Présentation de CESAM 2000

Avantages

Toutes les options (physiques, numériques) sont comprises dans un seul et même exécutable DONC plus besoin de recompiler quoi que ce soit lors d'un simple changement d'options.

Entre autres, le programme principal est réduit à sa plus simple expression :

```
Call CESAM2000
End
```

Réécrit en Fortran 90/95 standard DONC portable sur tout système informatique sans aucun problème (avec seulement un tout petit peu de travail si la bibliothèque graphique PGPLOT n'est pas disponible).

Utilisation : de la notion de `MODULE`,
des droits d'accessibilité `PRIVATE` et `PUBLIC`,
des vocations d'arguments `Intent(In)` et `Intent(InOut)`,
comme gages de sécurité.

Inconvénients (en fait, travail à faire)

Quelques améliorations à faire encore, comme revoir les lignes de programmation concernant les allocations, dé-allocations et re-allocations dynamiques, ….

Fabrication d'un 'makefile' pour gérer la dépendance et la hiérarchisation des modules (l'ordre de compilation des fichiers est désormais essentielle).

Description des fichiers 'source'

Après une première restructuration du code sous forme de 'gros' modules regroupant chacun un aspect complet d'une option physique (e.g. mod_nuc, mod_opacite, mod_perte, mod_etat, …), il est apparu que si ce choix était raisonnable pour la distribution du code (par exemple, présence de seulement 34 fichiers à compiler et de 11 niveaux hiérarchiques), ce n'était pas le meilleur en ce qui concerne les modifications, essentiellement lors des phases de test.

Dans cette présente structure, la quasi-totalité des routines de type 'option physique' sont implémentées à raison d'un module (les modules du type zzzmod_<*routine*>) par routine (routine dite 'spécifique'). Chaque aspect complet d'une option physique est implémenté dans un module mod_<*module*> à travers une routine de nom <*module*> (routine dite 'générique'). Ceci donne donc 90 fichiers à compiler et 17 niveaux hiérarchiques.

Les fichiers ayant pour nom '<*routine*>.f' implémentent le module 'zzmod_<*routine*>' contenant la routine '<*routine*>' comme entité publique (et en règle générale la seule !).

Les fichiers ayant pour nom 'mod_<*module*>.f' ou 'mod_<*module*>.f90' implémentent le module '<*module*>' contenant les routines ou les données tel que décrit dans le document 'Liste_modules_contenu.pdf'

Quelques illustrations et exemples valent mieux qu'un long discours !

Un contenu enrichi pour le fichier « *.don* »

La principale modification : &NL_NOMS

```
&NL_NOMS
 nom_ctes_phys = 'ctes_phys94',
 nom_des = 'des_m',
 nom_perte = 'perte_ext',
 nom_atm = 'lim_atm',
 nom_tdetau = 'k5777',
 nom_abon = 'solaire_gn',
 nom_diffm = 'diffm_mp',
 nom_difft = 'difft_nu',
 nom_conv = 'conv_jmj',
 nom_etat = 'etat_ceff',
 nom_opa = 'opa_yveline',
 nom_nuc = 'ppcno12',
 nom_nuc_cpl = 'NACRE',
 nom_osc = 'osc_adia',
 nom_chemin = 'K:\CESAM2000\SUN_STAR_DATA\'
/
```

Autres modifications : *cf.* aussi P.Morel

```
&NL_ESPACE
 MTOT =  1.0,
 MDOT =  0.0E+00,
 DER_NUM = f,
 PRECISION = 'np',
 N_MAX=1500                                    ← NEW
/

&NL_ATMOSPHERE                                 ← NEW
 TAU_MAX = 10.0,
 LIM_RO  = T
/

&NL_ROT
 W_ROT = 0.d0,
 DW = 0.d-7,                                   ← NEW
 ROT_SOLID = T
/

&NL_NUC
 MITLER = F
/
```

Quelques modules essentiels

Le module '`mod_kind`' :

```
Module mod_kind
     Integer, Parameter, Public ::  DP = KIND(1.d0) , &
                                    SP = KIND(1.)
End Module mod_kind
```

Le module '`mod_numerique`' :

Ce module regroupe toute la partie numérique de CESAM. C'est une composante essentielle de la qualité et de la robustesse de CESAM comparativement à d'autres codes !

Le module '`mod_donnees`' :

Ce module regroupe toutes les données initiales au calcul e.g. constantes physiques fondamentales, données du modèle en cours de calcul (masse, conditions d'arrêt, …)

Le module '`mod_variables`' :

Ce module regroupe toutes les variables du problème.
Il peut être envisagé (vote, référendum, …) de couper ce modules en deux modules, le premier regroupant les variables scalaires et tableaux dont le dimensionnement ne varie en aucune façon au cours du calcul, l'autre les tableaux dont le dimensionnement varie en cours du calcul (e.g. tout ce qui est lié aux nombres de couches, la gestion des zones convectives, …)

Le module '`mod_ini_ctes_phys`' :

```
Module mod_ini_ctes_phys
   PRIVATE
   PUBLIC :: ini_ctes_phys
CONTAINS
   Subroutine ini_ctes_phys
      USE mod_donnees, Only: nom_ctes_phys
      Implicit None
      Select Case ( nom_ctes_phys )
         Case ( "ctes_phys85" ) ; Call ctes_phys85
         Case ( "ctes_phys94" ) ; Call ctes_phys94
         Case Default
            Print*, "routine de constantes physiques inconnue: ", nom_ctes_phys
            Print*, "routines connues: ctes_phys94, ctes_phys85"
            STOP " Arret, PB. dans ini_ctes_phys "
       End Select
   End Subroutine ini_ctes_phys
End Module mod_ini_ctes_phys
```

← commentaires !!

