DE LA RECHERCHE À L'INDUSTRIE

DÉVELOPPER DANS CAST3M

DISPONIBLE SUR : **HTTP://WWW-CAST3M.CEA.FR/INDEX.PHP?XML=FORMATIONS**

Clément BERTHINIER

www.cea.fr

DERNIÈRE MODIFICATION : 09 MARS 2020

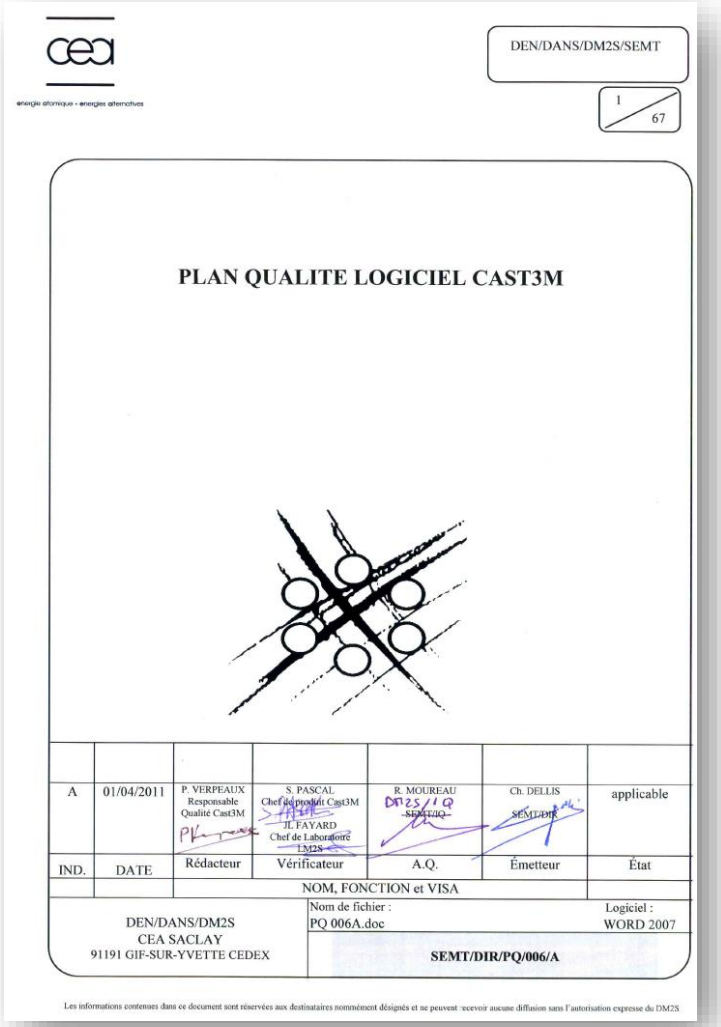
SOMMAIRE

CAST3M : ÉLÉMENTS DE QUALITÉ

CAST3M : ÉLÉMENTS DE QUALITÉ (V&V)

- **Cast3M est développé sous Assurance Qualité**

  Référence CEA 2011 : **SEMT/DIR/PQ/006/A**



DEN/DANS/DM2S/SEMT

1 / 67

PLAN QUALITE LOGICIEL CAST3M

IND.	DATE	Rédacteur	Vérificateur	A.Q.	Émetteur	État
A	01/04/2011	P. VERPEAUX Responsable Qualité Cast3M	S. PASCAL Chef de produit Cast3M / JL FAYARD Chef de Laboratoire LM2S	R. MOUREAU DM2S/LQ SEMT/IQ	Ch. DELLIS SEMT/DIR	applicable
		NOM, FONCTION et VISA				

