 1997 spcsi.F was part of spc.F)
!        Modified 91-05-21 by A.Lasserre-Bigorry (scalar variables)
!        Modified DEC 1992 by K. YESSAD: decentering factor in semi-lag scheme.
!        Modified JAN 1994 by D GIARD: CDCONF='I'.
!        Modified JAN 1994 by K. YESSAD: semi-implicit computations
!           with not reduced divergence + new formulation of variable
!           mesh horizontal diffusion + multitasking on m instead of n.
!        Modified 95-02-06 by C. Temperton: semi-implicit Coriolis term,
!           real/imag parts stored in conventional order in work arrays
!        K. YESSAD (MARCH 1995):
!            - Optimisation of semi-implicit scheme if variable decentering
!              and remove of IMSL routines.
!            - Use new array NSEO (of YOMLAP) to define ISO.
!        Modified 95-06-06 by L.Isaksen : Reordering of spectral arrays
!        Modified 96-04-16 by C. Temperton: "alternative averaging" in
!                                           two-time-level scheme
!        Modified 96-05-16 by C. Temperton: new semi-implicit solver
!        Modified OCT 1996 by K. YESSAD: removal of variable uncentering.
!        Modified 12/12/96 by L.Isaksen: Merged with message passing
!                                        version. Renamed and recoded.
!        C. Temperton 97-02-24: modified first timestep when XIDT>0
!        R. El Khatib : 01-08-07 Pruning options
!        M.Hamrud      01-Oct-2003 CY28 Cleaning
!        N.Wedi        08-Mar-2005 remove mass correction
!        K.Yessad 09-Dec-2004: move mass correction in SPCMASCOR + cleanings.
!        K. Yessad 15-May-2006: memory optimisations for stretched geometry
!   N. Wedi and K. Yessad (Jan 2008): different dev for NH model and PC scheme
!     ------------------------------------------------------------------

IMPLICIT NONE

CHARACTER(LEN=1)   ,INTENT(IN)    :: CDCONF
INTEGER(KIND=JPIM),INTENT(IN)     :: KM
INTEGER(KIND=JPIM),INTENT(IN)     :: KMLOC
INTEGER(KIND=JPIM),INTENT(IN)     :: KSTA
INTEGER(KIND=JPIM),INTENT(IN)     :: KEND
LOGICAL           ,INTENT(IN)     :: LDONEM
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPVORG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPDIVG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPTG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPSPG(:)

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

REAL(KIND=JPRB) :: ZZSPVORG(NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZZSPDIVG(NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZZSPTG  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZZSPSPG (       KSTA:KEND)

REAL(KIND=JPRB) :: ZSDIV  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZHELP  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZST    (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZSP    (       KSTA:KEND)
REAL(KIND=JPRB) :: ZSDIVP (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZSPDIVP(NFLEVG,KSTA:KEND)
```

```
REAL(KIND=JPRB) :: ZSDIVPL (NFLEVG,KM:NSMAX,2)
REAL(KIND=JPRB) :: ZSPDIVPL(NFLEVG,KM:NSMAX,2)

REAL(KIND=JPRB) :: ZALPHA (KM:NSMAX+1)
REAL(KIND=JPRB) :: ZDENIM (KM:NSMAX+1)
REAL(KIND=JPRB) :: ZEPSI  (KM:NSMAX)
REAL(KIND=JPRB) :: ZFPLUS (KM:NSMAX+1)
REAL(KIND=JPRB) :: ZFMINUS(KM:NSMAX+1)

INTEGER(KIND=JPIM) :: II, ILO, IMSP, IN, IOFF, IRSP, IS0, IS02,&
 & ISE, ISPCOL, ISTART, JL, JLEV, JN, JSP

REAL(KIND=JPRB) :: ZAL, ZBDT, ZBDT2, ZEM, ZEN, ZF, ZTEMP, ZCT

LOGICAL :: LL3D,LLDOSI,LLDOVERFLT

REAL(KIND=JPRB) :: ZHOOK_HANDLE

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

INTERFACE
#include "mxmaop.h"
END INTERFACE

#include "mxptma.h"
#include "mxture.h"
#include "mxturs.h"
#include "sigam.intfb.h"
#include "simplico.h"
#include "sitnu.intfb.h"
#include "tricsi.intfb.h"

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

IF (LHOOK) CALL DR_HOOK('SPCSI',0,ZHOOK_HANDLE)

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

LL3D=(NCONF == 1.OR.NCONF == 131.OR.NCONF == 302.OR.NCONF == 401 &
 & .OR.NCONF == 501.OR.NCONF == 601.OR.NCONF == 801)
LLDOSI=LL3D.AND.(CDCONF == 'A'.OR.CDCONF == 'I'.OR.&
 & CDCONF == 'S' .OR. CDCONF == 'T')
LLDOVERFLT=LL3D.AND.LVERFLT.AND.CDCONF == 'A'

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

!*       1.    MEMORY TRANSFER.
!              ----------------

IF (LL3D) THEN
  IF (LIMPF.OR.LVERFLT) ZZSPVORG(1:NFLEVG,KSTA:KEND)=&
   & PSPVORG(1:NFLEVG,KSTA:KEND)
  ZZSPDIVG(1:NFLEVG,KSTA:KEND)=PSPDIVG(1:NFLEVG,KSTA:KEND)
  ZZSPTG  (1:NFLEVG,KSTA:KEND)=PSPTG  (1:NFLEVG,KSTA:KEND)
  ZZSPSPG (         KSTA:KEND)=PSPSPG (         KSTA:KEND)
ENDIF

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

!*       2.    SEMI-IMPLICIT SPECTRAL COMPUTATIONS.
!              ------------------------------------

IF (LLDOSI) THEN

  !*        2.1  Preliminary initialisations.

  IF (LDONEM) THEN
    IOFF=NPTRSVF(MYSETV)-1
  ELSE
    IOFF=NPTRSV(MYSETV)-1
  ENDIF
  ISPCOL=KEND-KSTA+1

  IF (LSLAG) THEN
    IF (LTWOTL) THEN
      ZBDT=(1.0_JPRB+VESL)*0.5_JPRB*TDT*BETADT*(1.0_JPRB+XIDT)
    ELSE
      ZBDT=(1.0_JPRB+VESL)*0.5_JPRB*TDT*BETADT
    ENDIF
  ELSE
    ZBDT=0.5_JPRB*TDT*BETADT
  ENDIF
  ZBDT2=(ZBDT*RSTRET)**2

  !*        2.2  Set up helper arrays for implicit Coriolis case.

  IF (LIMPF) THEN
    ZEM=REAL(KM,JPRB)
    ZAL=2.0_JPRB*ZBDT*ROMEGA*ZEM
    ILO=KM
    IF (KM == 0) THEN
      ZALPHA(0)=0.0_JPRB
      ZDENIM(0)=0.0_JPRB
      ZEPSI(0)=0.0_JPRB
      ILO=1
    ENDIF
    DO JL=ILO,NSMAX
      ZEN=REAL(JL,JPRB)
      ZALPHA(JL)=ZAL/(ZEN*(ZEN+1.0_JPRB))
      ZDENIM(JL)=1.0_JPRB/(1.0_JPRB+ZALPHA(JL)**2)
      ZEPSI(JL)=SQRT((ZEN*ZEN-ZEM*ZEM)/(4.0_JPRB*ZEN*ZEN-1.0_JPRB))
    ENDDO
    ZALPHA(NSMAX+1)=0.0_JPRB
    ZDENIM(NSMAX+1)=0.0_JPRB

    IF (KM == 0) THEN
      ZFPLUS(0)=0.0_JPRB
      ZFMINUS(0)=0.0_JPRB
    ENDIF
    ZF=2.0_JPRB*ZBDT*ROMEGA
    DO JL=ILO,NSMAX-1
      ZEN=REAL(JL,JPRB)
      ZFPLUS(JL)=ZF*ZEN*ZEPSI(JL+1)/(ZEN+1.0_JPRB)
      ZFMINUS(JL)=ZF*(ZEN+1.0_JPRB)*ZEPSI(JL)/ZEN
    ENDDO
```

```
  ZEN=REAL(NSMAX,JPRB)
  ZFPLUS(NSMAX)=0.0_JPRB
  ZFMINUS(NSMAX)=ZF*(ZEN+1.0_JPRB)*ZEPSI(NSMAX)/ZEN
  ZFPLUS(NSMAX+1)=0.0_JPRB
  ZFMINUS(NSMAX+1)=0.0_JPRB
ENDIF

!*        2.3  Computes right-hand side of Helmholtz equation.

CALL SIGAM(1,NFLEVG,ZSDIV,ZZSPTG,ZZSPSPG,ISPCOL)

IF (LSIDG) THEN
  IF (KM > 0) THEN
    DO JLEV=1,NFLEVG
      DO JSP=KSTA,KEND
        IN=NVALUE(JSP+IOFF)
        ZSDIV(JLEV,JSP)=RLAPIN(IN)*ZZSPDIVG(JLEV,JSP)&
         & -ZBDT*ZSDIV(JLEV,JSP)
      ENDDO
    ENDDO
  ELSE
    DO JLEV=1,NFLEVG
      DO JSP=KSTA,KEND
        IN=NVALUE(JSP+IOFF)
        ZSDIV(JLEV,JSP)=ZZSPDIVG(JLEV,JSP)&
         & -ZBDT*RLAPDI(IN)*ZSDIV(JLEV,JSP)
      ENDDO
    ENDDO
  ENDIF
ELSE

  ! Case of No Stretching

  DO JLEV=1,NFLEVG
    DO JSP=KSTA,KEND
      IN=NVALUE(JSP+IOFF)
      ZSDIV(JLEV,JSP)=ZZSPDIVG(JLEV,JSP)-ZBDT*RLAPDI(IN)*ZSDIV(JLEV,JSP)
    ENDDO
  ENDDO
ENDIF

!        For implicit Coriolis case, multiply rhs of vorticity equn by INV[J]
!        Careful : LIMPF assumes all n-values are available !

IF (LIMPF) THEN
  IF (KM > 0) THEN
    DO JLEV=1,NFLEVG
      DO JN=KM,NSMAX
        IRSP=KSTA+(JN-KM)*2
        IMSP=IRSP+1
        ZTEMP=ZDENIM(JN)*(ZZSPVORG(JLEV,IMSP)&
         & +ZALPHA(JN)*ZZSPVORG(JLEV,IRSP))
        ZZSPVORG(JLEV,IRSP)=ZDENIM(JN)*(ZZSPVORG(JLEV,IRSP)&
         & -ZALPHA(JN)*ZZSPVORG(JLEV,IMSP))
        ZZSPVORG(JLEV,IMSP)=ZTEMP
      ENDDO
    ENDDO
  ENDIF

  !        Add [F] * result to rhs of Helmholtz equation

  DO JLEV=1,NFLEVG
    DO JN=KM+1,NSMAX
      IRSP=KSTA+(JN-KM)*2
      IMSP=IRSP+1
      ZSDIV(JLEV,IRSP)=ZSDIV(JLEV,IRSP)&
       & + ZFMINUS(JN)*ZZSPVORG(JLEV,IRSP-2)
      ZSDIV(JLEV,IMSP)=ZSDIV(JLEV,IMSP)&
       & + ZFMINUS(JN)*ZZSPVORG(JLEV,IMSP-2)
    ENDDO
  ENDDO

  DO JLEV=1,NFLEVG
    DO JN=KM,NSMAX-1
      IRSP=KSTA+(JN-KM)*2
      IMSP=IRSP+1
      ZSDIV(JLEV,IRSP)=ZSDIV(JLEV,IRSP)+ ZFPLUS(JN)*ZZSPVORG(JLEV,IRSP+2)
      ZSDIV(JLEV,IMSP)=ZSDIV(JLEV,IMSP)+ ZFPLUS(JN)*ZZSPVORG(JLEV,IMSP+2)
    ENDDO
  ENDDO
ENDIF

!*        2.4  Solve Helmholtz equation

IF (LSITRIC) THEN

  ! Tricky: The PE that owns wavenumber 0 starts at (m,n)=(1,1), so istart=3

  IF ((KM == 0.AND.LDONEM).OR.(MYSETW == NPROCM(0).AND..NOT.LDONEM)) THEN
    ISTART=KSTA+2
  ELSE
    ISTART=KSTA
  ENDIF
  CALL TRICSI(ZZSPDIVG(1,ISTART),ZSDIV(1,ISTART),&
   & SITRICA,SITRICB,SITRICC,SIRDEL,ISTART,KEND)
ELSE

  !           Current space --> vertical eigenmodes space.

