L'ANALYSE OFFLINE DE L'EAU TOTALE DU SOL DANS LE MODÈLE ALADIN

INTRODUCTION

L'objectif de ce travail est d'implémenter une procédure d'analyse variationnelle 2D-VAR simplifiée off-line utilisant le modèle de surface découplé SURFEX qui soit capable d'assimiler différents types de données (obs. Conventionnelles, satellites, précipitations, flux radiatifs) ainsi qu'analyser différentes variables prognostiques (Wg, W_2, T_s, T_p, Biomasse).

Notre objectif est de créer un programme suffisamment flexible permettant ainsi de choisir la longueur de la fenêtre d'assimilation, la fréquence d'assimilation, la variable de contrôle (essentiellement w_2, T_2, biomasse), les observations (T2m, HU2m de CANARI ou de VARPACK), wg(ERS, ASCAT, SMOS) et LAI. Un tel programme doit concilier les besoins de GMAP ainsi que ceux de l'équipe MC2 du CNRM (amélioration de la prévision de température et de l'humidité, dans ISBA standard ainsi que l'estimation de la biomasse dans ISBA-Ags).

2D-VAR SIMPLIFIE

Le domaine de travail choisi dans un premier temps est le domaine ALADIN-FRANCE et la période d'intérêt est celle de l'été 2006 (du 01 Juillet au 31 Août 2006). L'algorithme est donc de type variationnel visant à ajuster les simulations de Température et humidité à 2m du modèle ISBA-2L aux observations (horaires) analysées par CANARI-DIAGPACK. L'objectif du programme d'assimilation est de résoudre l'équation d'analyse suivante:

$$x^a = x^b + B\mathbf{H}^T(\mathbf{H}B\mathbf{H}^T + R)^{-1}(Yo-\mathbf{H}(x^a))=x^b + K(Yo - H(x^a))$$

avec,

x est la variable de contrôle à analyser et elle peut être (W_2, T_2, Biomasse)

H est linéaire et c'est le Jacobien de l'opérateur d'observation

H est l'opérateur d'observation

B est la matrice de variance covariance du modèle de surface

R est la matrice d'erreurs d'observations

Yo est le vecteur d'observation

$(Yo - H(x^b))$ est le vecteur d'innovation

K est la matrice de gain.

La méthode d'estimation de H est décrite dans Hess 2001 et dans Balsamo et al 2004. La simplicité de l'approche réside dans la non utilisation de l'adjoint ou du modèle linéaire tangent pour estimer le jacobien, mais on suppose que H est linéarisable et on détermine son Jacobien par différences finies (dT2m/dw2,dHU2m/dW2) en calculant l'impact d'une perturbation de la variable à analyser sur la variable observée à l'instant où on dispose d'observations.

MISE EN PLACE PRATIQUE

La version 2.0 de SURFEX off-line a été utilisée et forcée par les champs prévus au dernier niveau du modèle ALADIN (Niveau 46 du modèle). Ce niveau est situé à peu près à 17m du sol. Un calcul plus rigoureux de la hauteur de ce niveau est envisageable, mais on considère que 17m est une bonne approximation de cette hauteur.

La physiographie opérationnelle d'ALADIN a été retenue ainsi que sa géométrie (projection Conforme plan (Mercator, Lambertienne ou stéréographique polaire) ayant une grille régulière en x et en y (même résolution qu'ALADIN opérationnel). Les points de grille SURFEX sont des « tiles » purs nature (ayant 1 seul patch). À la sortie du PGD, l'orographie du modèle atmosphèrique remplace celle calculée par les routines PGD de SURFEX et qui pourrait différer de quelques dizaines de mètres. C'est le script corrige_zs.sh qui permet une telle manipulation.

La namelist de SURFEX a été modifiée de telle sorte qu'on prend en compte les champs physiographiques d'ALADIN, les fichiers de texture du sol (pourcentage de sable et d'argile), les constantes Rgl et Gamma et enfin, la topographie GTOPO30.

Nos choix sont motivés par un but essentiel, reproduire avec la surface externalisée SURFEX les prévisions du modèle ISBA opérationnel. Pour se faire, il faut avoir la même version scientifique d'ISBA et aussi exactement la même géométrie et les même constantes physiographiques. Or, une première différence essentielle est que dans SURFEX, chaque point de grille est représenté par plusieurs types de végétation, en n'utilisant pas ecoclimap, les fichiers "data" de végétation basse et haute sont complémentaires. Donc si un point de grille comporte de la végétation haute et de la végétation basse, le champ résultant sera une moyenne des paramètres en ce point. On obtient ainsi en sortie du PGD des différences au niveau des constantes RGL et GAMMA qui entrent en jeux dans le calcul de la résistance de surface utilisée pour la transpiration (Noilhan et Mahfouf 1996).

COHERENCE OFFLINE-INLINE

Un forçage ALADIN-FRANCE toutes les trois heures est désarchivé et transformé en fichier netCDF. Les premiers tests de SURFEX offline comparés à ISBA opérationnel donnent des différences assez prononcées. Les principales sources de différences ont été identifiées comme suit:

- Coefficient de la capacité thermique de la végétation Cv différents ($2\ 10^{-5}$ pour SURFEX et $0.8\ 10^{-5}$ pour ISBA oper)
- Coefficient de la conductivité thermique du sol nu C_{GMAX} différents ($2\ 10^{-5}$ dans SURFEX et $0.8\ 10^{-5}$ dans ISBA oper)
- Coefficients d'échanges turbulents C_D et C_H différents (au niveau des routines surface_aero_cond.f90 et surface_cd.f90 voir note interne de Patrick Le Moigne du 27 Avril 2007)

Ce premier travail de « cohérence » entre SURFEX offline et ISBA opérationnel nous a permis d'obtenir des flux, des variables dans le sol et des diagnostics à 2 mètres compatibles.

Il a été décidé de relancer un run d'ALADIN sur les 2 mois de l'expérience (Juillet et Août) et récuperer les prévisions horaires du modèle jusqu'à 6 heures d'échéance et produire ainsi les forçages en format netCDF sur le domaine ALADIN-FRANCE car le forçage horaire donne des résultats meilleurs qu'un forçage toutes les trois heures. Ce forçage horaire est maintenant disponible pour des périodes récentes, on pourra envisager après la mise en opérationnel d'une telle procédure d'analyse bénéficier de ces prévisions horaires archivées. L'exécutable SURFEX OFFLINE utilisé durant cette expérience comporte les modifications citées ci-dessus en plus de la correction d'un bug dans le programme principal offline.f90 dans lequel on récupère les prévisions SURFEX décalées d'un pas de temps en écrivant les résultats puis en incrémentant le temps. Les températures et humidité à 2m obtenues dans la version initiale sont décalées par rapport à la réalité de 300 secondes (le pas de temps SURFEX). Cette erreur est corrigée et donc l'éxécutable

OFFLINE que j'ai utilisé dans mes simulations comporte trois nouveaux programmes (offline.f90, surface_aero_cond.f90 et surface_cd.f90) en plus des nouvelles valeurs de Cv, C_D, C_H.