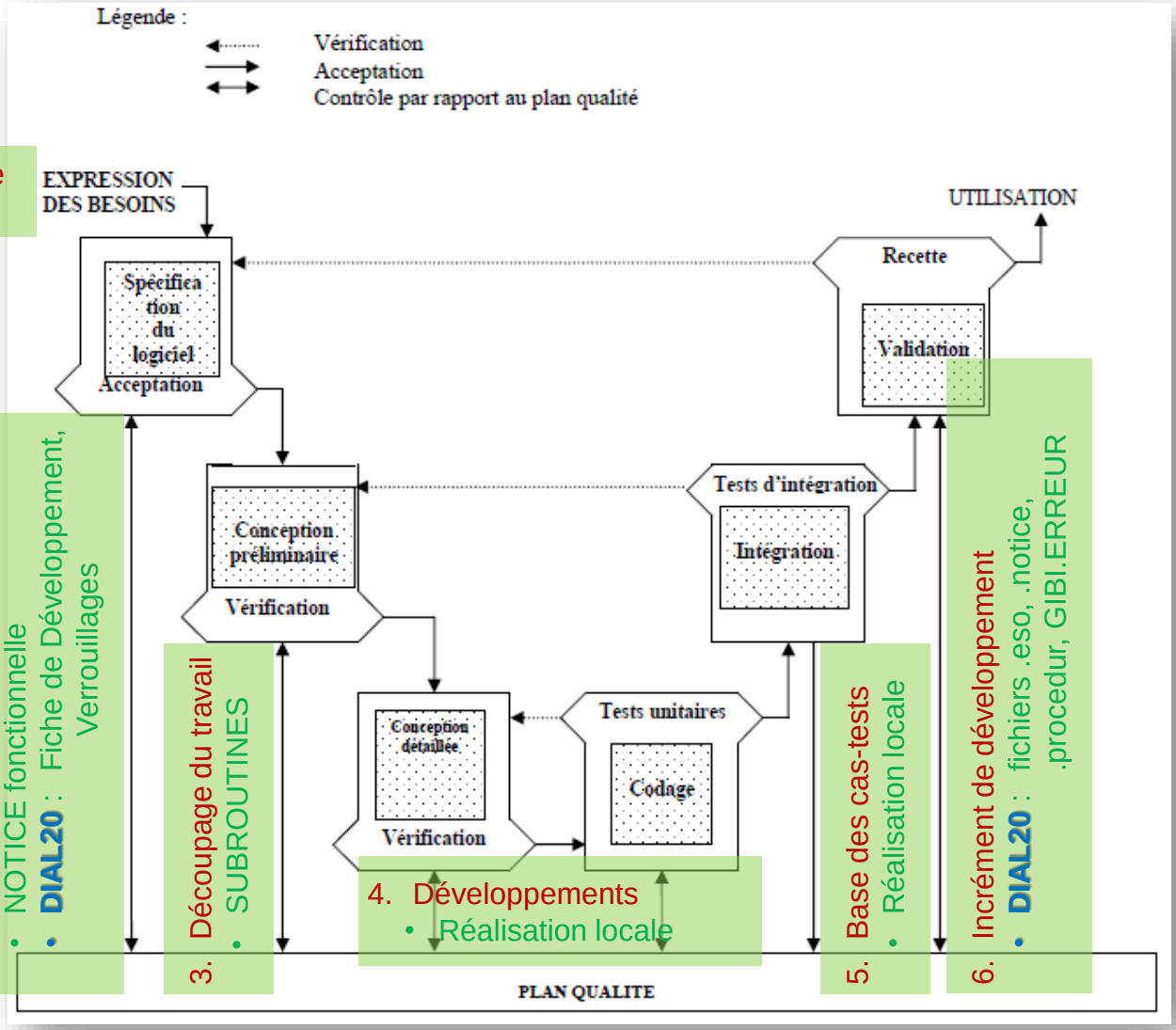
DEN/DANS/DM2S
CEA SACLAY
91191 GIF-SUR-YVETTE CEDEX

Nom de fichier : PQ 006A.doc — Logiciel : WORD 2007

SEMT/DIR/PQ/006/A

CAST3M : ÉLÉMENTS DE QUALITÉ (V&V)

- **Extrait du PQL** (pp.19/67) : Développement incrémental



CAST3M : ÉLÉMENTS DE QUALITÉ (V&V)

- **1 notice fonctionnelle par OPERATEUR / PROCEDURE**
  Le cas-test « `notice.dgibi` » vérifie automatiquement cela !
- **1 seule SUBROUTINE par fichier source (`.eso`)**
- **1 seule version des sources pour la version du jour**
- **1 version annuelle pour l'industrialisation hors DM2S**
- **1 nouveau développement ➔ cas-tests (`.dgibi`)**

  Non conseillé : Absence de cas-tests

  Minimaliste : **V**érification
  - ➔ Passage dans un maximum de lignes de code sans « planter »

  Très correct : **V**érification & Comparaison à une Réf. (code, publi, …)
  - ➔ Erreur si les valeurs diffèrent d'une référence (à une précision près)

  Idéalement : **V**érification & **V**alidation
  - ➔ Comparaison à une solution analytique (mettre Réf. Publi)
  - ➔ Erreur si les valeurs diffèrent de la solution (à une précision près)

  Utilisateurs : **V**érification & **V**alidation & **Q**ualification
  - ➔ Il incombe aux utilisateurs (industriel, académique, etc.) de qualifier Cast3M dans leur domaine d'application !!!
  - ➔ S'appuyer sur les cas-tests du logiciel n'est pas suffisant
  - ➔ Envoyer nous vos cas-tests de **VVQ**

PRÉREQUIS EN FORTRAN 77 POUR CAST3M

PRÉREQUIS EN FORTRAN 77 POUR CAST3M

Coder des instructions

Instructions entre la colonne 7 et 72

Aide : mettre la limitation dans l'éditeur de texte

```
12345678123456781234567812345678123456781234567812345678123456781234567812345678
```

Coder une instruction sur plusieurs lignes

Sur chaque ligne, instructions entre la colonne 7 et 72

Caractère ASCII de continuation en colonne 6 (ex : « **&** »)

```
12345678123456781234567812345678123456781234567812345678123456781234567812345678
      IF ((X(JLAST ).NE.ABS(X(ISAVE(2)))) .AND.
     &     (ISAVE(3) .LT.ITMAX)) THEN
```

Caractère ASCII de continuation d'une instruction

Commenter son code source

Caractère « **C** » ou « ***** » en colonne 1

```
12345678123456781234567812345678123456781234567812345678123456781234567812345678
C     Les deux objets doivent être de même taille
      IF (LDIM1 .NE. LDIM2) THEN
        instructions
      ENDIF
```