  CALL MXMAOP(SIMI,1,NFLEVG,ZSDIV,1,NFLEVG,ZSDIVP,1,NFLEVG,&
   & NFLEVG,NFLEVG,ISPCOL)
  IF (LSIDG) THEN

    !             Inversion of two tridiagonal systems (Helmholtz equation)
    !                --> (SIMI*DIVprim(t+dt)).

    !             Reorganisation of divergence

    ISO=NSEOL(KMLOC)
    ISO2=0
    II=MIN(KM,1)+1
    ZSDIVPL(:,:,:)=0.0_JPRB
    ZSPDIVPL(:,:,:)=0.0_JPRB

    DO JN=KM,NSMAX
```

```
      ISE=KSTA+2*(JN-KM)
      ZSDIVPL(:,JN,1:2)=ZSDIVP(:,ISE:ISE+1)
    ENDDO
    IF (KM > 0) THEN

      !               Inversion of a symmetric matrix.

      CALL MXTURS(NSMAX+1-KM,NFLEVG,NFLEVG,II,&
       & SIHEG(1,ISO+1,1),SIHEG(1,ISO+1,2),SIHEG(1,ISO+1,3),&
       & ZSDIVPL,ZSPDIVPL)
    ELSE

      !               Inversion of a non-symmetric matrix.

      CALL MXTURE(NSMAX+1-KM,NFLEVG,NFLEVG,II,-2,.TRUE.,&
       & SIHEG(1,ISO+1,1),SIHEG(1,ISO+1,2),SIHEG(1,ISO+1,3),&
       & ZSDIVPL,ZSPDIVPL)
      CALL MXTURE(NSMAX+1-KM,NFLEVG,NFLEVG,II,3,.FALSE.,&
       & SIHEG(1,ISO+1,1),SIHEG2(1,ISO2+1,2),&
       & SIHEG2(1,ISO2+1,3),ZSDIVPL,ZSPDIVPL)
    ENDIF

    DO JN=KM,NSMAX
      ISE=KSTA+2*(JN-KM)
      ZSPDIVP(:,ISE:ISE+1)=ZSPDIVPL(:,JN,1:2)
    ENDDO
  ELSE

    !             Case with NO Stretching :

    IF (LIMPF) THEN

      !               Solve complex pentadiagonal system

      CALL SIMPLICO(KM,NSMAX,NFLEVG,NFLEVG,ZALPHA(KM),&
       & ZDENIM(KM),ZFPLUS(KM),ZFMINUS(KM),SIVP,RLAPDI(0),&
       & ZBDT2,ZSDIVP,ZSPDIVP)
    ELSE

      !                 Inversion of a diagonal system (Helmholtz equation)
      !                 --> (SIMI*DIVprim(t+dt)).

      DO JLEV=1,NFLEVG
        DO JSP=KSTA,KEND
          ZSPDIVP(JLEV,JSP)=ZSDIVP(JLEV,JSP)&
           & /(1.0_JPRB-ZBDT2*SIVP(JLEV)*RLAPDI(NVALUE(JSP+IOFF)))
        ENDDO
      ENDDO
    ENDIF
  ENDIF

  !           Vertical eigenmodes space --> current space.

  CALL MXMAOP(SIMO,1,NFLEVG,ZSPDIVP,1,NFLEVG,ZZSPDIVG,1,&
   & NFLEVG,NFLEVG,NFLEVG,ISPCOL)
ENDIF
IF (LSIDG) THEN

  !           ZSPDIV=(DIVprim(t+dt)) --> ZSPDIVG=(GM**2 * DIVprim(t+dt)) .

  ZSDIVPL(:,:,:)=0.0_JPRB
  ZSPDIVPL(:,:,:)=0.0_JPRB

  !           Reorganisation of ZSDIVP (Back to the USSR)

  DO JN=KM,NSMAX
    ISE=KSTA+2*(JN-KM)
    ZSDIVPL(:,JN,1:2)=ZZSPDIVG(:,ISE:ISE+1)
  ENDDO

  !        ZSPDIV=(DIVprim(t+dt)) --> ZPSPDIVG=(GMBAR**2 * DIVprim(t+dt)).

  CALL MXPTMA(NSMAX+1-KM,NFLEVG,NFLEVG,II,SCGMAP(ISO+1,1),&
   & SCGMAP(ISO+1,2),SCGMAP(ISO+1,3),&
   & SCGMAP(ISO+1,2),SCGMAP(ISO+1,3),&
   & ZSDIVPL,ZSPDIVPL)

  !           Reorganisation of ZSPDIVPL

  DO JN=KM,NSMAX
    ISE=KSTA+2*(JN-KM)
    ZHELP(:,ISE:ISE+1)=ZSPDIVPL(:,JN,1:2)
  ENDDO
ELSE

  !       ZSPDIV=(DIVprim(t+dt)) --> ZSPDIVG=(GMBAR**2 * DIVprim(t+dt)) .

  DO JLEV=1,NFLEVG
    DO JSP=KSTA,KEND
      ZHELP(JLEV,JSP)=ZZSPDIVG(JLEV,JSP)*RSTRET*RSTRET
    ENDDO
  ENDDO

ENDIF

!       If LSIDG:
!         (GM**2 * DIVprim(t+dt)) --> [ tau * (GM**2 * DIVprim(t+dt)) ]
!                                 and [  nu * (GM**2 * DIVprim(t+dt)) ]
!       or if not LSIDG:
!         (GMBAR**2 * DIVprim(t+dt)) --> [ tau * (GMBAR**2 * DIVprim(t+dt)) ]
!                                    and [  nu * (GMBAR**2 * DIVprim(t+dt)) ]

CALL SITNU(1,NFLEVG,ZHELP,ZST,ZSP,ISPCOL)

!*       2.5  Increment Temperature and surface pressure

DO JLEV=1,NFLEVG
  DO JSP=KSTA,KEND
    ZZSPTG(JLEV,JSP)=ZZSPTG(JLEV,JSP)-ZBDT*ZST(JLEV,JSP)
  ENDDO
ENDDO

DO JSP=KSTA,KEND
  ZZSPSPG(JSP)=ZZSPSPG(JSP)-ZBDT*ZSP(JSP)
ENDDO
```

```
  !*       2.6  Increment vorticity

  IF (LIMPF) THEN
    DO JLEV=1,NFLEVG
      IF (KM == 0) THEN
        DO JN=2,NSMAX
          IRSP=KSTA+(JN-KM)*2
          ZZSPVORG(JLEV,IRSP)=ZZSPVORG(JLEV,IRSP)&
           & -ZDENIM(JN)*ZFMINUS(JN)*ZZSPDIVG(JLEV,IRSP-2)
        ENDDO
        DO JN=1,NSMAX-1
          IRSP=KSTA+(JN-KM)*2
          ZZSPVORG(JLEV,IRSP)=ZZSPVORG(JLEV,IRSP)&
           & -ZDENIM(JN)*ZFPLUS(JN)*ZZSPDIVG(JLEV,IRSP+2)
        ENDDO
      ELSE
        DO JN=KM+1,NSMAX
          IRSP=KSTA+(JN-KM)*2
          IMSP=IRSP+1
          ZZSPVORG(JLEV,IRSP)=ZZSPVORG(JLEV,IRSP)&
           & -ZDENIM(JN)*ZFMINUS(JN)*(ZZSPDIVG(JLEV,IRSP-2)&
           & -ZALPHA(JN)*ZZSPDIVG(JLEV,IMSP-2))
          ZZSPVORG(JLEV,IMSP)=ZZSPVORG(JLEV,IMSP)&
           & -ZDENIM(JN)*ZFMINUS(JN)*(ZZSPDIVG(JLEV,IMSP-2)&
           & +ZALPHA(JN)*ZZSPDIVG(JLEV,IRSP-2))
        ENDDO
        DO JN=KM,NSMAX-1
          IRSP=KSTA+(JN-KM)*2
          IMSP=IRSP+1
          ZZSPVORG(JLEV,IRSP)=ZZSPVORG(JLEV,IRSP)&
           & -ZDENIM(JN)*ZFPLUS(JN)*(ZZSPDIVG(JLEV,IRSP+2)&
           & -ZALPHA(JN)*ZZSPDIVG(JLEV,IMSP+2))
          ZZSPVORG(JLEV,IMSP)=ZZSPVORG(JLEV,IMSP)&
           & -ZDENIM(JN)*ZFPLUS(JN)*(ZZSPDIVG(JLEV,IMSP+2)&
           & +ZALPHA(JN)*ZZSPDIVG(JLEV,IRSP+2))
        ENDDO
      ENDIF
    ENDDO
  ENDIF

ENDIF

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

!*       3.    MEMORY TRANSFER.
!              ----------------

IF (LL3D) THEN
  IF (LIMPF.OR.LVERFLT)  PSPVORG(1:NFLEVG,KSTA:KEND)=&
   & ZZSPVORG(1:NFLEVG,KSTA:KEND)
  PSPDIVG(1:NFLEVG,KSTA:KEND)=ZZSPDIVG(1:NFLEVG,KSTA:KEND)
  PSPTG  (1:NFLEVG,KSTA:KEND)=ZZSPTG  (1:NFLEVG,KSTA:KEND)
  PSPSPG (         KSTA:KEND)=ZZSPSPG (         KSTA:KEND)
ENDIF

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

!*       4.    VERTICAL FILTER.
!              ----------------

! 2-del-eta filter:  eps*f_(i-1) +(1-2*eps)*f_i + eps*f_(i+1)
! on VOR and DIV for levs 1 to NLEVVF

IF (LLDOVERFLT) THEN

  ZCT=TDT/(1._JPRB+TDT)
  PSPVORG(1,KSTA:KEND)=  ZZSPVORG(1,KSTA:KEND)&
   & +ZCT*REPSVFVO*(ZZSPVORG(2,KSTA:KEND)-ZZSPVORG(1,KSTA:KEND))
  PSPDIVG(1,KSTA:KEND)=  ZZSPDIVG(1,KSTA:KEND)&
   & +ZCT*REPSVFDI*(ZZSPDIVG(2,KSTA:KEND)-ZZSPDIVG(1,KSTA:KEND))
  DO JLEV=2,NLEVVF
    PSPVORG(JLEV,KSTA:KEND)=  ZZSPVORG(JLEV,KSTA:KEND)&
     & + ZCT*REPSVFVO*(ZZSPVORG(JLEV-1,KSTA:KEND)&
     & -2.0_JPRB*ZZSPVORG(JLEV,KSTA:KEND) &
     & + ZZSPVORG(JLEV+1,KSTA:KEND))
    PSPDIVG(JLEV,KSTA:KEND)=  ZZSPDIVG(JLEV,KSTA:KEND)&
     & + ZCT*REPSVFDI*(ZZSPDIVG(JLEV-1,KSTA:KEND)&
     & -2.0_JPRB*ZZSPDIVG(JLEV,KSTA:KEND) &
     & + ZZSPDIVG(JLEV+1,KSTA:KEND))
  ENDDO