CHOIX ALGORITHMIQUES

La deuxième partie du travail a concerné la mise en place d'une procédure d'analyse 2D-VAR de l'humidité totale du sol. La stratégie adoptée est celle d'externaliser la procédure d'analyse et de la rendre complètement indépendante du modèle de surface utilisé. On n'aura pas à modifier les routines de SURFEX. Le module OFFLINE de SURFEX sera utilisé tel qu'il est actuellement. La procédure d'analyse varassim.f90 sera compilée en créant un lien vers les différents modules et routines de SURFEX.

Au cours de la procédure d'analyse, le programme varassim.f90 tournera en 3 modes :

1. Mode perturbé (LPRT = T) dans lequel on perturbe la variable de contrôle d'une perturbation dont la taille est définie en namelist.
2. Mode de lecture (LSIM = T) dans lequel on peut lire les observations simulées par le modèle (du mode perturbé ou du mode de référence) et sauvegarder les variables lues dans des fichiers ASCII.
3. Mode d'analyse (LPRT = F et LSIM = F) dans lequel on effectue l'analyse en lisant les observations, les runs perturbés et de référence, en calculant le jacobien de l'opérateur d'observation et enfin en calculant les incréments d'analyse pour produire le nouveau champ analysé par un transport du début de la fenêtre d'assimilation jusqu'à la fin de cette fenêtre.

L'inversion de la matrice ($HBH^T + R$) est effectuée par deux procédures de CHOLESKY (choldc.f90 et cholsl.f90) ce qui, la rend indépendante du type de calculateur utilisé.

La phase de lecture des fichiers d'observations qui sont écris dans un format ASCII issu du programme EditField (edf) permettra d'identifier le nombre d'observations disponibles et donc initialisera la taille des matrices qui seront utilisées par la suite dans l'analyse. Le programme permet ainsi une flexibilité au niveau des observations, on pourra ainsi utiliser un seul réseau d'observation, comme on peut utiliser dans l'analyse suivante, 6 réseaux d'observations. Une telle flexibilité permettra en plus de résoudre le problème de la fréquence des observations liée à la nature de ces observations (une fréquence horaire pour T_{2M} et HU_{2M}, une fréquence de 3 jours pour Ws et une fréquence de 10 jours pour LAI)

CHOIX DES PARAMÈTRES DE LA MÉTHODE

Ce programme d'analyse a été développé et testé initialement pour l'humidité totale du sol (W_2) en assimilant les températures et humidité à 2 mètres.

L'écart type de l'erreur d'observation en T2m = σ_{T2m} = 1.0 °K et σ_{HU2m} = 10%

L'écart type d'erreur du modèle est σ_B = 0.1 en Soil Wetness Index (Douville et al. (1998) ECMWF Technical Memo). Intuitivement, avec une telle valeur de σ_B on autorise une correction de l'ordre de 10mm dans le réservoir profond du sol dans le modèle ISBA.

Différentes tailles de perturbations ont été choisies, on a retenu pour cette étude la taille de 10e-3 du SWI. Les perturbations étant effectuées directement dans le fichier PREP.txt, donc en eau volumique, il a donc fallu dans le programme d'analyse multiplier l'écart type d'erreur du modèle σ_B par ($W_{fc} - W_{wilt}$) pour chaque point de grille en lisant à chaque fois le pourcentage d'argile.$\sigma_B = 0.1 * (W_{fc} - W_{wilt})$