PRÉREQUIS EN FORTRAN 77 POUR CAST3M

Typage implicite obligatoire des entiers et flottants

Nécessité pour le traducteur ESOPE qui ne type pas ses variables

```
      IMPLICIT INTEGER(I-N)
      IMPLICIT REAL*8(A-H,O-Z)
```

Déclaration & Typage de paramètres & variables

Surcharge le typage implicite

Un `PARAMETER` n'est pas ré-affectable dans la subroutine

```
      LOGICAL  B1,B2
      REAL*8   XX,XX1,ZERO
      INTEGER  I,LD,IT
      PARAMETER (IT=5,ZERO=0.D0)
C     Chaines de 4 caractères
      CHARACTER*(4)  CHA4a
```

Déclaration & Typage de tableaux variables

Surcharge le typage implicite

Tailles fixes pour des tableaux de travail

Tailles libres pour des tableaux donnés en argument

```
      LOGICAL  BTAB1(5), BTAB2(*)
      REAL*8   XTAB1(7), XTAB2(*)
      INTEGER  ITAB1(2), ITAB2(*)
C     TABLEAU de 26 chaines de 4 caractères
      CHARACTER*(4)  CHA4a(26)
```

PRÉREQUIS EN FORTRAN 77 POUR CAST3M

Déclarer & Invoquer une SUBROUTINE

Attention : Les arguments sont à la fois des entrées et des sorties (effet de bord)

Conseil : ajouter un code de retour IRET (savoir comment s'est déroulée l'exécution)

```
      SUBROUTINE MASUB1(ARG1,ARG2,ARG3,IRET)
      ...déclarations...
      ...instructions...
C     Sortir avant la fin (balise END)
      RETURN
      ...instructions...
      END
C     Invoquer
      CALL MASUB1(ARG1,ARG2,ARG3,IRET)
```

Déclarer & Invoquer une FONCTION

Attention : Les arguments sont à la fois des entrées et des sorties (effet de bord)

```
      FUNCTION MAFUN1(ARG1,ARG2,ARG3)
      ...déclarations...
      ...instructions...
C     Sortir avant la fin (balise END)
      MAFUN1 = VAL1
      RETURN
      ...instructions...
      MAFUN1 = VAL2
      END
C     Invoquer
      RES1 = MAFUN1(ARG1,ARG2,ARG3)
```

PRÉREQUIS EN FORTRAN 77 POUR CAST3M

Opérateurs de comparaison : .LT., .LE., .EQ., .NE., .GT., .GE.

Expression logique (portable pour les types INTEGER, LOGICAL et CHARACTER)

```
C     Vrai si X est strictement inférieur à Y
      X .LT. Y

C     Vrai si X est inférieur ou égal à Y
      X .LE. Y

C     Vrai si X est égal à Y : Attention avec les FLOTTANTS (voir après)
      X .EQ. Y

C     Vrai si X est différent de Y
      X .NE. Y

C     Vrai si X est strictement supérieur à Y
      X .GT. Y

C     Vrai si X est supérieur ou égal à Y
      X .GE. Y
```

Assigner le résultat d'une comparaison

```
C     Assignation du résultat
      LOGIQ1 = ...expression_logique...
```

Egalité entre 2 FLOTTANTS

Résultat incertain lors de la comparaisons entre 2 type `REAL*8` !!!

Les précisions sont contenues dans un include : `CCREEL.INC`

- **`XPETIT`** : plus petit `FLOTTANT` positif 10^{-304} dont l'inverse n'est pas `inf`
- **`XGRAND`** : plus grand `FLOTTANT` positif 10^{+304} dont l'inverse n'est pas `0.D0`
- **`XZPREC`** : plus petit `FLOTTANT` positif $8{,}8.10^{-16}$ qui peut être soustrait à `1.D0` sans que le ne soit `1.D0`

Remarque : Ces grandeurs sont accessibles en GIBIANE (respectivement)

```
'VALE' 'PETI';
'VALE' 'GRAN';
'VALE' 'PREC';
```

```
C     Tester « égalité » avec des REAL*8
      ABS(X-Y) .LT. MAX(XZPREC * MAX(ABS(X), ABS(Y)), XPETIT)
```

PRÉREQUIS EN FORTRAN 77 POUR CAST3M

Opérateurs logiques : .NOT., .AND., .OR., .XOR., .EQV., .NEQV.

Expression logique

```
C     Vrai si Y est faux (Négation logique)
      .NOT. Y

C     Vrai si toutes les opérandes sont vraies
      X .AND. Y .AND. Z

C     Vrai si au moins l'une des opérandes est vraie
      X .OR. Y .OR. Z

C     Vrai si une seule des opérandes est vraie
      X .XOR. Y

C     Vrai si X et Y sont dans le même état logique
      X .EQV. Y

C     Vrai si X et Y ne sont pas dans le même état logique
      X .NEQV. Y
```

Assigner le résultat d'une expression logique

```
C     Assignation du résultat
      LOGIQ1 = ...expression_logique...
```