ENDIF

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

IF (LHOOK) CALL DR_HOOK('SPCSI',1,ZHOOK_HANDLE)
END SUBROUTINE SPCSI
```

CY36T2:

```
SUBROUTINE SPCSI(&
 ! --- INPUT ----------------------------------------------------------------
 & CDCONF,KM,KMLOC,KSTA,KEND,LDONEM, &
 ! --- INOUT ----------------------------------------------------------------
 & PSPVORG,PSPDIVG,PSPTG,PSPSPG, &
 & PSPTNDSI_VORG,PSPTNDSI_DIVG,PSPTNDSI_TG)

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

USE YOMMP    , ONLY : MYSETV   ,NPTRSV   ,NPTRSVF
USE YOMDIM   , ONLY : NFLEVG   ,NSMAX
USE YOMCST   , ONLY : ROMEGA
USE YOMCTO   , ONLY : NCONF    ,LSLAG    ,LTWOTL
USE YOMDYN   , ONLY : SCGMAP   ,SIMO     ,SIMI     ,&
 & SIVP     ,SIHEG    ,SIHEG2   ,TDT      ,BETADT   ,VESL     ,LSIDG    ,&
 & XIDT     ,LIMPF    ,LVERFLT  ,REPSVFVO ,REPSVFDI ,NLEVVF
USE YOMLAP   , ONLY : NVALUE   ,NSEOL    ,RLAPDI   ,RLAPIN
USE YOMGEM   , ONLY : RSTRET
USE YOMLDDH   , ONLY : LRDDHDYN, LOPTSIDH

!**** *SPCSI* - SPECTRAL SPACE SEMI-IMPLICIT COMPUTATIONS FOR HYD MODEL.

!     Purpose.
!     --------

!**   Interface.
!     ----------
!        *CALL* *SPCSI(..)

!        Explicit arguments :  CDCONF  - configuration (see doc.)
!        --------------------  KM      - Zonal wavenumber
!                              KMLOC   - Zonal wavenumber (DM-local numbering)
!                              KSTA    - First column processed
!                              KEND    - Last column processed
!                              LDONEM  - T if only one m if processed
!                              PSPVORG - Vorticity columns
!                              PSPDIVG - Divergence columns
!                              PSPTG   - Temperature columns
!                              PSPSPG  - Surface Pressure

!     Method.
!     -------

!     Externals.
!     ----------

!     Reference.
!     ----------
!        ECMWF Research Department documentation of the IFS

!     Author.
!     -------
!        Mats Hamrud and Philippe Courtier  *ECMWF*

!     Modifications.
!     --------------
!        Original : 87-11-24 (before 1997 spcsi.F was part of spc.F)
!        Modified 91-05-21 by A.Lasserre-Bigorry (scalar variables)
!        Modified DEC 1992 by K. YESSAD: decentering factor in semi-lag scheme.
!        Modified JAN 1994 by D GIARD: CDCONF='I'.
!        Modified JAN 1994 by K. YESSAD: semi-implicit computations
!           with not reduced divergence + new formulation of variable
!           mesh horizontal diffusion + multitasking on m instead of n.
!        Modified 95-02-06 by C. Temperton: semi-implicit Coriolis term,
!           real/imag parts stored in conventional order in work arrays
!        K. YESSAD (MARCH 1995):
!            - Optimisation of semi-implicit scheme if variable decentering
!              and remove of IMSL routines.
!            - Use new array NSEO (of YOMLAP) to define ISO.
!        Modified 95-06-06 by L.Isaksen : Reordering of spectral arrays
!        Modified 96-04-16 by C. Temperton: "alternative averaging" in
!                                           two-time-level scheme
!        Modified 96-05-16 by C. Temperton: new semi-implicit solver
!        Modified OCT 1996 by K. YESSAD: removal of variable uncentering.
!        Modified 12/12/96 by L.Isaksen: Merged with message passing
!                                        version. Renamed and recoded.
!        C. Temperton 97-02-24: modified first timestep when XIDT>0
!        R. El Khatib : 01-08-07 Pruning options
!        M.Hamrud      01-Oct-2003 CY28 Cleaning
!        N.Wedi        08-Mar-2005 remove mass correction
!        K.Yessad 09-Dec-2004: move mass correction in SPCMASCOR + cleanings.
!        K. Yessad 15-May-2006: memory optimisations for stretched geometry
!   N. Wedi and K. Yessad (Jan 2008): different dev for NH model and PC scheme
!   K. Yessad (Aug 2009): remove LSITRIC option.
!     ------------------------------------------------------------------

IMPLICIT NONE

CHARACTER(LEN=1)   ,INTENT(IN)    :: CDCONF
INTEGER(KIND=JPIM),INTENT(IN)     :: KM
INTEGER(KIND=JPIM),INTENT(IN)     :: KMLOC
INTEGER(KIND=JPIM),INTENT(IN)     :: KSTA
INTEGER(KIND=JPIM),INTENT(IN)     :: KEND
LOGICAL            ,INTENT(IN)    :: LDONEM
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPVORG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPDIVG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPTG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPSPG(:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPTNDSI_VORG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPTNDSI_DIVG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPTNDSI_TG(:,:)

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

REAL(KIND=JPRB) :: ZZSPVORG(NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZZSPDIVG(NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZZSPTG  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZZSPSPG (       KSTA:KEND)

REAL(KIND=JPRB) :: ZSDIV  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZHELP  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZST    (NFLEVG,KSTA:KEND)
```

```
REAL(KIND=JPRB) :: ZSP    (       KSTA:KEND)
REAL(KIND=JPRB) :: ZSDIVP (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZSPDIVP(NFLEVG,KSTA:KEND)

REAL(KIND=JPRB) :: ZSDIVPL (NFLEVG,KM:NSMAX,2)
REAL(KIND=JPRB) :: ZSPDIVPL(NFLEVG,KM:NSMAX,2)

REAL(KIND=JPRB) :: ZALPHA (KM:NSMAX+1)
REAL(KIND=JPRB) :: ZDENIM (KM:NSMAX+1)
REAL(KIND=JPRB) :: ZEPSI  (KM:NSMAX)
REAL(KIND=JPRB) :: ZFPLUS (KM:NSMAX+1)
REAL(KIND=JPRB) :: ZFMINUS(KM:NSMAX+1)

INTEGER(KIND=JPIM) :: II, ILO, IMSP, IN, IOFF, IRSP, IS0, IS02,&
 & ISE, ISPCOL, JL, JLEV, JN, JSP

REAL(KIND=JPRB) :: ZAL, ZBDT, ZBDT2, ZEM, ZEN, ZF, ZTEMP, ZCT

LOGICAL :: LL3D,LLDOSI,LLDOVERFLT,LLDDHSI,LLOPTDDH

REAL(KIND=JPRB) :: ZHOOK_HANDLE

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

INTERFACE
#include "mxmaop.h"
END INTERFACE

#include "mxptma.h"
#include "mxture.h"
#include "mxturs.h"
#include "sigam.intfb.h"
#include "simplico.h"
#include "sitnu.intfb.h"

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

IF (LHOOK) CALL DR_HOOK('SPCSI',0,ZHOOK_HANDLE)

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

LL3D=(NCONF == 1.OR.NCONF == 131.OR.NCONF == 302.OR.NCONF == 401 &
 & .OR.NCONF == 501.OR.NCONF == 601.OR.NCONF == 801)
LLDOSI=LL3D.AND.(CDCONF == 'A'.OR.CDCONF == 'I'.OR.&
 & CDCONF == 'S' .OR. CDCONF == 'T')
LLDOVERFLT=LL3D.AND.LVERFLT.AND.CDCONF == 'A'
LLDDHSI=LL3D.AND.LRDDHDYN
LLOPTDDH=LLDDHSI.AND.LOPTSIDH

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

!*       1.    MEMORY TRANSFER.
!              ----------------

IF (LL3D) THEN
  IF (LIMPF.OR.LVERFLT) ZZSPVORG(1:NFLEVG,KSTA:KEND)=&
   & PSPVORG(1:NFLEVG,KSTA:KEND)
  ZZSPDIVG(1:NFLEVG,KSTA:KEND)=PSPDIVG(1:NFLEVG,KSTA:KEND)
  ZZSPTG  (1:NFLEVG,KSTA:KEND)=PSPTG  (1:NFLEVG,KSTA:KEND)
  ZZSPSPG (         KSTA:KEND)=PSPSPG (         KSTA:KEND)
ENDIF
IF (LLOPTDDH) THEN
  IF (LIMPF.OR.LVERFLT) PSPTNDSI_VORG(1:NFLEVG,KSTA:KEND)=&
    & - PSPVORG(1:NFLEVG,KSTA:KEND)
  PSPTNDSI_DIVG(1:NFLEVG,KSTA:KEND)=-PSPDIVG(1:NFLEVG,KSTA:KEND)
  PSPTNDSI_TG  (1:NFLEVG,KSTA:KEND)=-PSPTG  (1:NFLEVG,KSTA:KEND)
  !the case of surface pressure has not been treated yet
ENDIF

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

!*       2.    SEMI-IMPLICIT SPECTRAL COMPUTATIONS.
!              ------------------------------------

IF (LLDOSI) THEN

  !*        2.1  Preliminary initialisations.

  IF (LDONEM) THEN
    IOFF=NPTRSVF(MYSETV)-1
  ELSE
    IOFF=NPTRSV(MYSETV)-1
  ENDIF
  ISPCOL=KEND-KSTA+1

  IF (LSLAG) THEN
    IF (LTWOTL) THEN
      ZBDT=(1.0_JPRB+VESL)*0.5_JPRB*TDT*BETADT*(1.0_JPRB+XIDT)
    ELSE
      ZBDT=(1.0_JPRB+VESL)*0.5_JPRB*TDT*BETADT
    ENDIF
  ELSE
    ZBDT=0.5_JPRB*TDT*BETADT
  ENDIF
  ZBDT2=(ZBDT*RSTRET)**2

  !*        2.2  Set up helper arrays for implicit Coriolis case.