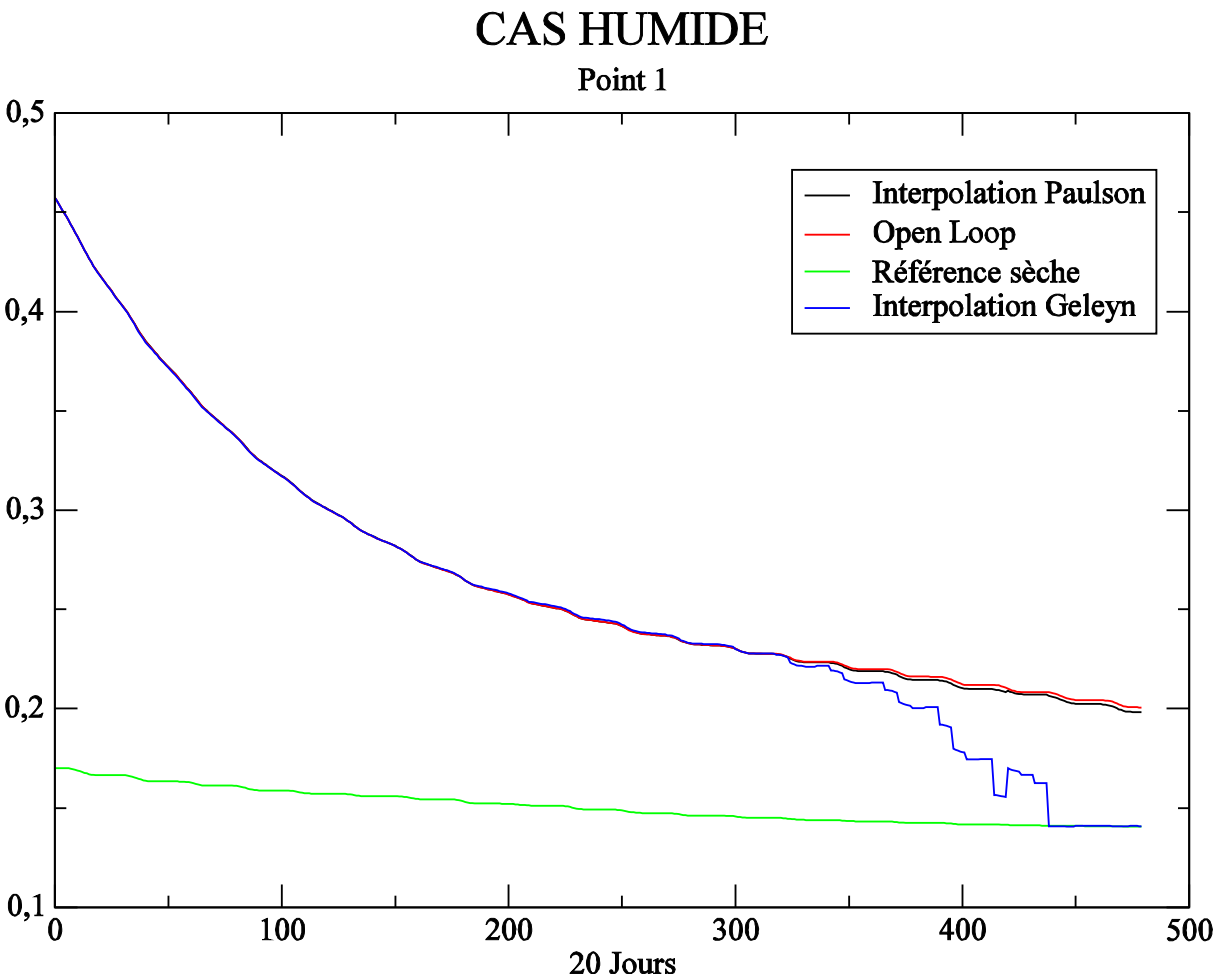
VALIDATION DE LA METHODE

Pour valider la méthode, on a travaillé dans un cadre d'observations simulées permettant de définir une référence vers laquelle on veut converger. De plus on a pris un domaine constitué de 9 points de grilles ayant des propriétés physiographiques très proches.

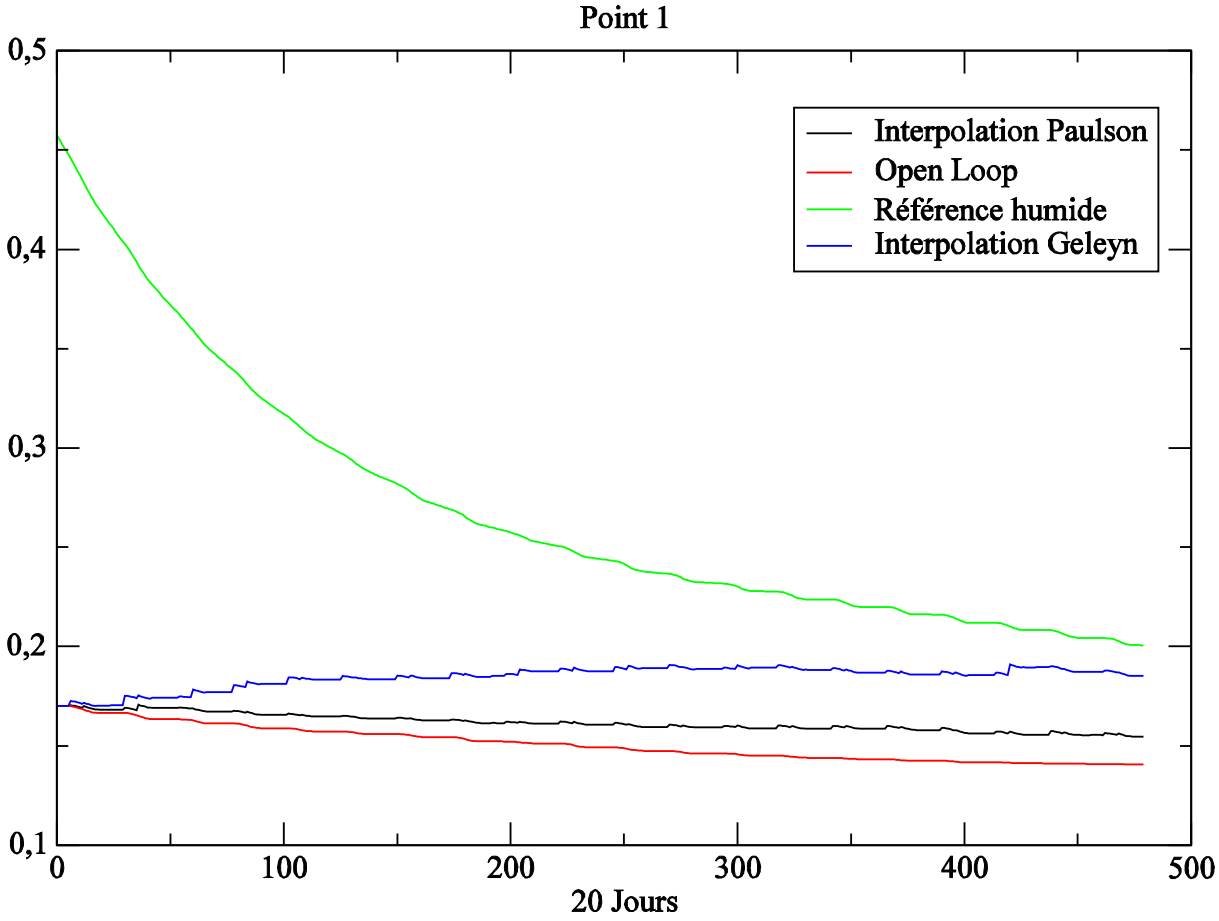
On a alors considéré deux cas de figures, une référence sèche (eau volumique initiale ~ 0.17) puis une référence humide (eau volumique initiale ~ 0.9). Les températures et humidité relatives à 2 mètres horaires sont utilisées comme « pseudo-observations » pour les expériences d'assimilation. Elles sont diagnostiquées au moyen de deux méthodes d'interpolation verticale dans la couche limite de surface (entre le sol et le niveau situé à 17m) suivant la formulation de Geleyn (1988) (N2M=2) ou bien selon la méthode des flux de Paulson (N2M=1). Une seconde intégration dite « Open Loop », consiste à initialiser le sol par l'état inverse et à laisser évoluer le schéma de surface SURFEX sans faire de corrections aux variables dans le sol. Les intégrations commencent le 01 juillet 2006 pendant une période de 20 jours.

RESULTATS

La validation de la procédure dans le cas où on part d'un état sec vers une référence (des observations similés) humide montre une convergence pour les deux méthodes avec une convergence plus prononcée pour l'opérateur d'observation issu de la méthode d'interpolation de Geleyn.



Dans le cas humide l'analyse se basant sur l'opérateur d'observation issu de l'interpolation de Paulson n'a pas pu converger. Elle est restée très proche de la simulation Open-Loop où le modèle seul essaie de ramener l'état du sol vers un état de référence.