PRÉREQUIS EN FORTRAN 77 POUR CAST3M

Instructions conditionnelles : IF, THEN, ELSE, ELSEIF, ENDIF

Imbrication possible

Indentation du code conseillée pour une relecture facile

```
C     1 seule instruction conditionnelle
      IF (expression_logique) instruction

C     Plusieurs instructions conditionnelles
      IF (expression_logique) THEN
        instruction_1
        instruction_2
        instruction_3
      ENDIF

C     Plusieurs instructions conditionnelles exclusives
      IF    (expression_logique) THEN
        instruction_1
        instruction_2
        instruction_3
      ELSEIF(expression_logique) THEN
        instruction_4
        instruction_5

C     Le ELSE (sans expression logique) est optionnel
      ELSE
        instruction_6
        instruction_7
      ENDIF
```

PRÉREQUIS EN FORTRAN 77 POUR CAST3M

Les sauts : GOTO

Un *label* est un entier compris entre 0 et 99999
Un *label* doit rentrer dans les 5 premières colonnes de la ligne
Un *label* doit être unique
Un *label* peut être pointé par plusieurs GOTO
Un *label* peut être situé en dessus ou en dessous des GOTO associés

```
12345
      GOTO 99999
      ...instructions...

C     Tous les GOTO 99999 renvoient ici
99999 CONTINUE
      ...instructions...

      GOTO 99999

C     Autre utilisation possible de GOTO
      GOTO (13,8,300),IVAL
C     Il faut lire :
C     Si IVAL=1 ➔ GOTO 13
C     Si IVAL=2 ➔ GOTO 8
C     Si IVAL=3 ➔ GOTO 300
C     Pour toute autre valeur de IVAL l'instruction est ignorée
```

PRÉREQUIS EN FORTRAN 77 POUR CAST3M

Boucles : DO, ENDDO, CONTINUE

Imbrication possible & indentation conseillée

```
C     Boucle incrémentale simple sans label
      DO I=IDEB,IFIN
        ...instructions...
      ENDDO

C     Boucle incrémentale simple de label 10
      DO 10 I=IDEB,IFIN
C       Si IFIN < IDEB : Boucle 10 sautée
        ...instructions...

C       Itérer la boucle 10
        IF (logique_1) GOTO 10
        ...instructions...

C       Quitter une boucle
        IF (logique_2) GOTO 11
        ...instructions...
 10   CONTINUE
      ...instructions...
 11   CONTINUE

C     Boucle avec un incrément donné
      DO I=11,7,-2
C       L'indice de boucle I vaudra respectivement 11, 9 et 7
        ...instructions...
      ENDDO
```

EXTENSION ESOPE DE FORTRAN 77

- SEGMENT / ENDSEGMENT
  - SEGXXX
- MACRO
- CASE / ENDCASE

EXTENSION ESOPE DE FORTRAN 77

Présentation de ESOPE

Extension du langage FORTRAN 77

Apporte la notion de **structures de données dynamiques**

Une structure de données au sens de Cast3M se caractérise par :

- Son ***TYPE*** : ESOPE introduit un nouveau type : **SEGMENT**
- Son ***NOM*** : Mot utilisé pour nommer la structure de données
- Sa ***VALEUR*** : Pointeur (type **INTEGER**) vers la structure de données
- Son ***CONTENU*** : Données contenues dans la structure
- Ses ***ALIAS*** : Autres noms pouvant pointer sur la même structure

6 instructions à apprendre pour la manipulation d'un **SEGMENT**

- *Déclaration* : **SEGMENT** & **ENDSEGMENT**
- *Initialisation* : **SEGINI,...**
- *Suppression* : **SEGSUP,...**
- *Activation* : **SEGACT,...**
- *Désactivation* : **SEGDES,...**
- *Ajustement dynamique* : **SEGADJ,...**

1 syntaxe pour accéder au contenu d'un **SEGMENT**

3 types de **MACRO** : Expression récurrente, Fonctionnelle, Enumération

1 instruction **CASE** : Clarifie la manipulation des expressions logiques

EXTENSION ESOPE DE FORTRAN 77

Déclaration d'un SEGMENT

Positionnement dans les SUBROUTINES en tête avec les déclarations FORTRAN

Peut contenir tous les types FORTRAN compatibles avec Cast3M

3 types de structures de SEGMENTS

- simple (Règle des **IMPLICIT** pour le type FORTRAN sous-jacent)
- extensibles (Règle des **IMPLICIT** pour le type FORTRAN sous-jacent) ➔ Pour la performance
- complexes

```
C     Structures simples
      SEGMENT ISEG1(N1)
      SEGMENT XSEG1(N2)
C        ISEG1 & XSEG1(POINTEUR) | ISEG1(i)(ENTIER) | XSEG1(i)(FLOTTANT)

C     Structure simple extensible (Explication plus loin)
      SEGMENT ISEG2(0)
      SEGMENT XSEG2(0)
        ISEG2 & XSEG2(POINTEUR) | ISEG2(i)(ENTIER) | XSEG2(i)(FLOTTANT)

C     Structure complexe
      SEGMENT MASTRU
        INTEGER          ITAB(N1),IVAL1
        REAL*8           XTAB(N1,N2),XVAL1
        CHARACTER*(N0)   CTAB(N2),CVAL1
        LOGICAL          BTAB(N2,N1),BVAL1
      ENDSEGMENT

C     ALIAS vers la structure MASTRU
      POINTEUR MASTR1.MASTRU,MASTR2.MASTRU,ALIAS1.MASTRU
```