  IF (LIMPF) THEN
    ZEM=REAL(KM,JPRB)
    ZAL=2.0_JPRB*ZBDT*ROMEGA*ZEM
    ILO=KM
    IF (KM == 0) THEN
      ZALPHA(0)=0.0_JPRB
      ZDENIM(0)=0.0_JPRB
      ZEPSI(0)=0.0_JPRB
      ILO=1
    ENDIF
    DO JL=ILO,NSMAX
      ZEN=REAL(JL,JPRB)
      ZALPHA(JL)=ZAL/(ZEN*(ZEN+1.0_JPRB))
      ZDENIM(JL)=1.0_JPRB/(1.0_JPRB+ZALPHA(JL)**2)
      ZEPSI(JL)=SQRT((ZEN*ZEN-ZEM*ZEM)/(4.0_JPRB*ZEN*ZEN-1.0_JPRB))
    ENDDO
    ZALPHA(NSMAX+1)=0.0_JPRB
```

```
  ZDENIM(NSMAX+1)=0.0_JPRB

  IF (KM == 0) THEN
    ZFPLUS(0)=0.0_JPRB
    ZFMINUS(0)=0.0_JPRB
  ENDIF
  ZF=2.0_JPRB*ZBDT*ROMEGA
  DO JL=ILO,NSMAX-1
    ZEN=REAL(JL,JPRB)
    ZFPLUS(JL)=ZF*ZEN*ZEPSI(JL+1)/(ZEN+1.0_JPRB)
    ZFMINUS(JL)=ZF*(ZEN+1.0_JPRB)*ZEPSI(JL)/ZEN
  ENDDO
  ZEN=REAL(NSMAX,JPRB)
  ZFPLUS(NSMAX)=0.0_JPRB
  ZFMINUS(NSMAX)=ZF*(ZEN+1.0_JPRB)*ZEPSI(NSMAX)/ZEN
  ZFPLUS(NSMAX+1)=0.0_JPRB
  ZFMINUS(NSMAX+1)=0.0_JPRB
ENDIF

!*        2.3  Computes right-hand side of Helmholtz equation.

CALL SIGAM(1,NFLEVG,ZSDIV,ZZSPTG,ZZSPSPG,ISPCOL)

IF (LSIDG) THEN
  IF (KM > 0) THEN
    DO JLEV=1,NFLEVG
      DO JSP=KSTA,KEND
        IN=NVALUE(JSP+IOFF)
        ZSDIV(JLEV,JSP)=RLAPIN(IN)*ZZSPDIVG(JLEV,JSP)&
         & -ZBDT*ZSDIV(JLEV,JSP)
      ENDDO
    ENDDO
  ELSE
    DO JLEV=1,NFLEVG
      DO JSP=KSTA,KEND
        IN=NVALUE(JSP+IOFF)
        ZSDIV(JLEV,JSP)=ZZSPDIVG(JLEV,JSP)&
         & -ZBDT*RLAPDI(IN)*ZSDIV(JLEV,JSP)
      ENDDO
    ENDDO
  ENDIF
ELSE

  ! Case of No Stretching

  DO JLEV=1,NFLEVG
    DO JSP=KSTA,KEND
      IN=NVALUE(JSP+IOFF)
      ZSDIV(JLEV,JSP)=ZZSPDIVG(JLEV,JSP)-ZBDT*RLAPDI(IN)*ZSDIV(JLEV,JSP)
    ENDDO
  ENDDO
ENDIF

!        For implicit Coriolis case, multiply rhs of vorticity equn by INV[J]
!        Careful : LIMPF assumes all n-values are available !

IF (LIMPF) THEN
  IF (KM > 0) THEN
    DO JLEV=1,NFLEVG
      DO JN=KM,NSMAX
        IRSP=KSTA+(JN-KM)*2
        IMSP=IRSP+1
        ZTEMP=ZDENIM(JN)*(ZZSPVORG(JLEV,IMSP)&
         & +ZALPHA(JN)*ZZSPVORG(JLEV,IRSP))
        ZZSPVORG(JLEV,IRSP)=ZDENIM(JN)*(ZZSPVORG(JLEV,IRSP)&
         & -ZALPHA(JN)*ZZSPVORG(JLEV,IMSP))
        ZZSPVORG(JLEV,IMSP)=ZTEMP
      ENDDO
    ENDDO
  ENDIF

  !        Add [F] * result to rhs of Helmholtz equation

  DO JLEV=1,NFLEVG
    DO JN=KM+1,NSMAX
      IRSP=KSTA+(JN-KM)*2
      IMSP=IRSP+1
      ZSDIV(JLEV,IRSP)=ZSDIV(JLEV,IRSP)&
       & + ZFMINUS(JN)*ZZSPVORG(JLEV,IRSP-2)
      ZSDIV(JLEV,IMSP)=ZSDIV(JLEV,IMSP)&
       & + ZFMINUS(JN)*ZZSPVORG(JLEV,IMSP-2)
    ENDDO
  ENDDO

  DO JLEV=1,NFLEVG
    DO JN=KM,NSMAX-1
      IRSP=KSTA+(JN-KM)*2
      IMSP=IRSP+1
      ZSDIV(JLEV,IRSP)=ZSDIV(JLEV,IRSP)+ ZFPLUS(JN)*ZZSPVORG(JLEV,IRSP+2)
      ZSDIV(JLEV,IMSP)=ZSDIV(JLEV,IMSP)+ ZFPLUS(JN)*ZZSPVORG(JLEV,IMSP+2)
    ENDDO
  ENDDO
ENDIF

!*        2.4  Solve Helmholtz equation

!           Current space --> vertical eigenmodes space.

CALL MXMAOP(SIMI,1,NFLEVG,ZSDIV,1,NFLEVG,ZSDIVP,1,NFLEVG,&
 & NFLEVG,NFLEVG,ISPCOL)
IF (LSIDG) THEN

  !             Inversion of two tridiagonal systems (Helmholtz equation)
  !                --> (SIMI*DIVprim(t+dt)).

  !             Reorganisation of divergence

  ISO=NSEOL(KMLOC)
  ISO2=0
  II=MIN(KM,1)+1
  ZSDIVPL(:,:,:)=0.0_JPRB
  ZSPDIVPL(:,:,:)=0.0_JPRB

  DO JN=KM,NSMAX
    ISE=KSTA+2*(JN-KM)
```

```
    ZSDIVPL(:,JN,1:2)=ZSDIVP(:,ISE:ISE+1)
  ENDDO
  IF (KM > 0) THEN

    !               Inversion of a symmetric matrix.

    CALL MXTURS(NSMAX+1-KM,NFLEVG,NFLEVG,II,&
     & SIHEG(1,ISO+1,1),SIHEG(1,ISO+1,2),SIHEG(1,ISO+1,3),&
     & ZSDIVPL,ZSPDIVPL)
  ELSE

    !               Inversion of a non-symmetric matrix.

    CALL MXTURE(NSMAX+1-KM,NFLEVG,NFLEVG,II,-2,.TRUE.,&
     & SIHEG(1,ISO+1,1),SIHEG(1,ISO+1,2),SIHEG(1,ISO+1,3),&
     & ZSDIVPL,ZSPDIVPL)
    CALL MXTURE(NSMAX+1-KM,NFLEVG,NFLEVG,II,3,.FALSE.,&
     & SIHEG(1,ISO+1,1),SIHEG2(1,ISO2+1,2),&
     & SIHEG2(1,ISO2+1,3),ZSDIVPL,ZSPDIVPL)
  ENDIF

  DO JN=KM,NSMAX
    ISE=KSTA+2*(JN-KM)
    ZSPDIVP(:,ISE:ISE+1)=ZSPDIVPL(:,JN,1:2)
  ENDDO
ELSE

  !             Case with NO Stretching :

  IF (LIMPF) THEN

    !               Solve complex pentadiagonal system

    CALL SIMPLICO(KM,NSMAX,NFLEVG,NFLEVG,ZALPHA(KM),&
     & ZDENIM(KM),ZFPLUS(KM),ZFMINUS(KM),SIVP,RLAPDI(0),&
     & ZBDT2,ZSDIVP,ZSPDIVP)
  ELSE

    !                 Inversion of a diagonal system (Helmholtz equation)
    !                 --> (SIMI*DIVprim(t+dt)).

    DO JLEV=1,NFLEVG
      DO JSP=KSTA,KEND
        ZSPDIVP(JLEV,JSP)=ZSDIVP(JLEV,JSP)&
         & /(1.0_JPRB-ZBDT2*SIVP(JLEV)*RLAPDI(NVALUE(JSP+IOFF)))
      ENDDO
    ENDDO
  ENDIF
ENDIF

!           Vertical eigenmodes space --> current space.

CALL MXMAOP(SIMO,1,NFLEVG,ZSPDIVP,1,NFLEVG,ZZSPDIVG,1,&
 & NFLEVG,NFLEVG,NFLEVG,ISPCOL)

IF (LSIDG) THEN

  !           ZSPDIV=(DIVprim(t+dt)) --> ZSPDIVG=(GM**2 * DIVprim(t+dt)) .

  ZSDIVPL(:,:,:)=0.0_JPRB
  ZSPDIVPL(:,:,:)=0.0_JPRB

  !           Reorganisation of ZSDIVP (Back to the USSR)

  DO JN=KM,NSMAX
    ISE=KSTA+2*(JN-KM)
    ZSDIVPL(:,JN,1:2)=ZZSPDIVG(:,ISE:ISE+1)
  ENDDO

  !        ZSPDIV=(DIVprim(t+dt)) --> ZPSPDIVG=(GMBAR**2 * DIVprim(t+dt)).

  CALL MXPTMA(NSMAX+1-KM,NFLEVG,NFLEVG,II,SCGMAP(ISO+1,1),&
   & SCGMAP(ISO+1,2),SCGMAP(ISO+1,3),&
   & SCGMAP(ISO+1,2),SCGMAP(ISO+1,3),&
   & ZSDIVPL,ZSPDIVPL)

  !           Reorganisation of ZSPDIVPL

  DO JN=KM,NSMAX
    ISE=KSTA+2*(JN-KM)
    ZHELP(:,ISE:ISE+1)=ZSPDIVPL(:,JN,1:2)
  ENDDO
ELSE

  !       ZSPDIV=(DIVprim(t+dt)) --> ZSPDIVG=(GMBAR**2 * DIVprim(t+dt)) .

  DO JLEV=1,NFLEVG
    DO JSP=KSTA,KEND
      ZHELP(JLEV,JSP)=ZZSPDIVG(JLEV,JSP)*RSTRET*RSTRET
    ENDDO
  ENDDO

ENDIF

!       If LSIDG:
!         (GM**2 * DIVprim(t+dt)) --> [ tau * (GM**2 * DIVprim(t+dt)) ]
!                                 and [  nu * (GM**2 * DIVprim(t+dt)) ]
!       or if not LSIDG:
!         (GMBAR**2 * DIVprim(t+dt)) --> [ tau * (GMBAR**2 * DIVprim(t+dt)) ]
!                                        and [  nu * (GMBAR**2 * DIVprim(t+dt)) ]

CALL SITNU(1,NFLEVG,ZHELP,ZST,ZSP,ISPCOL)