CONCLUSIONS ET PERSPECTIVES

Une procédure flexible d'analyse de surface a été mise au point et validé sur la surface externalisée SURFEX, forcée par le modèle atmosphérique ALADIN. Cette première étape du travail a montré l'importance du choix de l'opérateur d'observation qui est dans le cas de l'analyse de l'humidité totale du sol est ou bien l'interpolation Geleyn ou bien l'interpolation de Paulson.

Bien que l'opérateur Geleyn semble converger plus rapidement, il sera préférable d'utiliser l'autre opérateur. Ceci est due au fait que la méthode des flux pour l'interpolation des T2m et HU2m peut se passer de la connaissance de la Ts et de Ws ces deux champs dépendront du patch utilisé et de la présence ou pas de la végétation et c'est ce qui correspond à une T_{NK} et HU_{NK} uniques ce qui pose un vrai problème pour la détermination des température et humidité à deux mètres. Alors que la méthode des flux elle dépendra uniquement du dernier niveau du modèle et des flux. On considère qu'à 17 m le modèle est indépendant des petites structures de la surface. On obtient alors une meilleure estimation des champs à 2 mètres.

Pour la suite de ce travail on essayera de:

1. Installer Surfex sur le calculateur IBM de l'INM.
2. Régler la méthode d'interpolation des flux.
3. Appliquer l'analyse sur tout le domaine ALADIN-FRANCE et ALADIN-TUNISIE.
4. Étudier les sensibilités à la perturbation de l'eau, typiquement les composantes h_1 et h_2 du Jacobien de l'opérateur d'observation.
5. Tracer des composantes de la matrice de gain.
6. Étudier l'effet des précipitations sur l'analyse.

Article à soumettre au

Quarterly Journal of the Royal Meteorological Society

(QJRMS)

Article à soumis au journal

TELLUS

(Tellus A)