Entiers dimensionnels

Eléments de SEGMENTS

EXTENSION ESOPE DE FORTRAN 77

Initialisation d'un SEGMENT

Attention : Bien initialiser les ENTIERS dimensionnels avant !!!

```
C     Initialisation d'un ou plusieurs SEGMENTS
      N1=2
      N2=1000
      SEGINI,MASTRU,MASTR2,...
C     MASTRU et MASTR2 : Leur valeur (ENTIER) correspond a des POINTEURS
C                        valides
C     Les elements de SEGMENT sont initialisés selon leur type
C     INTEGER ==> 0       | REAL*8         ==> 0.D0
C     LOGICAL ==>.FALSE.  | CHARACTER(NN)==> ' ' (NN caractères)
```

ENTIERS dimmensionnants invalides : GEMAT ERROR

```
 --- LONGUEUR DU SEGMENT INVALIDE
0--- GEMAT ERROR          --- SUBROUTINE  : PILOT
 ---                          INSTRUCTION : 144
 ---                          SEGINI , MASTRU
```

Description de l'erreur

Repérage facile dans la SUBROUTINE pilot.f

Instruction impliquée avec le nom du SEGMENT

ENTIERS dimmensionnant trop grands : GEMAT ERROR

```
 --- PAS ASSEZ DE PLACE EN MEMOIRE
0--- GEMAT ERROR          --- SUBROUTINE  : PILOT
 ---                          INSTRUCTION : 144
 ---                          SEGINI , MASTRU
```

Description de l'erreur

Repérage facile dans la SUBROUTINE pilot.f

Instruction impliquée avec le nom du SEGMENT

Initialisation d'un SEGMENT par copie d'un autre

Intérêt : Reprendre un certain nombre de données sans avoir à faire la copie

```
C     Initialisation d'un SEGMENT par copie d'un autre
      SEGINI,MASTR2=MASTRU
C     MASTR2 : copie de MASTRU avec un pointeur différent

C     Exemple POINTEUR invalide : MASTRU=-4 ; SEGINI,MASTR2=MASTRU
```

POINTEUR à copier invalide : GEMAT ERROR (mauvaise affectation)

```
 --- POINTEUR ARGUMENT INVALIDE
0--- GEMAT ERROR          --- SUBROUTINE  : PILOT
 ---                          INSTRUCTION : 168
 ---                          SEGINI = MASTR2
```

Description de l'erreur

Repérage facile dans la SUBROUTINE pilot.f

Instruction impliquée avec le nom du SEGMENT

Alias vers une structure différente : Erreur de traduction (avant la compilation)

```
      Sommaire affiché et erreur dans le listing (pilot.lst)
0***  ERROR  : PILOT   *** : LSEGEG , P=Q : TYPES DIFFERENTS ---
```

EXTENSION ESOPE DE FORTRAN 77

Ajustement dynamique de la taille d'un élément de SEGMENT

Intérêt : Lorsqu'on ne connait pas la taille a priori

Attention : Si possible éviter l'ajustement 1 par 1 (contre-performant)

```
      SEGMENT MASTRU
        INTEGER ITAB(N1)
        REAL*8  XTAB(N1,N2)
      ENDSEGMENT
      ...instructions...

C     Initialisation de MASTRU
      N1=2
      N2=1000
      SEGINI,MASTRU
      ...instructions...

C     Extension de MASTRU pour la taille N1 (Préférer des gros paquets)
      N1=5
      SEGADJ,MASTRU
      ...instructions...

C     Ajustement de MASTRU pour les tailles N1 et N2 simultanément
      N1=1
      N2=100
      SEGADJ,MASTRU
```

Mêmes erreurs (GEMAT ERROR) possibles que **SEGINI**

EXTENSION ESOPE DE FORTRAN 77

Suppression d'un SEGMENT

Supprimer les SEGMENTS de travail avant de quitter les SUBROUTINES

Libère l'espace mémoire et accélère le travail du ménage automatique

```
C     Suppression d'un ou plusieurs SEGMENTS
      SEGSUP,MASTRU,MASTR2,...
```

Suppression effective

SEGMENTS qui ont l'horodatage courant

Horodatage : compteur incrémenté à chaque instruction GIBIANE
répétable entre deux exécution d'un même jeu de données (facilite le débogage)

Queue de suppression (voir `ooosuq.eso` dans ESOPE)
- une queue par ASSISTANT (en parallèle)
- 64 : taille de la queue (voir variable `SUPQ` dans `IOOCOM.INC` dans ESOPE)
- Suppression effective : queue pleine, ménage
- Effet de bord : Contenu accessible tant que la suppression effective n'a pas eu lieu

Ménage automatique

Supprime tous les SEGMENTS inaccessibles depuis l'ensemble des OBJETS nommés dans Cast3M

POINTEUR supprimé invalide : GEMAT ERROR (mauvaise affectation)

```
 --- POINTEUR ARGUMENT INVALIDE
0--- GEMAT ERROR         --- SUBROUTINE  : PILOT
 ---                         INSTRUCTION : 154
 ---                         SEGSUP , MASTRU
 ---                         POINTEUR    :  1
```