!*       2.5  Increment Temperature and surface pressure

DO JLEV=1,NFLEVG
  DO JSP=KSTA,KEND
    ZZSPTG(JLEV,JSP)=ZZSPTG(JLEV,JSP)-ZBDT*ZST(JLEV,JSP)
  ENDDO
ENDDO

DO JSP=KSTA,KEND
  ZZSPSPG(JSP)=ZZSPSPG(JSP)-ZBDT*ZSP(JSP)
ENDDO
```

```
  !*       2.6  Increment vorticity

  IF (LIMPF) THEN
    DO JLEV=1,NFLEVG
      IF (KM == 0) THEN
        DO JN=2,NSMAX
          IRSP=KSTA+(JN-KM)*2
          ZZSPVORG(JLEV,IRSP)=ZZSPVORG(JLEV,IRSP)&
           & -ZDENIM(JN)*ZFMINUS(JN)*ZZSPDIVG(JLEV,IRSP-2)
        ENDDO
        DO JN=1,NSMAX-1
          IRSP=KSTA+(JN-KM)*2
          ZZSPVORG(JLEV,IRSP)=ZZSPVORG(JLEV,IRSP)&
           & -ZDENIM(JN)*ZFPLUS(JN)*ZZSPDIVG(JLEV,IRSP+2)
        ENDDO
      ELSE
        DO JN=KM+1,NSMAX
          IRSP=KSTA+(JN-KM)*2
          IMSP=IRSP+1
          ZZSPVORG(JLEV,IRSP)=ZZSPVORG(JLEV,IRSP)&
           & -ZDENIM(JN)*ZFMINUS(JN)*(ZZSPDIVG(JLEV,IRSP-2)&
           & -ZALPHA(JN)*ZZSPDIVG(JLEV,IMSP-2))
          ZZSPVORG(JLEV,IMSP)=ZZSPVORG(JLEV,IMSP)&
           & -ZDENIM(JN)*ZFMINUS(JN)*(ZZSPDIVG(JLEV,IMSP-2)&
           & +ZALPHA(JN)*ZZSPDIVG(JLEV,IRSP-2))
        ENDDO
        DO JN=KM,NSMAX-1
          IRSP=KSTA+(JN-KM)*2
          IMSP=IRSP+1
          ZZSPVORG(JLEV,IRSP)=ZZSPVORG(JLEV,IRSP)&
           & -ZDENIM(JN)*ZFPLUS(JN)*(ZZSPDIVG(JLEV,IRSP+2)&
           & -ZALPHA(JN)*ZZSPDIVG(JLEV,IMSP+2))
          ZZSPVORG(JLEV,IMSP)=ZZSPVORG(JLEV,IMSP)&
           & -ZDENIM(JN)*ZFPLUS(JN)*(ZZSPDIVG(JLEV,IMSP+2)&
           & +ZALPHA(JN)*ZZSPDIVG(JLEV,IRSP+2))
        ENDDO
      ENDIF
    ENDDO
  ENDIF

ENDIF

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

!*       3.    MEMORY TRANSFER.
!              ----------------

IF (LL3D) THEN
  IF (LIMPF.OR.LVERFLT) PSPVORG(1:NFLEVG,KSTA:KEND)=&
   & ZZSPVORG(1:NFLEVG,KSTA:KEND)
  PSPDIVG(1:NFLEVG,KSTA:KEND)=ZZSPDIVG(1:NFLEVG,KSTA:KEND)
  PSPTG  (1:NFLEVG,KSTA:KEND)=ZZSPTG  (1:NFLEVG,KSTA:KEND)
  PSPSPG (         KSTA:KEND)=ZZSPSPG (         KSTA:KEND)
ENDIF

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

!*       4.    MEMORY TRANSFER FOR DDH SI.
!              ----------------

IF (LLDDHSI) THEN
  IF (LIMPF.OR.LVERFLT) PSPTNDSI_VORG(1:NFLEVG,KSTA:KEND)=&
   & PSPTNDSI_VORG(1:NFLEVG,KSTA:KEND) + PSPVORG(1:NFLEVG,KSTA:KEND)
  PSPTNDSI_DIVG(1:NFLEVG,KSTA:KEND)=PSPTNDSI_DIVG(1:NFLEVG,KSTA:KEND)&
   & + PSPDIVG(1:NFLEVG,KSTA:KEND)
  PSPTNDSI_TG(1:NFLEVG,KSTA:KEND)=PSPTNDSI_TG(1:NFLEVG,KSTA:KEND)&
   & + PSPTG(1:NFLEVG,KSTA:KEND)
ENDIF

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

!*       5.    VERTICAL FILTER.
!              ----------------

! 2-del-eta filter:  eps*f_(i-1) +(1-2*eps)*f_i + eps*f_(i+1)
! on VOR and DIV for levs 1 to NLEVVF

IF (LLDOVERFLT) THEN

  ZCT=TDT/(1._JPRB+TDT)
  PSPVORG(1,KSTA:KEND)=  ZZSPVORG(1,KSTA:KEND)&
   & +ZCT*REPSVFVO*(ZZSPVORG(2,KSTA:KEND)-ZZSPVORG(1,KSTA:KEND))
  PSPDIVG(1,KSTA:KEND)=  ZZSPDIVG(1,KSTA:KEND)&
   & +ZCT*REPSVFDI*(ZZSPDIVG(2,KSTA:KEND)-ZZSPDIVG(1,KSTA:KEND))
  DO JLEV=2,NLEVVF
    PSPVORG(JLEV,KSTA:KEND)=  ZZSPVORG(JLEV,KSTA:KEND)&
     & + ZCT*REPSVFVO*(ZZSPVORG(JLEV-1,KSTA:KEND)&
     & -2.0_JPRB*ZZSPVORG(JLEV,KSTA:KEND) &
     & + ZZSPVORG(JLEV+1,KSTA:KEND))
    PSPDIVG(JLEV,KSTA:KEND)=  ZZSPDIVG(JLEV,KSTA:KEND)&
     & + ZCT*REPSVFDI*(ZZSPDIVG(JLEV-1,KSTA:KEND)&
     & -2.0_JPRB*ZZSPDIVG(JLEV,KSTA:KEND) &
     & + ZZSPDIVG(JLEV+1,KSTA:KEND))
  ENDDO

ENDIF

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

IF (LHOOK) CALL DR_HOOK('SPCSI',1,ZHOOK_HANDLE)
END SUBROUTINE SPCSI
```

CY36R4:

```
SUBROUTINE SPCSI(&
 ! --- INPUT -------------------------------------------------------------
 & CDCONF,KM,KMLOC,KSTA,KEND,LDONEM,PSPAUXG, &
 ! --- INOUT -------------------------------------------------------------
 & PSPVORG,PSPDIVG,PSPTG,PSPSPG)

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

USE YOMMP    , ONLY : MYSETV, MYSETW, NPROCM, NPTRSV, NPTRSVF, NSPEC2V, NSPEC2VF
USE YOMDIM   , ONLY : NFLEVG, NSMAX
USE YOMCT0   , ONLY : NCONF, LSLAG, LTWOTL, LSITRIC
USE YOMDYN   , ONLY : SCGMAP, SIRDEL, SIMO, SIMI, SIVP, SIHEG, SIHEG2, &
 & SITRICA, SITRICB, SITRICC, TDT, BETADT, VESL, LSIDG, XIDT, LIMPF
USE YOMLAP   , ONLY : NVALUE, NSEOL, RLAPDI, RLAPIN
USE YOMGEM   , ONLY : RSTRET

!**** *SPCSI* - SPECTRAL SPACE SEMI-IMPLICIT COMPUTATIONS FOR HYD MODEL.

!     Purpose.
!     --------

!**   Interface.
!     ----------
!        *CALL* *SPCSI(..)

!        Explicit arguments :  CDCONF  - configuration (see doc.)
!        --------------------  KM      - Zonal wavenumber
!                              KMLOC   - Zonal wavenumber (DM-local numbering)
!                              KSTA    - First column processed
!                              KEND    - Last column processed
!                              LDONEM  - T if only one m if processed
!                              PSPVORG - Vorticity columns
!                              PSPDIVG - Divergence columns
!                              PSPTG   - Temperature columns
!                              PSPSPG  - Surface Pressure

!     Method.
!     -------

!     Externals.
!     ----------

!     Reference.
!     ----------
!        ECMWF Research Department documentation of the IFS

!     Author.
!     -------
!        Mats Hamrud and Philippe Courtier  *ECMWF*

!     Modifications.
!     --------------
!        Original : 87-11-24 (before 1997 spcsi.F was part of spc.F)
!        Modified 91-05-21 by A.Lasserre-Bigorry (scalar variables)
!        Modified DEC 1992 by K. YESSAD: decentering factor in semi-lag scheme.
!        Modified JAN 1994 by D GIARD: CDCONF='I'.
!        Modified JAN 1994 by K. YESSAD: semi-implicit computations
!           with not reduced divergence + new formulation of variable
!           mesh horizontal diffusion + multitasking on m instead of n.
!        Modified 95-02-06 by C. Temperton: semi-implicit Coriolis term,
!           real/imag parts stored in conventional order in work arrays
!        K. YESSAD (MARCH 1995):
!            - Optimisation of semi-implicit scheme if variable decentering
!              and remove of IMSL routines.
!            - Use new array NSEO (of YOMLAP) to define ISO.
!        Modified 95-06-06 by L.Isaksen : Reordering of spectral arrays
!        Modified 96-04-16 by C. Temperton: "alternative averaging" in
!                                           two-time-level scheme
!        Modified 96-05-16 by C. Temperton: new semi-implicit solver
!        Modified OCT 1996 by K. YESSAD: removal of variable uncentering.
!        Modified 12/12/96 by L.Isaksen: Merged with message passing
!                                        version. Renamed and recoded.
!        C. Temperton 97-02-24: modified first timestep when XIDT>0
!        R. El Khatib : 01-08-07 Pruning options
!        M.Hamrud      01-Oct-2003 CY28 Cleaning
!        N.Wedi        08-Mar-2005 remove mass correction
!        K.Yessad 09-Dec-2004: move mass correction in SPCMASCOR + cleanings.
!        K. Yessad 15-May-2006: memory optimisations for stretched geometry
!   N. Wedi and K. Yessad (Jan 2008): different dev for NH model and PC scheme
!        T.Wilhelmsson 09-09-25: Remove LFULLM requirement for LIMPF
!     ----------------------------------------------------------------------

IMPLICIT NONE

CHARACTER(LEN=1)  ,INTENT(IN)    :: CDCONF
INTEGER(KIND=JPIM),INTENT(IN)    :: KM
INTEGER(KIND=JPIM),INTENT(IN)    :: KMLOC
INTEGER(KIND=JPIM),INTENT(IN)    :: KSTA
INTEGER(KIND=JPIM),INTENT(IN)    :: KEND
LOGICAL           ,INTENT(IN)    :: LDONEM
REAL(KIND=JPRB)   ,INTENT(IN)    :: PSPAUXG(:,:)
REAL(KIND=JPRB)   ,INTENT(INOUT) :: PSPVORG(:,:)
REAL(KIND=JPRB)   ,INTENT(INOUT) :: PSPDIVG(:,:)
REAL(KIND=JPRB)   ,INTENT(INOUT) :: PSPTG(:,:)
REAL(KIND=JPRB)   ,INTENT(INOUT) :: PSPSPG(:)

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

REAL(KIND=JPRB), ALLOCATABLE :: ZSDIVP (:,:)
REAL(KIND=JPRB), ALLOCATABLE :: ZSPDIVP(:,:)

REAL(KIND=JPRB) :: ZSDIV  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZHELP  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZST    (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZSP    (       KSTA:KEND)

REAL(KIND=JPRB) :: ZSDIVPL (NFLEVG,KM:NSMAX,2)
REAL(KIND=JPRB) :: ZSPDIVPL(NFLEVG,KM:NSMAX,2)


INTEGER(KIND=JPIM) :: II, IN, IOFF, ISO, ISO2,&
 & ISE, ISPCOL, ISTART, JLEV, JN, JSP
```

```
LOGICAL :: LL3D,LLDOSI

REAL(KIND=JPRB) :: ZBDT, ZBDT2
REAL(KIND=JPRB) :: ZHOOK_HANDLE

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

INTERFACE
#include "mxmaop.h"
END INTERFACE

#include "mxptma.h"
#include "mxture.h"
#include "mxturs.h"
#include "sigam.intfb.h"
#include "spcimpfsolve.intfb.h"
#include "sitnu.intfb.h"
#include "tricsi.intfb.h"

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

IF (LHOOK) CALL DR_HOOK('SPCSI',0,ZHOOK_HANDLE)

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

LL3D=(NCONF == 1.OR.NCONF == 131.OR.NCONF == 302.OR.NCONF == 401 &
 & .OR.NCONF == 501.OR.NCONF == 601.OR.NCONF == 801)
LLDOSI=LL3D.AND.(CDCONF == 'A'.OR.CDCONF == 'I'.OR. &
 & CDCONF == 'S' .OR. CDCONF == 'T')

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

!*       2.    SEMI-IMPLICIT SPECTRAL COMPUTATIONS.
!              ------------------------------------

IF (LLDOSI) THEN

  ALLOCATE(ZSDIVP(NFLEVG,MAX(NSPEC2V,NSPEC2VF)))
  ALLOCATE(ZSPDIVP(NFLEVG,MAX(NSPEC2V,NSPEC2VF)))

  !*        2.1  Preliminary initialisations.