Manuscript ID TeA-07-06-0073

Rapport Technique

Deuxième partie du stage

Annexe 1

Code source de la nouvelle routine d'analyse de surface

```
PROGRAM VARASSIM
! ------------------------------------------------

AUTHOR  Karim BERGAOUI
        INM Tunis
        30 Juin 2007

USE MODD_TYPE_DATE_SURF
USE MODI_READ_SURF
USE MODI_WRITE_SURF
USE MODI_OL_ALLOC_ATM
USE MODD_OL_FILEID
USE MODI_INIT_IO_SURF_n
USE MODI_END_IO_SURF_n
USE MODI_OPEN_NAMELIST
USE MODI_CLOSE_NAMELIST
USE MODI_TEST_NAM_VAR_SURF
USE MODE_POS_SURF
USE MODD_IO_SURF_ASC,ONLY : CFILEIN
USE MODD_CSTS,       ONLY : XDAY
USE MODI_CONVERT_COVER_FRAC
USE MODI_GET_SIZE_FULL_n
USE MODI_READ_COVER_n
USE MODD_SURF_ATM_n,      ONLY :  CSEA,      CWATER,      CTOWN,      CNATURE, &
                                  XSEA,      XWATER,      XTOWN,      XNATURE,&
                                  NSIZE_SEA, NSIZE_WATER, NSIZE_TOWN,
NSIZE_NATURE, &
                                  NR_SEA,    NR_WATER,    NR_TOWN,&
                                  NR_NATURE, XCOVER,NDIM_FULL, NSIZE_FULL, &
                                  NDIM_NATURE, NDIM_SEA, NDIM_WATER, NDIM_TOWN
!
! -------------------------------------------------------------
!
IMPLICIT NONE
INTEGER                                 :: NVAR,NDIM,NOBSTYPE
CHARACTER(LEN=28)                       :: YNAMELIST = 'OPTIONS.nam
'
!
!    Declarations of local variables
!
CHARACTER(LEN=3), PARAMETER             :: YINIT     = 'ALL'
CHARACTER(LEN=6)                        :: YPROGRAM  = 'ASCII '
INTEGER                                 :: IYEAR                      ! current
year (UTC)
INTEGER                                 :: IMONTH                     ! current
month (UTC)
INTEGER                                 :: IDAY                       ! current
day (UTC)
INTEGER                                 :: IHOUR                      ! current
day (UTC)
REAL                                    :: ZTIME                      ! current
time since start of the run (s)
REAL                                    :: PTSTEP_OUTPUT,PTSTEP_FORC  ! OUPUT
step, Duration and FORCING Step
INTEGER                                 :: IRESP,PATCH_NUMBER         ! return
code
INTEGER                                 :: ILOBS                      !
Namelist unit number
TYPE (DATE_TIME)                        :: TTIME                      ! Current
date and time
INTEGER                                 :: INB_FORC                   ! NB
Forcing step
INTEGER                                 :: NBOUTPUT                   ! Number
of time step
```

```
INTEGER                                 :: ISTEP                        !
REAL,DIMENSION(:,:,:,:),ALLOCATABLE     :: YF                           ! Vector
of model observations
REAL,DIMENSION(:,:,:,:),ALLOCATABLE     :: XI                           ! Vector
of control variables
REAL,DIMENSION(:,:,:,:),ALLOCATABLE     :: HO                           !
Jacobian of observation operator
REAL,DIMENSION(:,:,:,:),ALLOCATABLE     :: HOT                          !
Transpose of HO
REAL,DIMENSION(:,:,:,:),ALLOCATABLE     :: R                            !
covariance matrix of observation errors
REAL,DIMENSION(:,:,:,:),ALLOCATABLE     :: B                            !
background error covariance matrix
REAL,DIMENSION(:,:,:),ALLOCATABLE       :: YO                           ! vector
of observations
REAL,DIMENSION(:,:,:),ALLOCATABLE       :: ZCLAY                        !
Pourcentage of clay (varies from 0 to 1)
REAL,DIMENSION(:,:,:),ALLOCATABLE       :: COFSWI                       ! The
difference (Wfc - Wwilt)
REAL,DIMENSION(:,:,:),ALLOCATABLE       :: ZEPS                         ! The
perturbation amplitude
REAL,DIMENSION(:,:,:),ALLOCATABLE       :: XINCR                        !
Analysis increment
REAL,DIMENSION(:,:),ALLOCATABLE         :: XT2M,XHU2M                   !
Simulated temperature and relative humidity
REAL,DIMENSION(:),ALLOCATABLE           :: VECT                         ! The
analysed variable
REAL,DIMENSION(:),ALLOCATABLE           :: XERROBS                      !
Observational standard deviation
CHARACTER(LEN=200)                      :: NMFILE_CANARI                ! Name of
the observation, perturbed or reference file
CHARACTER(LEN=50)                       :: CMONTH,CDAY,CYEAR,CHOUR
INTEGER                                 :: IND
!
! Local Matrix for Analysis calculation
!
REAL,DIMENSION(:,:,:),ALLOCATABLE       :: K1
REAL,DIMENSION(:,:,:,:,:),ALLOCATABLE   :: HBHT
REAL,DIMENSION(:,:,:),ALLOCATABLE       :: ZX,ZB,ZP
REAL,DIMENSION(:,:,:,:),ALLOCATABLE     :: YFTEMP
REAL,DIMENSION(:,:),ALLOCATABLE         :: MATTEMP3
REAL,DIMENSION(:,:,:),ALLOCATABLE       :: YOWR
INTEGER                                 :: I,J,K,kk,L,NOBS
LOGICAL                                 :: LPRT                         ! Running
VARASSIM in a perturbation mode
LOGICAL                                 :: LSIM                         ! Running
VARASSIM in a reading mode
REAL                                    :: TPRT                         ! The
perturbation amplitude
REAL                                    :: XSIGMAB                      !
covariance matrix of background errors