Description de l'erreur

Repérage facile dans la SUBROUTINE pilot.f

Instruction impliquée avec le nom du SEGMENT

Valeur affectée au POINTEUR

Activation d'un SEGMENT

Indique a GEMAT que le contenu du SEGMENT est désormais accessible

2 Modes d'accès

- **RO** Lecture seule : Ecriture impossible dans les éléments du SEGMENT
  Plusieurs ASSISTANTS peuvent être en RO sur un même SEGMENT
- **RW** Lecture & écriture : Lecture & écriture possible dans les éléments du SEGMENT
  Suite à leur création (**SEGINI**), les SEGMENTS sont en RW par défaut
  Un seul ASSISTANT à la fois peut être en **RW** sur un même SEGMENT

```
C     Activation d'un ou plusieurs SEGMENTS en lecture seule
      SEGACT,MASTRU,MASTR2,...

C     Activation d'un ou plusieurs SEGMENTS en lecture & ecriture
      SEGACT,MASTRU*MOD,MASTR2*MOD,...
```

POINTEUR activé invalide : GEMAT ERROR (mauvaise affectation)

```
 --- POINTEUR ARGUMENT INVALIDE
0--- GEMAT ERROR          --- SUBROUTINE  : PILOT
 ---                          INSTRUCTION : 154
 ---                          SEGACT , MASTRU
 ---                          POINTEUR    :  1
```

Description de l'erreur

Repérage facile dans la SUBROUTINE pilot.f

Instruction impliquée avec le nom du SEGMENT

Valeur affectée au POINTEUR

Désactivation d'un SEGMENT

Indique a GEMAT que le SEGMENT n'a plus besoin d'être accédé en RO ou RW

Intérêt : SEGMENTS déplaçables sur le SWAP en cas de manque de mémoire

```
C     Désactivation d'un ou plusieurs SEGMENTS
      SEGDES,MASTRU,MASTR2,...
```

Désactivation effective : queue de désactivation (voir ooodeq.eso dans ESOPE)

Mise en queue de désactivation : Pour des raisons de performance
- Une queue par ASSISTANT (en parallèle)
- 64 : taille de la queue (Voir variable DESQ dans IOOCOM.INC dans ESOPE)
- Désactivation effective : queue pleine, **SEGINI**, **SEGADJ**

Effet de bord : Contenu accessible tant que la désactivation effective n'a pas eu lieu

Ménage automatique

Tous les SEGMENTS sont désactivés en sortie du ménage (ou en queue de désactivation)

POINTEUR désactivé invalide : GEMAT ERROR (mauvaise affectation)

```
 --- POINTEUR ARGUMENT INVALIDE
0--- GEMAT ERROR          --- SUBROUTINE  : PILOT
 ---                          INSTRUCTION : 154
 ---                          SEGDES , MASTRU
 ---                          POINTEUR    :  1
```

Description de l'erreur

Repérage facile dans la SUBROUTINE pilot.f

Instruction impliquée avec le nom du SEGMENT

Valeur affectée au POINTEUR

EXTENSION ESOPE DE FORTRAN 77

Accéder aux tailles des éléments d'un SEGMENT

Exemple avec le SEGMENT MASTRU

```
      SEGMENT MASTRU
        INTEGER          ITAB(N1)
        REAL*8           XTAB(N1,N2)
        CHARACTER*(N0)   CTAB(N2)
        LOGICAL          BTAB(N2,N1)
      ENDSEGMENT
```

Récupérer les dimensions des éléments du SEGMENT

```
C     Dimension d'un élément de segment a 1 dimension
      IVAL1=MASTRU.ITAB(/1)   ➔ récupère N1

C     Dimension d'un élément de segment a plusieurs dimensions
      IVAL2=MASTRU.XTAB(/1)   ➔ récupère N1
      IVAL3=MASTRU.XTAB(/2)   ➔ récupère N2

      IVAL4=MASTRU.BTAB(/1)   ➔ récupère N2
      IVAL5=MASTRU.BTAB(/2)   ➔ récupère N1

C     Attention avec les Tableaux de chaines de caractère
      IVAL6=MASTRU.CTAB(/1)   ➔ récupère N0
      IVAL7=MASTRU.CTAB(/2)   ➔ récupère N2
```

Ecrire des valeurs dans un SEGMENT

Exemple de structure avec le SEGMENT SAMOI

```
      SEGMENT SAMOI
        INTEGER          ITEST
        INTEGER          ITAB(N1)
        REAL*8           XTAB(N1,N2)
      ENDSEGMENT
```

Ecriture des valeurs dans les éléments de SEGMENT

```
C     Initialisation
      N1=10
      N2=40
      SEGINI,SAMOI

C     Remplissage (on a les droit en RW après le SEGINI)
      SAMOI.ITEST=1
      DO ii=1, N1
        SAMOI.ITAB(ii) =ii**2
      ENDDO

      DO jj=1, N2
        DO ii=1, N1
          SAMOI.XTAB(ii,jj)=REAL((ii-1)*jj + ii) ** 3
        ENDDO
      ENDDO
```