  IF (LDONEM) THEN
    IOFF=NPTRSVF(MYSETV)-1
  ELSE
    IOFF=NPTRSV(MYSETV)-1
  ENDIF
  ISPCOL=KEND-KSTA+1

  IF (LSLAG) THEN
    IF (LTWOTL) THEN
      ZBDT=(1.0_JPRB+VESL)*0.5_JPRB*TDT*BETADT*(1.0_JPRB+XIDT)
    ELSE
      ZBDT=(1.0_JPRB+VESL)*0.5_JPRB*TDT*BETADT
    ENDIF
  ELSE
    ZBDT=0.5_JPRB*TDT*BETADT
  ENDIF
  ZBDT2=(ZBDT*RSTRET)**2

  !*        2.3  Computes right-hand side of Helmholtz equation.

  IF( .NOT.LDONEM ) CALL GSTATS(1655,0)
  CALL SIGAM(1,NFLEVG,ZSDIV,PSPTG(:,KSTA:KEND),PSPSPG(KSTA:KEND),ISPCOL)
  IF( .NOT.LDONEM ) CALL GSTATS(1655,1)

  IF( .NOT.LDONEM ) CALL GSTATS(1656,0)
  IF (LSIDG) THEN
    IF (KM > 0) THEN
!$OMP PARALLEL DO PRIVATE(JSP,JLEV,IN)
      DO JSP=KSTA,KEND
        DO JLEV=1,NFLEVG
          IN=NVALUE(JSP+IOFF)
          ZSDIV(JLEV,JSP)=RLAPIN(IN)*PSPDIVG(JLEV,JSP)&
           & -ZBDT*ZSDIV(JLEV,JSP)
        ENDDO
      ENDDO
!$OMP END PARALLEL DO
    ELSE
!$OMP PARALLEL DO PRIVATE(JSP,JLEV,IN)
      DO JSP=KSTA,KEND
        DO JLEV=1,NFLEVG
          IN=NVALUE(JSP+IOFF)
          ZSDIV(JLEV,JSP)=PSPDIVG(JLEV,JSP)&
           & -ZBDT*RLAPDI(IN)*ZSDIV(JLEV,JSP)
        ENDDO
      ENDDO
!$OMP END PARALLEL DO
    ENDIF
  ELSE

    ! Case of No Stretching
!$OMP PARALLEL DO PRIVATE(JSP,JLEV,IN)
    DO JSP=KSTA,KEND
      DO JLEV=1,NFLEVG
        IN=NVALUE(JSP+IOFF)
        ZSDIV(JLEV,JSP)=PSPDIVG(JLEV,JSP)-ZBDT*RLAPDI(IN)*ZSDIV(JLEV,JSP)
      ENDDO
    ENDDO
!$OMP END PARALLEL DO
  ENDIF

  !        Add [F] * result to rhs of Helmholtz equation

  IF (LIMPF) THEN
!$OMP PARALLEL DO PRIVATE(JSP,JLEV)
    DO JSP=KSTA,KEND
      DO JLEV=1,NFLEVG
        ZSDIV(JLEV,JSP)=ZSDIV(JLEV,JSP) + PSPAUXG(JLEV,JSP)
      ENDDO
    ENDDO
!$OMP END PARALLEL DO
  ENDIF
  IF( .NOT.LDONEM ) CALL GSTATS(1656,1)
```

```
!*        2.4  Solve Helmholtz equation

IF (LSITRIC) THEN

  ! Tricky: The PE that owns wavenumber 0 starts at (m,n)=(1,1), so istart=3

  IF ((KM == 0.AND.LDONEM).OR.(MYSETW == NPROCM(0).AND..NOT.LDONEM)) THEN
    ISTART=KSTA+2
  ELSE
    ISTART=KSTA
  ENDIF
  CALL TRICSI(PSPDIVG(:,ISTART:KEND),ZSDIV(:,ISTART:KEND),&
   & SITRICA,SITRICB,SITRICC,SIRDEL,ISTART,KEND)
ELSE

  !           Current space --> vertical eigenmodes space.

  IF( .NOT.LDONEM ) CALL GSTATS(2020,0)
  CALL MXMAOP(SIMI,1,NFLEVG,ZSDIV,1,NFLEVG,ZSDIVP(:,KSTA:KEND),1,NFLEVG,&
   & NFLEVG,NFLEVG,ISPCOL)
  IF( .NOT.LDONEM ) CALL GSTATS(2020,1)

  IF (LSIDG) THEN

    !             Inversion of two tridiagonal systems (Helmholtz equation)
    !                --> (SIMI*DIVprim(t+dt)).

    !             Reorganisation of divergence

    ISO=NSEOL(KMLOC)
    ISO2=0
    II=MIN(KM,1)+1
    ZSDIVPL(:,:,:)=0.0_JPRB
    ZSPDIVPL(:,:,:)=0.0_JPRB

    DO JN=KM,NSMAX
      ISE=KSTA+2*(JN-KM)
      ZSDIVPL(:,JN,1:2)=ZSDIVP(:,ISE:ISE+1)
    ENDDO
    IF (KM > 0) THEN

      !               Inversion of a symmetric matrix.

      CALL MXTURS(NSMAX+1-KM,NFLEVG,NFLEVG,II,&
       & SIHEG(1,ISO+1,1),SIHEG(1,ISO+1,2),SIHEG(1,ISO+1,3),&
       & ZSDIVPL,ZSPDIVPL)
    ELSE

      !               Inversion of a non-symmetric matrix.

      CALL MXTURE(NSMAX+1-KM,NFLEVG,NFLEVG,II,-2,.TRUE.,&
       & SIHEG(1,ISO+1,1),SIHEG(1,ISO+1,2),SIHEG(1,ISO+1,3),&
       & ZSDIVPL,ZSPDIVPL)
      CALL MXTURE(NSMAX+1-KM,NFLEVG,NFLEVG,II,3,.FALSE.,&
       & SIHEG(1,ISO+1,1),SIHEG2(1,ISO2+1,2),&
       & SIHEG2(1,ISO2+1,3),ZSDIVPL,ZSPDIVPL)
    ENDIF

    DO JN=KM,NSMAX
      ISE=KSTA+2*(JN-KM)
      ZSPDIVP(:,ISE:ISE+1)=ZSPDIVPL(:,JN,1:2)
    ENDDO
  ELSE

    !             Case with NO Stretching :

    IF (LIMPF) THEN

      !               Solve complex pentadiagonal system

      CALL SPCIMPFSOLVE(LDONEM,ZSDIVP,ZSPDIVP)

    ELSE

      !                 Inversion of a diagonal system (Helmholtz equation)
      !                 --> (SIMI*DIVprim(t+dt)).

      DO JSP=KSTA,KEND
        DO JLEV=1,NFLEVG
          ZSPDIVP(JLEV,JSP)=ZSDIVP(JLEV,JSP)&
           & /(1.0_JPRB-ZBDT2*SIVP(JLEV)*RLAPDI(NVALUE(JSP+IOFF)))
        ENDDO
      ENDDO
    ENDIF
  ENDIF

  !           Vertical eigenmodes space --> current space.

  IF( .NOT.LDONEM ) CALL GSTATS(2020,0)
  CALL MXMAOP(SIMO,1,NFLEVG,ZSPDIVP(:,KSTA:KEND),1,NFLEVG,PSPDIVG(:,KSTA:KEND),1,&
   & NFLEVG,NFLEVG,NFLEVG,ISPCOL)
  IF( .NOT.LDONEM ) CALL GSTATS(2020,1)
ENDIF
IF (LSIDG) THEN

  !           ZSPDIV=(DIVprim(t+dt)) --> ZSPDIVG=(GM**2 * DIVprim(t+dt)) .

  ZSDIVPL(:,:,:)=0.0_JPRB
  ZSPDIVPL(:,:,:)=0.0_JPRB

  !           Reorganisation of ZSDIVP (Back to the USSR)

  DO JN=KM,NSMAX
    ISE=KSTA+2*(JN-KM)
    ZSDIVPL(:,JN,1:2)=PSPDIVG(:,ISE:ISE+1)
  ENDDO

  !        ZSPDIV=(DIVprim(t+dt)) --> ZPSPDIVG=(GMBAR**2 * DIVprim(t+dt)).

  CALL MXPTMA(NSMAX+1-KM,NFLEVG,NFLEVG,II,SCGMAP(ISO+1,1),&
   & SCGMAP(ISO+1,2),SCGMAP(ISO+1,3),&
   & SCGMAP(ISO+1,2),SCGMAP(ISO+1,3),&
   & ZSDIVPL,ZSPDIVPL)
```

```
    !           Reorganisation of ZSPDIVPL

    DO JN=KM,NSMAX
      ISE=KSTA+2*(JN-KM)
      ZHELP(:,ISE:ISE+1)=ZSPDIVPL(:,JN,1:2)
    ENDDO
  ELSE

    !       ZSPDIV=(DIVprim(t+dt)) --> ZSPDIVG=(GMBAR**2 * DIVprim(t+dt)) .