CHARACTER(LEN=3)                        :: YVAR                         ! Name of
the control variable (syntax of surfex in PREP.txt file )
CHARACTER(LEN=50)                       :: PREFIX                       ! The
prefix of the control variable (in PREP.txt file)
CHARACTER(LEN=50)                       :: HCOMMENT
REAL,ALLOCATABLE                        :: XSIMISBA(:,:)                ! must
replace XT2M and XHU2M !!!!!
INTEGER                                 :: ILUOUT                       ! ascii
output unit number
INTEGER                                 :: ILUNAM                       ! namelist
unit number
LOGICAL                                 :: GFOUND                       ! return
logical when reading namelist
```

```
NAMELIST/NAM_IO_VARASSIM/LPRT, LSIM
NAMELIST/NAM_OBS/NOBSTYPE, NDIM, NVAR, TPRT
NAMELIST/NAM_VAR/YVAR, PREFIX
!
!
! -----------------------------------------------------------------------
!
! Initialisation of  WG2 file name (pour la lecture des observations)
!
!ILOBS=55
!NOBSTYPE = 2
!ALLOCATE (XERROBS(NOBSTYPE))
!NDIM = 2
!NVAR = 1
!TPRT = 0.001
!YVAR = 'WG2'
!PREFIX    = 'X_Y_WG2 (m3/m3)                                    '
!XERROBS(1) = 1.0
!XERROBS(2) = 0.1
!XSIGMAB = 0.1
!
CALL GOTO_SURFEX(1)
!
!*      0.1   Open namelist
!
CALL OPEN_NAMELIST('ASCII ','SURF  ',ILUNAM,YNAMELIST)
CALL POSNAM(ILUNAM,'NAM_IO_VARASSIM',GFOUND,ILUOUT)
IF (GFOUND) READ (UNIT=ILUNAM,NML=NAM_IO_VARASSIM)
CALL CLOSE_NAMELIST('ASCII ',ILUNAM)
CALL OPEN_NAMELIST('ASCII ','SURF  ',ILUNAM,YNAMELIST)
CALL POSNAM(ILUNAM,'NAM_OBS',GFOUND,ILUOUT)
IF (GFOUND) READ (UNIT=ILUNAM,NML=NAM_OBS)
CALL CLOSE_NAMELIST('ASCII ',ILUNAM)
CALL OPEN_NAMELIST('ASCII ','SURF  ',ILUNAM,YNAMELIST)
CALL POSNAM(ILUNAM,'NAM_VAR',GFOUND,ILUOUT)
IF (GFOUND) READ (UNIT=ILUNAM,NML=NAM_VAR)
CALL CLOSE_NAMELIST('ASCII ',ILUNAM)
!
!       1.    Initializations
!
ILOBS=55
ALLOCATE (XERROBS(NOBSTYPE))
XERROBS(1) = 1.0
XERROBS(2) = 0.1

CALL INI_CSTS
!
CALL INI_DATA_COVER
!
!       ascii file handling
!
CFILEIN = 'PREP.txt'        ! output of PREP
!
!       read grid dimension for allocation
!
CALL INIT_IO_SURF_n(YPROGRAM,'FULL  ','SURF  ','READ ')
!
! find current time
!
CALL READ_SURF(YPROGRAM,'DTCUR',TTIME,IRESP)
!
! reading grid characteristics to perform nature mask
!
CALL READ_SURF(YPROGRAM,'SEA   ',CSEA   ,IRESP)
```

```
CALL READ_SURF(YPROGRAM,'WATER ',CWATER ,IRESP)
CALL READ_SURF(YPROGRAM,'NATURE',CNATURE,IRESP)
CALL READ_SURF(YPROGRAM,'TOWN  ',CTOWN  ,IRESP)
!
CALL READ_SURF(YPROGRAM,'DIM_FULL  ',NDIM_FULL,  IRESP)
CALL READ_SURF(YPROGRAM,'DIM_SEA   ',NDIM_SEA,   IRESP)
CALL READ_SURF(YPROGRAM,'DIM_NATURE',NDIM_NATURE,IRESP)
CALL READ_SURF(YPROGRAM,'DIM_WATER ',NDIM_WATER, IRESP)
CALL READ_SURF(YPROGRAM,'DIM_TOWN  ',NDIM_TOWN,  IRESP)
!
!  Get number of points on this proc
!
CALL GET_SIZE_FULL_n(YPROGRAM,NDIM_FULL,NSIZE_FULL)
CALL READ_COVER_n(YPROGRAM)
CALL END_IO_SURF_n(YPROGRAM)
!
! Perform masks (only nature used)
!
ALLOCATE(XSEA   (NDIM_FULL))
ALLOCATE(XNATURE(NDIM_FULL))
ALLOCATE(XWATER (NDIM_FULL))
ALLOCATE(XTOWN  (NDIM_FULL))
!
CALL CONVERT_COVER_FRAC(XCOVER,XSEA,XNATURE,XTOWN,XWATER)
!
NSIZE_NATURE    = COUNT(XNATURE(:) > 0.0)
NSIZE_TOWN      = COUNT(XTOWN(:)   > 0.0)
NSIZE_WATER     = COUNT(XWATER(:)  > 0.0)
NSIZE_SEA       = COUNT(XSEA(:)    > 0.0)
!
ALLOCATE(NR_NATURE (NSIZE_NATURE))
ALLOCATE(NR_TOWN   (NSIZE_TOWN  ))
ALLOCATE(NR_WATER  (NSIZE_WATER ))
ALLOCATE(NR_SEA    (NSIZE_SEA   ))
!
CALL GET_1D_MASK( NSIZE_SEA,    NDIM_FULL, XSEA   , NR_SEA   )
CALL GET_1D_MASK( NSIZE_WATER,  NDIM_FULL, XWATER , NR_WATER )
CALL GET_1D_MASK( NSIZE_TOWN,   NDIM_FULL, XTOWN  , NR_TOWN  )
CALL GET_1D_MASK( NSIZE_NATURE, NDIM_FULL, XNATURE, NR_NATURE)
!
! Read number of patch and initial WG2
!
CALL INIT_IO_SURF_n(YPROGRAM,'NATURE','ISBA  ','READ ')
CALL READ_SURF(YPROGRAM,'PATCH_NUMBER',PATCH_NUMBER,IRESP)
ALLOCATE(XI(NSIZE_NATURE,PATCH_NUMBER,NDIM,NVAR))
ALLOCATE(VECT(NSIZE_NATURE))
ALLOCATE(ZCLAY(NSIZE_NATURE,PATCH_NUMBER,NVAR))
ALLOCATE(COFSWI(NSIZE_NATURE,PATCH_NUMBER,NVAR))
!
! READ LES T2M SIMULES
!
CALL READ_SURF(YPROGRAM,'WG2',XI(:,:,1,1),IRESP)
CALL READ_SURF(YPROGRAM,'CLAY',ZCLAY(:,:,1),IRESP)
DO L=1,NVAR
DO I=1,NSIZE_NATURE
  DO J=1,PATCH_NUMBER
       COFSWI(I,J,L)=0.001*(89.0467*((100.*ZCLAY(I,J,L))**0.3496)-
37.1342*((100.*ZCLAY(I,J,L))**0.5))
  ENDDO
ENDDO
ENDDO

IF ( LPRT ) THEN
DO I=1,NSIZE_NATURE
```

```
 VECT(I)=XI(I,1,1,1)+TPRT*XI(I,1,1,1)
ENDDO
CALL END_IO_SURF_n(YPROGRAM)
CALL INIT_IO_SURF_n(YPROGRAM,'NATURE','ISBA  ','WRITE')
CALL WRITE_SURF(YPROGRAM,YVAR,VECT,IRESP,HCOMMENT=PREFIX)
STOP
ENDIF
CALL END_IO_SURF_n(YPROGRAM)

!
! Time
!
IYEAR  = TTIME%TDATE%YEAR
IMONTH = TTIME%TDATE%MONTH
IDAY   = TTIME%TDATE%DAY
ZTIME  = TTIME%TIME
!
! find nb forcing time step
!
CALL READ_SURF('OFFLIN','NB_TIMESTP',INB_FORC,IRESP)
CALL READ_SURF('OFFLIN','FORC_TIME_STEP',PTSTEP_FORC,IRESP)
CALL READ_SURF('OFFLIN','OUTPUT_TIME_STEP',PTSTEP_OUTPUT,IRESP)
!
! calculate number of ouput time steps
!
NBOUTPUT = (PTSTEP_FORC*INB_FORC)/PTSTEP_OUTPUT-1
IF ( LSIM ) THEN
ALLOCATE(XT2M(NSIZE_NATURE,NBOUTPUT))
ALLOCATE(XHU2M(NSIZE_NATURE,NBOUTPUT))
CALL READ_SURF('OFFLIN','T2M',XT2M(:,:),IRESP)
CALL READ_SURF('OFFLIN','HU2M',XHU2M(:,:),IRESP)