Un exemple de routine « spécifique »

```
Module zzmod_iben
  USE mod_variables, ONLY: c_iben
  PRIVATE
  PUBLIC :: iben
CONTAINS
  Subroutine iben(t,dcomp,jac,deriv,fait,epsilon,et,ero,ex)
!
! fausses reactions thermonucleaires pour modele initial de
!     pre main sequence: epsilon = c T    d'apres Iben
! entree : (voir aussi plus loin)
!   fait = 1 : initialisation de la composition chimique
!        = 2 : calcul de dcomp et jacobien si deriv
!        = 3 : energie nucleaire et derivees / t et ro
!        = 4 : production de neutrinos
! sorties :
!   dcomp : derivee temporelle (unite de temps : 10**6 ans)
!   jac : jacobien (unite de temps : 10**6 ans)
!   epsilon, et, ero, ex : energie thermonucleaire (unite de
!         temps: s) et derivees par rapport a t, ro, X
!-------------------------------------------------------------
!
    USE mod_kind
    Implicit None
    !
    Real(DP), Intent(In) :: t      ! temperature
    Integer, Intent(In) :: fait    ! type de travail
    Logical, Intent(In) :: deriv   ! calcul evt des derivees
    !
    Real(DP), Intent(Out), Dimension(:,:) :: jac
    Real(DP), Intent(Out), Dimension(:) :: dcomp, epsilon, ex
    Real(DP), Intent(Out) :: et, ero
    !
    Select Case (fait)
      Case (1)
         Write(*,1)
         Write(2,1)
 1       Format(/,"methode de Iben pour la recherche du modele initial PMS",/)
      Case (2)
         dcomp = 0._DP
         If (deriv) jac = 0._DP
      Case (3)
         Epsilon = 0._DP ; epsilon(1) = c_iben * t
         If (deriv) Then ; ero = 0.d0 ; et = c_iben ; ex = 0.d0 ; End If
      Case (4)
         Continue
    End Select
    !
  End Subroutine iben
  !
End Module zzmod_iben
```

Un exemple de routine « générique »

```
Module mod_nuc
  USE mod_donnees, ONLY : nom_nuc
  USE zzmod_iben
  USE zzmod_pp1
  USE zzmod_ppcno9
  USE zzmod_ppcno10
  [...]
  !
  PRIVATE
  PUBLIC :: nuc
  !
CONTAINS
  !
  Subroutine nuc (t, ro, comp, dcomp, jac, deriv, fait, epsilon,    &
                  et, ero, ex, hhe, be7e, b8e, n13e, o15e, f17e )
    USE mod_kind
    Implicit None
    !
    Real(DP), Intent(In):: t, ro
    Integer, Intent(In) :: fait
    Logical, Intent(In) :: deriv
    Real(DP), Intent(InOut), Dimension(:) :: comp
    Real(DP), Intent(Out), Dimension(:,:) :: jac
    Real(DP), Intent(Out), Dimension(:) :: dcomp, ex, epsilon
    Real(DP), Intent(Out) :: et, ero, hhe, be7e, b8e, n13e, o15e, f17e
    !
    Select Case ( nom_nuc )
      Case ( "iben" )
        Call iben (t,ro,comp,dcomp,jac,deriv,fait,epsilon,et,ero,ex)
      Case ( "pp1" )
        Call pp1 (t,ro,comp,dcomp,jac,deriv,fait,epsilon,et,ero,ex)
      Case ( "ppcno9" )
        Call ppcno9 (t,ro,comp,dcomp,jac,deriv,fait,epsilon,      &
                     et,ero,ex,hhe,be7e,b8e,n13e,o15e,f17e)
      Case( "ppcno10" )
        Call ppcno10 (t,ro,comp,dcomp,jac,deriv,fait,epsilon,     &
                      et,ero,ex,hhe,be7e,b8e,n13e,o15e,f17e)
      [...]
      Case Default
         Print*, "routine de reaction nucleaire inconnue: ', nom_nuc
         Print*, "routines connues: pp1, ppcno9, ppcno10, "
         [...]
         STOP " Arret dans nuc "
    End Select
    !
  End Subroutine nuc
  !
End Module mod_nuc
```

Documentation

Actuellement disponibles sur CESAM2000 (CESAM2k)

Notice de CESAM5

Pour tout ce qui concerne les mathématiques utilisées, les options physiques disponibles ;

« Liste_modules_contenu.pdf »

est le fichier descriptif des fichiers source de CESAM2000 par ordre hiérarchique de compilation des modules.

Exemple :

```
| Niveau 09 |
mod_evol.f
     (INCLUDE) diffus.f         (PRIVATE)
     (INCLUDE) eq_diffus.f      (PRIVATE)
eq_atm.f
lim_zc.f

| Niveau 10 |
col_atm.f
perte_tot.f

| Niveau 11 |
mod_perte.f
lim_atm.f
```

« Liste_routines_dans_modules.pdf »

est le fichier décrivant, par ordre alphabétique, l'ensemble des routines présentes dans CESAM2000 avec indication du module les définissant (donc le nom du fichier) et les différentes routines les utilisant.

Exemple :

```
abon_ini     zzmod_abon_ini       ppcno10, ppcno10Fe, ppcno10K.f, ppcno11,
                                  ppcno12, ppcno12Be, ppcno12Li, ppcno3a9,
                                  ppcno3ac10, ppcno9

acc_rad      zzmod_acc_rad        diffm_br

arb_rom      mod_util             cesam

atm          mod_atm              cesam, static_m, static_r

bsp1ddn      mod_numerique        cesam, hopf, k5750, k5777
```