    IF( .NOT.LDONEM ) CALL GSTATS(1656,0)
!$OMP PARALLEL DO PRIVATE(JSP,JLEV)
    DO JSP=KSTA,KEND
      DO JLEV=1,NFLEVG
        ZHELP(JLEV,JSP)=PSPDIVG(JLEV,JSP)*RSTRET*RSTRET
      ENDDO
    ENDDO
!$OMP END PARALLEL DO
    IF( .NOT.LDONEM ) CALL GSTATS(1656,1)

  ENDIF

  !       If LSIDG:
  !         (GM**2 * DIVprim(t+dt)) --> [ tau * (GM**2 * DIVprim(t+dt)) ]
  !                                 and [  nu * (GM**2 * DIVprim(t+dt)) ]
  !       or if not LSIDG:
  !         (GMBAR**2 * DIVprim(t+dt)) --> [ tau * (GMBAR**2 * DIVprim(t+dt)) ]
  !                                    and [  nu * (GMBAR**2 * DIVprim(t+dt)) ]

  IF( .NOT.LDONEM ) CALL GSTATS(1657,0)
  CALL SITNU(1,NFLEVG,ZHELP,ZST,ZSP,ISPCOL)
  IF( .NOT.LDONEM ) CALL GSTATS(1657,1)

  !*       2.5  Increment Temperature and surface pressure

  IF( .NOT.LDONEM ) CALL GSTATS(1656,0)
!$OMP PARALLEL DO PRIVATE(JSP,JLEV)
  DO JSP=KSTA,KEND
    DO JLEV=1,NFLEVG
      PSPTG(JLEV,JSP)=PSPTG(JLEV,JSP)-ZBDT*ZST(JLEV,JSP)
    ENDDO
    PSPSPG(JSP)=PSPSPG(JSP)-ZBDT*ZSP(JSP)
  ENDDO
!$OMP END PARALLEL DO
  IF( .NOT.LDONEM ) CALL GSTATS(1656,1)

  DEALLOCATE(ZSDIVP)
  DEALLOCATE(ZSPDIVP)

ENDIF

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

IF (LHOOK) CALL DR_HOOK('SPCSI',1,ZHOOK_HANDLE)
END SUBROUTINE SPCSI
```

We now examinate modifications introduced on both sides:
Modifications introduced between CY36 and CY36T2:

- Elimination of option LSITRIC=T, and of all the LSITRIC=T code.
- Add of DDH diagnostics (temporal tendencies) stored in new arrays, the name of which start by PSPTNDSI: this piece of code has limited locations in the code and requires three additional dummy arguments.

Modifications introduced between CY36 and CY36R4:

- Different modularisation of the code under LIMPF=T. A subset of this code has been moved out of SPCSI. Another subset of this code (calling SIMPLICO) has been put under the new routine SPCIMPFSOLVE. This modification is associated with an optimisation of the distributed memory treatment. An additional input array PSPAUXG appears.
- Some intermediate local arrays ZZSP[X]G have been removed (and also the corresponding memory transfers) and dummy arguments PSP[X]G are now used where ZZSP[X]G were used formerly.
- Code under LVERFLT (vertical filter) has been removed.
- OpenMp directives have been added (spread in the code).
- Lines IF(.NOT.LDONEM) CALL GSTATS(..) have been added (spread in the code).
- Inversion of some loops DO JSP and DO JLEV. JLEV loop becomes inner loop.

We can see that it is significantly easier to reintroduce the CY36T2 modifications on the top of the CY36R4 code than to do the reverse thing. So we start from the CY36R4 source and:

- we remove the option LSITRIC=T and all the LSITRIC=T code.
- we add DDH diagnostics (code under keys LLOPTDDH and LLDDHSI).

We can notice two things:

- we must reintroduce paragraph 1 (removed in CY36R4) to put the piece of code under LLOPTDDH (filling PSPTNDSI_[X]G).
- code under keys LLOPTDDH and LLDDHSI contain tests IF(LIMPF.OR.LVERFLT), and we should remember that option (and key) LVERFLT has been removed in CY36R4. Tests IF(LIMPF.OR.LVERFLT) must be rewritten IF(LIMPF).

Taking account of these remarks, the merged CY36T2R4 code becomes:

```
SUBROUTINE SPCSI(&
 ! --- INPUT -------------------------------------------------------------
 & CDCONF,KM,KMLOC,KSTA,KEND,LDONEM,PSPAUXG, &
 ! --- INOUT ------------------------------------------------------------
 & PSPVORG,PSPDIVG,PSPTG,PSPSPG, &
 & PSPTNDSI_VORG,PSPTNDSI_DIVG,PSPTNDSI_TG)

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

USE YOMMP     , ONLY : MYSETV, NPTRSV, NPTRSVF, NSPEC2V, NSPEC2VF
USE YOMDIM    , ONLY : NFLEVG, NSMAX
USE YOMCTO    , ONLY : NCONF, LSLAG, LTWOTL,
USE YOMDYN    , ONLY : SCGMAP, SIMO, SIMI, SIVP, SIHEG, SIHEG2, &
 & TDT, BETADT, VESL, LSIDG, XIDT, LIMPF
USE YOMLAP    , ONLY : NVALUE, NSEOL, RLAPDI, RLAPIN
USE YOMGEM    , ONLY : RSTRET
USE YOMLDDH   , ONLY : LRDDHDYN, LOPTSIDH

!**** *SPCSI* - SPECTRAL SPACE SEMI-IMPLICIT COMPUTATIONS FOR HYD MODEL.

!     Purpose.
!     --------

!**   Interface.
!     ----------
!        *CALL* *SPCSI(..)

!        Explicit arguments :  CDCONF  - configuration (see doc.)
!        --------------------  KM      - Zonal wavenumber
!                              KMLOC   - Zonal wavenumber (DM-local numbering)
!                              KSTA    - First column processed
!                              KEND    - Last column processed
!                              LDONEM  - T if only one m if processed
!                              PSPVORG - Vorticity columns
!                              PSPDIVG - Divergence columns
!                              PSPTG   - Temperature columns
!                              PSPSPG  - Surface Pressure
!                              PSPTNDSI_VORG  - [D vor/Dt]_SI
!                              PSPTNDSI_DIVG  - [D div/Dt]_SI
!                              PSPTNDSI_TG    - [D T/Dt]_SI

!     Method.
!     -------

!     Externals.
!     ----------

!     Reference.
!     ----------
!        ECMWF Research Department documentation of the IFS

!     Author.
!     -------
!        Mats Hamrud and Philippe Courtier  *ECMWF*

!     Modifications.
!     --------------
!        Original : 87-11-24 (before 1997 spcsi.F was part of spc.F)
!        Modified 91-05-21 by A.Lasserre-Bigorry (scalar variables)
!        Modified DEC 1992 by K. YESSAD: decentering factor in semi-lag scheme.
!        Modified JAN 1994 by D GIARD: CDCONF='I'.
```

```
!        Modified JAN 1994 by K. YESSAD: semi-implicit computations
!           with not reduced divergence + new formulation of variable
!           mesh horizontal diffusion + multitasking on m instead of n.
!        Modified 95-02-06 by C. Temperton: semi-implicit Coriolis term,
!           real/imag parts stored in conventional order in work arrays
!        K. YESSAD (MARCH 1995):
!            - Optimisation of semi-implicit scheme if variable decentering
!              and remove of IMSL routines.
!            - Use new array NSEO (of YOMLAP) to define ISO.
!        Modified 95-06-06 by L.Isaksen : Reordering of spectral arrays
!        Modified 96-04-16 by C. Temperton: "alternative averaging" in
!                                           two-time-level scheme
!        Modified 96-05-16 by C. Temperton: new semi-implicit solver
!        Modified OCT 1996 by K. YESSAD: removal of variable uncentering.
!        Modified 12/12/96 by L.Isaksen: Merged with message passing
!                                        version. Renamed and recoded.
!        C. Temperton 97-02-24: modified first timestep when XIDT>0
!        R. El Khatib : 01-08-07 Pruning options
!        M.Hamrud      01-Oct-2003 CY28 Cleaning
!        N.Wedi        08-Mar-2005 remove mass correction
!        K.Yessad 09-Dec-2004: move mass correction in SPCMASCOR + cleanings.
!        K. Yessad 15-May-2006: memory optimisations for stretched geometry
!   N. Wedi and K. Yessad (Jan 2008): different dev for NH model and PC scheme
!   K. Yessad (Aug 2009): remove LSITRIC option.
!   F. Voitus: add DDH diagnostics.
!   T.Wilhelmsson 09-09-25: Remove LFULLM requirement for LIMPF
!     ------------------------------------------------------------------

IMPLICIT NONE

CHARACTER(LEN=1)   ,INTENT(IN)    :: CDCONF
INTEGER(KIND=JPIM),INTENT(IN)     :: KM
INTEGER(KIND=JPIM),INTENT(IN)     :: KMLOC
INTEGER(KIND=JPIM),INTENT(IN)     :: KSTA
INTEGER(KIND=JPIM),INTENT(IN)     :: KEND
LOGICAL           ,INTENT(IN)     :: LDONEM
REAL(KIND=JPRB)    ,INTENT(IN)    :: PSPAUXG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPVORG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPDIVG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPTG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPSPG(:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPTNDSI_VORG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPTNDSI_DIVG(:,:)
REAL(KIND=JPRB)    ,INTENT(INOUT) :: PSPTNDSI_TG(:,:)

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

REAL(KIND=JPRB), ALLOCATABLE :: ZSDIVP (:,:)
REAL(KIND=JPRB), ALLOCATABLE :: ZSPDIVP(:,:)

REAL(KIND=JPRB) :: ZSDIV  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZHELP  (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZST    (NFLEVG,KSTA:KEND)
REAL(KIND=JPRB) :: ZSP    (       KSTA:KEND)

REAL(KIND=JPRB) :: ZSDIVPL (NFLEVG,KM:NSMAX,2)
REAL(KIND=JPRB) :: ZSPDIVPL(NFLEVG,KM:NSMAX,2)

INTEGER(KIND=JPIM) :: II, IN, IOFF, ISO, ISO2, ISE, ISPCOL, JLEV, JN, JSP

LOGICAL :: LL3D,LLDOSI,LLDDHSI,LLOPTDDH

REAL(KIND=JPRB) :: ZBDT, ZBDT2
REAL(KIND=JPRB) :: ZHOOK_HANDLE

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

INTERFACE
#include "mxmaop.h"
END INTERFACE

#include "mxptma.h"
#include "mxture.h"
#include "mxturs.h"
#include "sigam.intfb.h"
#include "spcimpfsolve.intfb.h"
#include "sitnu.intfb.h"

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

IF (LHOOK) CALL DR_HOOK('SPCSI',0,ZHOOK_HANDLE)

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

LL3D=(NCONF == 1.OR.NCONF == 131.OR.NCONF == 302.OR.NCONF == 401 &
 & .OR.NCONF == 501.OR.NCONF == 601.OR.NCONF == 801)
LLDOSI=LL3D.AND.(CDCONF == 'A'.OR.CDCONF == 'I'.OR. &
 & CDCONF == 'S' .OR. CDCONF == 'T')
LLDDHSI=LL3D.AND.LRDDHDYN
LLOPTDDH=LLDDHSI.AND.LOPTSIDH

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

!*       1.    MEMORY TRANSFER.
!              ----------------

IF (LLOPTDDH) THEN
  IF (LIMPF) PSPTNDSI_VORG(1:NFLEVG,KSTA:KEND)=&
    & - PSPVORG(1:NFLEVG,KSTA:KEND)
  PSPTNDSI_DIVG(1:NFLEVG,KSTA:KEND)=-PSPDIVG(1:NFLEVG,KSTA:KEND)
  PSPTNDSI_TG  (1:NFLEVG,KSTA:KEND)=-PSPTG  (1:NFLEVG,KSTA:KEND)
  !the case of surface pressure has not been treated yet
ENDIF

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

!*       2.    SEMI-IMPLICIT SPECTRAL COMPUTATIONS.
!              ------------------------------------

IF (LLDOSI) THEN

  ALLOCATE(ZSDIVP(NFLEVG,MAX(NSPEC2V,NSPEC2VF)))
  ALLOCATE(ZSPDIVP(NFLEVG,MAX(NSPEC2V,NSPEC2VF)))

  !*        2.1  Preliminary initialisations.

  IF (LDONEM) THEN
    IOFF=NPTRSVF(MYSETV)-1
  ELSE
    IOFF=NPTRSV(MYSETV)-1
  ENDIF
  ISPCOL=KEND-KSTA+1
```

```
  IF (LSLAG) THEN
    IF (LTWOTL) THEN
      ZBDT=(1.0_JPRB+VESL)*0.5_JPRB*TDT*BETADT*(1.0_JPRB+XIDT)
    ELSE
      ZBDT=(1.0_JPRB+VESL)*0.5_JPRB*TDT*BETADT
    ENDIF
  ELSE
    ZBDT=0.5_JPRB*TDT*BETADT
  ENDIF
  ZBDT2=(ZBDT*RSTRET)**2

  !*       2.3  Computes right-hand side of Helmholtz equation.