OPEN (unit=111,file='OBSIMU',status='unknown')
do I=1,NSIZE_NATURE
! le 6 ici correspond à la sixiemme heure..il faut le parametrer pour ne plus
! fixer le nombre d'observations à utiliser
write(111,*) XT2M(I,6),XHU2M(I,6)
enddo
CLOSE(111)
STOP
ENDIF
!
! Je calcule le nombre de fichiers d'obs disponibles
!
NOBS=0
ZTIME=PTSTEP_OUTPUT
DO ISTEP=1,NBOUTPUT
   CALL ADD_FORECAST_TO_DATE_SURF(IYEAR,IMONTH,IDAY,ZTIME)
   ZTIME = ZTIME + PTSTEP_OUTPUT
   IHOUR=int (ZTIME/PTSTEP_OUTPUT - 1.)
   CALL TRANS_CHAINE(CMONTH,IMONTH,2)
   CALL TRANS_CHAINE(CDAY,IDAY,2)
   CALL TRANS_CHAINE(CYEAR,IYEAR,4)
   CALL TRANS_CHAINE(CHOUR,IHOUR,2)
   if (IHOUR.eq.0.or.IHOUR.eq.24.or.IHOUR.eq.48) then
      CHOUR='00'
   endif
!
   NMFILE_CANARI='CANARI_OPER_'
   NMFILE_CANARI=TRIM(NMFILE_CANARI)//'06'
   NMFILE_CANARI=TRIM(NMFILE_CANARI)//CMONTH
   NMFILE_CANARI=TRIM(NMFILE_CANARI)//CDAY
   NMFILE_CANARI=TRIM(NMFILE_CANARI)//'H'
   NMFILE_CANARI=TRIM(NMFILE_CANARI)//CHOUR
   NMFILE_CANARI=TRIM(NMFILE_CANARI)//'.DAT'
```

```
!
   OPEN(UNIT=ILOBS,FILE=NMFILE_CANARI,FORM='UNFORMATTED',STATUS='OLD',ERR=22)
   NOBS=NOBS+NOBSTYPE
   CLOSE(ILOBS)
22 CONTINUE
ENDDO
!
! Réinitialisation du temps
!
IYEAR  = TTIME%TDATE%YEAR
IMONTH = TTIME%TDATE%MONTH
IDAY   = TTIME%TDATE%DAY
ZTIME  = TTIME%TIME
!
! Allocation
!
! Perturbed simulations
ALLOCATE(YFTEMP(NDIM_FULL,NDIM,NOBS,NVAR))
ALLOCATE(YF(NSIZE_NATURE,NDIM,NOBS,NVAR))
ALLOCATE(ZEPS(NSIZE_NATURE,PATCH_NUMBER,NVAR))
! Initial values (to be analysed)
! Observations
ALLOCATE(YO(NSIZE_NATURE,NOBS,NVAR))
ALLOCATE(YOWR(NDIM_FULL,NOBS,NVAR))
! Temporary vectors used by the 2DVAR approach
ALLOCATE(ZB(NSIZE_NATURE,NOBS,NVAR))
ALLOCATE(HO(NSIZE_NATURE,PATCH_NUMBER,NOBS,NVAR))
ALLOCATE(HOT(NSIZE_NATURE,PATCH_NUMBER,NVAR,NOBS))
ALLOCATE(R(NSIZE_NATURE,NOBS,NOBS,NVAR))
ALLOCATE(B(NSIZE_NATURE,PATCH_NUMBER,NVAR,NVAR))
ALLOCATE(ZX(NSIZE_NATURE,NOBS,NVAR))
ALLOCATE(ZP(NSIZE_NATURE,NOBS,NVAR))
ALLOCATE(XINCR(NSIZE_NATURE,PATCH_NUMBER,NVAR))
ALLOCATE(K1(NOBS,NOBS,NVAR))
ALLOCATE(HBHT(NSIZE_NATURE,PATCH_NUMBER,NOBS,NOBS,NVAR))
!
! Initialisation
!
HO(:,:,:,:)=999.0 ! Linearized observation matrix
R(:,:,:,:)=0.0 ! Observation error matrix
B(:,:,:,:)=999.0 ! Background error matrix
YOWR(:,:,:)=999.0 ! Observation vector on the full grid to be written on file
ZB(:,:,:)=999.0 ! Innovation vector
!
! Delta initial WG2
!
DO L=1,NVAR
  WHERE(XI(:,:,1,L).NE.999.0)
      XI(:,:,2,L)=XI(:,:,1,L)+TPRT*XI(:,:,1,L)
      ZEPS(:,:,L)=XI(:,:,2,L)-XI(:,:,1,L)
  ENDWHERE
ENDDO
!
NOBS=0
ZTIME=PTSTEP_OUTPUT
DO ISTEP=1,NBOUTPUT
!
! Update date
!
CALL ADD_FORECAST_TO_DATE_SURF(IYEAR, IMONTH, IDAY, ZTIME)
ZTIME = ZTIME + PTSTEP_OUTPUT
IHOUR=INT(ZTIME/PTSTEP_OUTPUT - 1.)
!
           CALL TRANS_CHAINE(CMONTH,IMONTH,2)
```

```
            CALL TRANS_CHAINE(CDAY,IDAY,2)
            CALL TRANS_CHAINE(CYEAR,IYEAR,4)
            CALL TRANS_CHAINE(CHOUR,IHOUR,2)
            IF (IHOUR.EQ.0.OR.IHOUR.EQ.24.OR.IHOUR.EQ.48) THEN
               CHOUR='00'
            ENDIF
!
            NMFILE_CANARI='CANARI_OPER_'
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//'06'
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//CMONTH
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//CDAY
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//'H'
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//CHOUR
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//'.DAT'
            !
            ! Open WG2 observations file
            !
            OPEN(UNIT=ILOBS,FILE=NMFILE_CANARI,FORM='FORMATTED',STATUS='OLD',&
            ERR=10)
            NOBS=NOBS+NOBSTYPE
            !
            ! If it exists, read WG2 observation
            !
            DO I=1,NDIM_FULL
               READ (ILOBS,*) ((YOWR(I,J+NOBS-NOBSTYPE,L),J=1,NOBSTYPE),L=1,NVAR)
            ENDDO
            !
            ! Calculate perturbations at the date of observations and innovation
            !
            NMFILE_CANARI='OBSIMU_PERT_'
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//'06'
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//CMONTH
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//CDAY
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//'H'
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//CHOUR
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//'.DAT'

            OPEN(UNIT=25,FILE=NMFILE_CANARI,FORM='FORMATTED',STATUS='OLD')
            DO I=1,NDIM_FULL
               READ(25,*) ((YFTEMP(I,2,K+NOBS-NOBSTYPE,L),K=1,NOBSTYPE),L=1,NVAR)
            ENDDO
            CLOSE(25)

            NMFILE_CANARI='OBSIMU_REFR_'
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//'06'
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//CMONTH
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//CDAY
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//'H'
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//CHOUR
            NMFILE_CANARI=TRIM(NMFILE_CANARI)//'.DAT'

            OPEN(UNIT=25,FILE=NMFILE_CANARI,FORM='FORMATTED',STATUS='OLD')
            DO I=1,NDIM_FULL
               READ(25,*) ((YFTEMP(I,1,K+NOBS-NOBSTYPE,L),K=1,NOBSTYPE),L=1,NVAR)
            ENDDO
            CLOSE(25)