EXTENSION ESOPE DE FORTRAN 77

Ecrire des valeurs dans un SEGMENT extensible

Exemple de structure avec le SEGMENT ITEST

```
      SEGMENT ITEST(0)
```

Ecriture des valeurs a la suite d'un SEGMENT extensible

```
C     Initialisation
      SEGINI,ITEST

C     Concaténation a la fin du SEGMENT
      ITEST(**)=-1
      ITEST(**)=-16
C     Ici la taille de ITEST(/1) vaut 2

      DO ii=1, 3
        ITEST(**)=ii**2
      ENDDO
C     Ici la taille de ITEST(/1) vaut 5
```

EXTENSION ESOPE DE FORTRAN 77

Lire des valeurs dans un SEGMENT

Exemple de structure avec le SEGMENT SAMOI

```
      SEGMENT SAMOI
        INTEGER          ITEST
        INTEGER          ITAB(N1)
        REAL*8           XTAB(N1,N2)
      ENDSEGMENT
```

Lecture des valeurs contenues dans les éléments du SEGMENT

```
      IF(SAMOI.ITEST .EQ. 1)THEN
C       Calcul de la SOMME des termes du tableau d'entier
        ISOMM=0
        DO ii=1, SAMOI.ITAB(/1)
          ISOMM = ISOMM + SAMOI.ITAB(ii)
        ENDDO

      ELSE
C       Calcul de la SOMME des termes du tableau de flottant
        XSOMM=0.D0
        ITAIL=SAMOI.XTAB(/1)   ➔ Perf : copie dans une variable locale
        DO jj=1, SAMOI.XTAB(/2)
          DO ii=1, ITAIL
            XSOMM = XSOMM + SAMOI.XTAB(ii,jj)
          ENDDO
        ENDDO
      ENDIF
```

EXTENSION ESOPE DE FORTRAN 77

Contrôler les erreurs PAS ASSEZ DE PLACE EN MEMOIRE

Renvoi a une étiquette (**label**) au lieu de s'arrêter dans GEMAT (**STOP 16**)
Peut s'ajouter à toutes les instructions **SEGXXX**

```
C     Exemple de contrôle sur un SEGINI
      SEGINI, /ERR=10/ SAMOI

 10   CONTINUE
C     On arrive ici en cas d'erreur « PAS ASSEZ DE PLACE EN MÉMOIRE »
C     On peut avoir une stratégie pour le contourner
```

EXTENSION ESOPE DE FORTRAN 77

Manipuler des super-segments (Tableaux de SEGMENTS)

Possibilité de faire des **SEGXXX** par paquets sans écrire la boucle

```
C     Déclaration d'un super-segment contenant des segments MELEME
      SEGMENT SUPERS(L1).MELEME

C     Utilisation avec les SEGXXX
      SEGINI, SUPERS(*)
      SEGADJ, SUPERS(*)
      SEGACT, SUPERS(*)
      SEGDES, SUPERS(*)
      SEGSUP, SUPERS(*)
```

Exemple de traduction FORTRAN 77 avec **SEGACT**

```
C     SEGACT,SUPERS(*)
      OO2=OOA(OOA(OOT+SUPERS)+3)
      DO 99007 OO3=1,OO2
        OO1=SUP_01(-008+OOA(OOT+SUPERS)+1+OO3)
        CALL OOOWAC(OO4,0,'SUPERS 47 MELEME ',OO1,1)
99007 CONTINUE
```

Peu utile pour les **SEGINI** et **SEGADJ** a cause des entiers dimensionnant qui sont souvent différents entre 2 SEGMENT

EXTENSION ESOPE DE FORTRAN 77

Instruction MACRO : Expression récurrente

Lorsqu'une expression revient régulièrement dans le code
Déclaration avant l'utilisation

```
      MACRO, EXPR1=(SQRT(EXP(X1+1)**2 / 2.D0)+1), EXPR2='foo'

C     X1 doit être définie avant sinon SIGFPE
      XFLOT1=EXPR1 * 2.D0
```

Traduction en FORTRAN 77 (traducteur ESOPE) :

```
C     XFLOT1=EXPR1 * 2.D0
      XFLOT1=(SQRT(EXP(X1+1)**2 / 2.D0)+1)*2.D0
```

EXTENSION ESOPE DE FORTRAN 77

Instruction MACRO : Notation fonctionnelle

Crée une FONCTION en remplaçant simplement le texte à la traduction
Déclaration avant l'utilisation

```
C     Déclaration de MA_FONCTION de 3 variables
C     Importance des parenthèses pour les priorités
      MACRO, MA_FONCTION(A,B,C)=(A+B+C)/(1.D0 + (A)*X1)

C     Utilisation
      XFLOT1 = MA_FONCTION(XVAL1+1.D0, XVAL2, XVAL3+2.D0)
```

Traduction en FORTRAN 77 (traducteur ESOPE) :

```
C     XFLOT1 = MA_FONCTION(XVAL1+1.D0, XVAL2, XVAL3*2)
      XFLOT1 = (XVAL1+1.D0+XVAL2+XVAL3+2.D0)/(1.D0+(XVAL1+1.D0)*X1)
```