  IF( .NOT.LDONEM ) CALL GSTATS(1655,0)
  CALL SIGAM(1,NFLEVG,ZSDIV,PSPTG(:,KSTA:KEND),PSPSPG(KSTA:KEND),ISPCOL)
  IF( .NOT.LDONEM ) CALL GSTATS(1655,1)

  IF( .NOT.LDONEM ) CALL GSTATS(1656,0)
  IF (LSIDG) THEN
    IF (KM > 0) THEN
!$OMP PARALLEL DO PRIVATE(JSP,JLEV,IN)
      DO JSP=KSTA,KEND
        DO JLEV=1,NFLEVG
          IN=NVALUE(JSP+IOFF)
          ZSDIV(JLEV,JSP)=RLAPIN(IN)*PSPDIVG(JLEV,JSP)&
           & -ZBDT*ZSDIV(JLEV,JSP)
        ENDDO
      ENDDO
!$OMP END PARALLEL DO
    ELSE
!$OMP PARALLEL DO PRIVATE(JSP,JLEV,IN)
      DO JSP=KSTA,KEND
        DO JLEV=1,NFLEVG
          IN=NVALUE(JSP+IOFF)
          ZSDIV(JLEV,JSP)=PSPDIVG(JLEV,JSP)&
           & -ZBDT*RLAPDI(IN)*ZSDIV(JLEV,JSP)
        ENDDO
      ENDDO
!$OMP END PARALLEL DO
    ENDIF
  ELSE

    ! Case of No Stretching
!$OMP PARALLEL DO PRIVATE(JSP,JLEV,IN)
    DO JSP=KSTA,KEND
      DO JLEV=1,NFLEVG
        IN=NVALUE(JSP+IOFF)
        ZSDIV(JLEV,JSP)=PSPDIVG(JLEV,JSP)-ZBDT*RLAPDI(IN)*ZSDIV(JLEV,JSP)
      ENDDO
    ENDDO
!$OMP END PARALLEL DO
  ENDIF

  !        Add [F] * result to rhs of Helmholtz equation

  IF (LIMPF) THEN
!$OMP PARALLEL DO PRIVATE(JSP,JLEV)
    DO JSP=KSTA,KEND
      DO JLEV=1,NFLEVG
        ZSDIV(JLEV,JSP)=ZSDIV(JLEV,JSP) + PSPAUXG(JLEV,JSP)
      ENDDO
    ENDDO
!$OMP END PARALLEL DO
  ENDIF
  IF( .NOT.LDONEM ) CALL GSTATS(1656,1)

  !*       2.4  Solve Helmholtz equation

  !          Current space --> vertical eigenmodes space.

  IF( .NOT.LDONEM ) CALL GSTATS(2020,0)
  CALL MXMAOP(SIMI,1,NFLEVG,ZSDIV,1,NFLEVG,ZSDIVP(:,KSTA:KEND),1,NFLEVG,&
   & NFLEVG,NFLEVG,ISPCOL)
  IF( .NOT.LDONEM ) CALL GSTATS(2020,1)

  IF (LSIDG) THEN

    !             Inversion of two tridiagonal systems (Helmholtz equation)
    !                --> (SIMI*DIVprim(t+dt)).

    !             Reorganisation of divergence

    ISO=NSEOL(KMLOC)
    ISO2=0
    II=MIN(KM,1)+1
    ZSDIVPL(:,:,:)=0.0_JPRB
    ZSPDIVPL(:,:,:)=0.0_JPRB

    DO JN=KM,NSMAX
      ISE=KSTA+2*(JN-KM)
      ZSDIVPL(:,JN,1:2)=ZSDIVP(:,ISE:ISE+1)
    ENDDO
    IF (KM > 0) THEN

      !               Inversion of a symmetric matrix.

      CALL MXTURS(NSMAX+1-KM,NFLEVG,NFLEVG,II,&
       & SIHEG(1,ISO+1,1),SIHEG(1,ISO+1,2),SIHEG(1,ISO+1,3),&
       & ZSDIVPL,ZSPDIVPL)
    ELSE

      !               Inversion of a non-symmetric matrix.

      CALL MXTURE(NSMAX+1-KM,NFLEVG,NFLEVG,II,-2,.TRUE.,&
       & SIHEG(1,ISO+1,1),SIHEG(1,ISO+1,2),SIHEG(1,ISO+1,3),&
       & ZSDIVPL,ZSPDIVPL)
      CALL MXTURE(NSMAX+1-KM,NFLEVG,NFLEVG,II,3,.FALSE.,&
       & SIHEG(1,ISO+1,1),SIHEG2(1,ISO2+1,2),&
       & SIHEG2(1,ISO2+1,3),ZSDIVPL,ZSPDIVPL)
    ENDIF

    DO JN=KM,NSMAX
      ISE=KSTA+2*(JN-KM)
      ZSPDIVP(:,ISE:ISE+1)=ZSPDIVPL(:,JN,1:2)
    ENDDO
  ELSE

    !             Case with NO Stretching :

    IF (LIMPF) THEN

      !               Solve complex pentadiagonal system
```

```
      CALL SPCIMPFSOLVE(LDONEM,ZSDIVP,ZSPDIVP)

    ELSE

      !                 Inversion of a diagonal system (Helmholtz equation)
      !                 --> (SIMI*DIVprim(t+dt)).

      DO JSP=KSTA,KEND
        DO JLEV=1,NFLEVG
          ZSPDIVP(JLEV,JSP)=ZSDIVP(JLEV,JSP)&
           & /(1.0_JPRB-ZBDT2*SIVP(JLEV)*RLAPDI(NVALUE(JSP+IOFF)))
        ENDDO
      ENDDO
    ENDIF
  ENDIF

  !           Vertical eigenmodes space --> current space.

  IF( .NOT.LDONEM ) CALL GSTATS(2020,0)
  CALL MXMAOP(SIMO,1,NFLEVG,ZSPDIVP(:,KSTA:KEND),1,NFLEVG,PSPDIVG(:,KSTA:KEND),1,&
   & NFLEVG,NFLEVG,NFLEVG,ISPCOL)
  IF( .NOT.LDONEM ) CALL GSTATS(2020,1)

  IF (LSIDG) THEN

    !           ZSPDIV=(DIVprim(t+dt)) --> ZSPDIVG=(GM**2 * DIVprim(t+dt)) .

    ZSDIVPL(:,:,:)=0.0_JPRB
    ZSPDIVPL(:,:,:)=0.0_JPRB

    !           Reorganisation of ZSDIVP (Back to the USSR)

    DO JN=KM,NSMAX
      ISE=KSTA+2*(JN-KM)
      ZSDIVPL(:,JN,1:2)=PSPDIVG(:,ISE:ISE+1)
    ENDDO

    !        ZSPDIV=(DIVprim(t+dt)) --> ZPSPDIVG=(GMBAR**2 * DIVprim(t+dt)).

    CALL MXPTMA(NSMAX+1-KM,NFLEVG,NFLEVG,II,SCGMAP(ISO+1,1),&
     & SCGMAP(ISO+1,2),SCGMAP(ISO+1,3),&
     & SCGMAP(ISO+1,2),SCGMAP(ISO+1,3),&
     & ZSDIVPL,ZSPDIVPL)

    !           Reorganisation of ZSPDIVPL

    DO JN=KM,NSMAX
      ISE=KSTA+2*(JN-KM)
      ZHELP(:,ISE:ISE+1)=ZSPDIVPL(:,JN,1:2)
    ENDDO
  ELSE

    !       ZSPDIV=(DIVprim(t+dt)) --> ZSPDIVG=(GMBAR**2 * DIVprim(t+dt)) .

    IF( .NOT.LDONEM ) CALL GSTATS(1656,0)
!$OMP PARALLEL DO PRIVATE(JSP,JLEV)
    DO JSP=KSTA,KEND
      DO JLEV=1,NFLEVG
        ZHELP(JLEV,JSP)=PSPDIVG(JLEV,JSP)*RSTRET*RSTRET
      ENDDO
    ENDDO
!$OMP END PARALLEL DO
    IF( .NOT.LDONEM ) CALL GSTATS(1656,1)

  ENDIF

  !       If LSIDG:
  !         (GM**2 * DIVprim(t+dt)) --> [ tau * (GM**2 * DIVprim(t+dt)) ]
  !                                 and [  nu * (GM**2 * DIVprim(t+dt)) ]
  !       or if not LSIDG:
  !         (GMBAR**2 * DIVprim(t+dt)) --> [ tau * (GMBAR**2 * DIVprim(t+dt)) ]
  !                                    and [  nu * (GMBAR**2 * DIVprim(t+dt)) ]

  IF( .NOT.LDONEM ) CALL GSTATS(1657,0)
  CALL SITNU(1,NFLEVG,ZHELP,ZST,ZSP,ISPCOL)
  IF( .NOT.LDONEM ) CALL GSTATS(1657,1)

  !*       2.5  Increment Temperature and surface pressure

  IF( .NOT.LDONEM ) CALL GSTATS(1656,0)
!$OMP PARALLEL DO PRIVATE(JSP,JLEV)
  DO JSP=KSTA,KEND
    DO JLEV=1,NFLEVG
      PSPTG(JLEV,JSP)=PSPTG(JLEV,JSP)-ZBDT*ZST(JLEV,JSP)
    ENDDO
    PSPSPG(JSP)=PSPSPG(JSP)-ZBDT*ZSP(JSP)
  ENDDO
!$OMP END PARALLEL DO
  IF( .NOT.LDONEM ) CALL GSTATS(1656,1)

  DEALLOCATE(ZSDIVP)
  DEALLOCATE(ZSPDIVP)

ENDIF

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

!*       3.    MEMORY TRANSFER FOR DDH SI.
!              ---------------

IF (LLDDHSI) THEN
  IF (LIMPF) PSPTNDSI_VORG(1:NFLEVG,KSTA:KEND)=&
   & PSPTNDSI_VORG(1:NFLEVG,KSTA:KEND) + PSPVORG(1:NFLEVG,KSTA:KEND)
  PSPTNDSI_DIVG(1:NFLEVG,KSTA:KEND)=PSPTNDSI_DIVG(1:NFLEVG,KSTA:KEND)&
   & + PSPDIVG(1:NFLEVG,KSTA:KEND)
  PSPTNDSI_TG(1:NFLEVG,KSTA:KEND)=PSPTNDSI_TG(1:NFLEVG,KSTA:KEND)&
   & + PSPTG(1:NFLEVG,KSTA:KEND)
ENDIF

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

IF (LHOOK) CALL DR_HOOK('SPCSI',1,ZHOOK_HANDLE)
END SUBROUTINE SPCSI
```

∗ Conclusion:

To sum-up, we can say that merging is not an easy task and knowledge about scientific aspects of the code is required (this is not only an informatical task). In particuliar:

- even if an automatic source merging performs well without any conflict, we cannot always guarantee that the result is good; in some cases additional manual modifications may be required.
- side effects may be encountered on some callers of modified routines.
- side effects may be encountered on some callees of modified routines.
- merging may be very difficult to do under the source library (a lot of conflicts), and can be easy to solve provided the different contributions are understood (on informatical and scientific aspects).