            DO I=1,NSIZE_NATURE
               IND=NR_NATURE(I)
               YO(I,:,:)=YOWR(IND,:,:)
               YF(I,:,:,:)=YFTEMP(IND,:,:,:)
            ENDDO

            DO L=1,NVAR
```

```
           DO I=1,NSIZE_NATURE
             DO J=1,PATCH_NUMBER
                IF((YO(I,1,L).NE.999.0).AND.(YO(I,2,L).NE.999.0).AND.(XI(I,J,1,L
).NE.999.0)) THEN
                  DO K=1,NOBSTYPE
                    HO(I,J,K+NOBS-NOBSTYPE,L)=(YF(I,2,K+NOBS-NOBSTYPE,L)-
YF(I,1,K+NOBS-NOBSTYPE,L))/ZEPS(I,J,L)
                    ZB(I,K+NOBS-NOBSTYPE,L) = YO(I,K+NOBS-NOBSTYPE,L)-
YF(I,1,K+NOBS-NOBSTYPE,L)
                  ENDDO
                ENDIF
             ENDDO
             DO K=1,NOBSTYPE
             R(I,NOBS-(NOBSTYPE-K),NOBS-(NOBSTYPE-K),L)=XERROBS(K)*XERROBS(K)
             ENDDO
           ENDDO
           ENDDO
           CLOSE(ILOBS)
10  CONTINUE ! if T2m HU2m observations does not exist
ENDDO
DEALLOCATE(YOWR)
!print *, 'NOBS = ',NOBS
!print *, 'YO ... '
!print *, (YO(I,1,1),I=1,9)
!print *, 'YF référence ... '
!do I=1,9
!print *, (YF(I,1,J,1),J=1,NOBS)
!enddo
print *, 'HO ... '
do I=1,9
print *, (HO(I,1,J,1),J=1,NOBS)
enddo
print *, 'ZB ... '
do I=1,9
print *, (ZB(I,J,1),J=1,NOBS)
enddo
!print *, 'R ... '
!do I=1,9
!print *, 'point ',I
!do J=1,NOBS
!print *, (R(I,J,K,1),K=1,NOBS)
!enddo
!enddo

!
! ANALYSIS STEP
!
! Background error matrix
!
DO L=1,NVAR
DO I=1,NSIZE_NATURE
  DO J=1,PATCH_NUMBER
     IF((YO(I,1,L).NE.999.0).AND.(YO(I,2,L).NE.999.0).AND.(XI(I,J,1,L).NE.999.0)
) THEN
       B(I,J,L,L)=XSIGMAB*XSIGMAB*COFSWI(I,J,L)*COFSWI(I,J,L)
     ENDIF
  ENDDO
ENDDO
ENDDO
!
! WG2init analysis
!
! WG2a=WG2f + K*(WG2f-WG2obs) (with WG2 = 'WG2init')
!       K=PHT(HPHT+R)^-1
```

```
!
ALLOCATE(MATTEMP3(NOBS,NVAR))
!
DO L=1,NVAR
 DO I=1,NSIZE_NATURE
   DO J=1,PATCH_NUMBER
     IF((YO(I,1,L).NE.999.0).AND.(YO(I,2,L).NE.999.0).AND.(XI(I,J,1,L).NE.999.0)
) THEN
        !
        ! HPHT+R --> K1
        !
        HOT(I,J,:,:) = TRANSPOSE(HO(I,J,:,:))
        K1(:,:,L) = MATMUL(HO(I,J,:,:),MATMUL(B(I,J,:,:),HOT(I,J,:,:))) +
R(I,:,:,L)
        HBHT(I,J,:,:,L) = MATMUL(HO(I,J,:,:),MATMUL(B(I,J,:,:),HOT(I,J,:,:)))
        CALL CHOLDC(NOBS,K1(:,:,L),ZP(I,:,L))                    ! Cholesky
decomposition (1)
!        ZB(I,:,L) = YO(I,:,L) - YF(I,1,:,L)
        CALL CHOLSL(NOBS,K1(:,:,L),ZP(I,:,L),ZB(I,:,L),ZX(I,:,L)) ! Cholesky
decomposition (2)
        XINCR(I,J,:) = MATMUL(B(I,J,:,:),MATMUL(HOT(I,J,:,:),ZX(I,:,L)))
        XI(I,J,1,L) = XI(I,J,1,L) + XINCR(I,J,L)
        !
! (HPHT+R)^-1) --> ZX
        !
        ! PHT*(HPHT+R)^-1 --> MATTEMP3
        !
        ! PHT*(HPHT+R)^-1*(WG2obs-WG2f) --> ZY
        !
     ENDIF
   ENDDO
ENDDO
ENDDO

print *, 'XINCR ...'
do I=1,9
print *, (XINCR(I,1,J),J=1,1)
enddo
print *, 'HBHT ...'
do I=1,9
print *, 'POINT : ',I
do K=1,NOBS
print *, (HBHT(I,1,J,K,1),J=1,NOBS)
enddo
enddo

!print *, 'R Matrix ...'
!do I=1,9
!print *, 'POINT : ',I
!do K=1,NOBS
!print *, (R(I,J,K,1),J=1,NOBS)
!enddo
!enddo



CALL INIT_IO_SURF_n(YPROGRAM,'NATURE','ISBA  ','WRITE')
CALL WRITE_SURF(YPROGRAM,YVAR,XI(:,1,1,1),IRESP,HCOMMENT=PREFIX)
CALL END_IO_SURF_n(YPROGRAM)

140 FORMAT(1000000(1f20.8))

DEALLOCATE(VECT)
DEALLOCATE(YF)
```

```
DEALLOCATE(YFTEMP)
DEALLOCATE(XI)
DEALLOCATE(ZB)
DEALLOCATE(HO)
DEALLOCATE(R)
DEALLOCATE(B)
DEALLOCATE(ZP)
DEALLOCATE(ZEPS)
DEALLOCATE(YO)
DEALLOCATE(HOT)
DEALLOCATE(K1)
DEALLOCATE(HBHT)
DEALLOCATE(ZX)
DEALLOCATE(XINCR)
DEALLOCATE(MATTEMP3)
!
END PROGRAM VARASSIM
!
!-------------------------------------------------------
SUBROUTINE TRANS_CHAINE(CHAINE,ENTIER,OPTION)
!-------------------------------------------------------
!
!
! transforme l'entier 23416 en la chaine de caracteres '23416'
! option est utilise si superieur a 0. Dans ce cas si l'entier
! a un nombre de digits inferieur a option, la difference est
! remplie avec des '0' dans chaine.
!
!
! ex: entier=256 option=0  ===>   chaine='256'
!     entier=256 option=2  ===>   chaine='256'
!     entier=256 option=7  ===>   chaine='0000256'

      DOUBLE PRECISION REEL1
      CHARACTER*50 CHAINE
      INTEGER OPTION
      CHARACTER              :: C
      INTEGER                :: RESTE , NBDIGIT , NUM , ENTIER
      DOUBLE PRECISION       :: DIVI

      REEL1  = ENTIER/1.
      CHAINE = ''
      RESTE  = ENTIER
      NBDIGIT= INT(LOG(REEL1)/LOG(10.)+0.001)+1
      DIVI   = 1

      DIG_LOOP: DO I = 1 , NBDIGIT - 1
      !-------------------------------

          DIVI = DIVI * 10

      END DO DIG_LOOP
      !---------------

      IF (OPTION.GT.0) THEN
         IF (OPTION.GT.NBDIGIT) THEN

            OPT_LOOP: DO I = 1 , OPTION - NBDIGIT
            !-----------------------------------

               CHAINE(I:I)='0'

            END DO OPT_LOOP
            !---------------
```

```
         ENDIF
      ENDIF

      DIGI_LOOP: DO I = 1 , NBDIGIT
      !----------------------------

         NUM=INT(RESTE/DIVI)
         C=CHAR(NUM+48)
         IF (OPTION.GT.0) J=OPTION-NBDIGIT+I
         IF (OPTION.EQ.0) J=I
         CHAINE(J:J)=C
         RESTE=NINT(RESTE-NUM*DIVI)
         DIVI=DIVI/10

      END DO DIGI_LOOP
      !---------------

      RETURN
END SUBROUTINE TRANS_CHAINE
```