EXTENSION ESOPE DE FORTRAN 77

Instruction MACRO : Type énuméré

Clarté de lecture, commentaires inutiles, déclaration avant l'utilisation

```
      MACRO, (MECANIQUE, THERMIQUE, DIFFUSION, NAVIER_STOKES)
      IF    (IFORMU .EQ. MECANIQUE    )THEN
C       Instructions
      ELSEIF(IFORMU .EQ. THERMIQUE    )THEN
C       Instructions
      ELSEIF(IFORMU .EQ. DIFFUSION    )THEN
C       Instructions
      ELSEIF(IFORMU .EQ. NAVIER_STOKES)THEN
C       Instructions
      ELSE
C       Instructions
      ENDIF
```

Traduction en FORTRAN 77 (traducteur ESOPE) :

```
      IF    (IFORMU .EQ. 1 )THEN
C       Instructions
      ELSEIF(IFORMU .EQ. 2 )THEN
C       Instructions
      ELSEIF(IFORMU .EQ. 3 )THEN
C       Instructions
      ELSEIF(IFORMU .EQ. 4 )THEN
C       Instructions
      ELSE
C       Instructions
      ENDIF
```

Instructions CASE – WHEN – WHENOTHERS – ENDCASE

Clarté, commentaires inutiles, évite les tests logiques, **GOTO**, **IF**, **ELSEIF**, **ELSE**
Imbrication des **CASE** possible

```
      MACRO, (MECANIQUE, THERMIQUE, DIFFUSION, NAVIER_STOKES)
      CASE, IFORMU
        WHEN, MECANIQUE
C         Instructions
        WHEN, THERMIQUE, DIFFUSION
C         Instructions dans le cas THERMIQUE OU DIFFUSION
        WHENOTHERS
C         Instructions
      ENDCASE
```

Traduction en fortran (traducteur ESOPE) :

```
      GOTO(99001,99002,99004,99004),IFORMU
      CALL OOCASE(IFORMU)
99001 CONTINUE
C       Instructions
        GOTO 99003
99002 CONTINUE
C       Instructions
        GOTO 99003
99004 CONTINUE
C       Instructions
99003 CONTINUE
```

Cas déclenchant une erreur de ESOPE : STOP 16

- **IFORMU** sort de la liste énumérée dans la **MACRO**
- **IFORMU** non cité dans les **WHEN** et absence de **WHENOTHERS**

```
'--- ESOPE ---   ERREUR INSTRUCTION : CASE ,',IFORMU
```

COMPILATION, ÉDITION DES LIENS, EXÉCUTION ET DÉBOGAGE

Scripts mis à disposition « clé en main » dans Cast3M

Compilation : FORTRAN 77 (`.f`), ESOPE (`.eso`), C (`.c`), MFRONT (`.mfront`)

- Manuel d'utilisation : `compilcast20 --aide`
- Version « Standard » : `compilcast20 liste_subroutines...`
- Version « Debug & Contrôle » : `compilcast20 -cd liste_subroutines...`

Edition des liens (construire un nouvel exécutable Cast3M)

- Manuel d'utilisation : `essaicast20 --aide`
- Utilisation : `essaicast20`

Exécution de Cast3M

- Manuel d'utilisation : `castem20 --aide`
- Exécution « Debug » (`gdb`) : `castem -d fichier.dgibi`

Remarque : Compiler `main.eso` en Debug pour générer des `SIGFPE` sur des variables non-initialisées

Manipuler les POINTEURS en GIBIANE (DEBUG uniquement)

Récupérer le POINTEUR d'un OBJET en GIBIANE

```
VALP1='VALE' 'POIN' Objet1 ;
```

Suivre un POINTEUR dans Cast3M au cours de l'exécution

permet d'afficher les SUBROUTINES et les instructions qui manipulent les SEGMENTS
Les manipulations suivies sont les **SEGXXX**

```
'OPTI' 'SURV' VALP1 ; //VALP1 est un numéro de POINTEUR

0--- GEMAT SURVEILLE    --- SUBROUTINE  : ACTOBJ
 ---                        INSTRUCTION : 360
 ---                        SEGACT ,   IPT1
 ---                        POINTEUR    :   2121547
 ---                        HORO SEGMENT:      1495
 ---                        HORO COURANT:      1495
 ---                        THREAD      :   0
```

Nommer un OBJET en GIBIANE a partir de son POINTEUR

Attention : Si le pointeur n'est pas du bon type il faut s'attendre à obtenir n'importe quoi !

```
OBJ1='MANU' 'OBJE' VALP1 'MAILLAGE';
```

Erreurs de manipulation de SEGMENTS

Captées automatiquement par GEMAT - ESOPE

```
--- POINTEUR ARGUMENT INVALIDE
--- DIMENSION NEGATIVE DEMANDEE
--- PAS ASSEZ DE PLACE EN MÉMOIRE
--- LE POINTEUR DESIGNE UN SEGMENT SUPPRIME
--- LE FICHIER DE DEBORDEMENT MEMOIRE EST PLEIN

--- ARGUMENT(S) ELEMENT(S) DE SEGMENT ...
--- DESTRUCTION MÉMOIRE
--- DEADLOCK DETECTEE
```

Stopper l'exécution de Cast3M au moment de l'erreur GEMAT

- ➔ ESOPE émet toutes ses erreurs dans la SUBROUTINE `oooerr.eso`
- ➔ Dans **`gdb`**, faire **`break`** `oooerr_` avant de faire **`run`**
- ➔ Une fois **`gdb`** arrêté dans `oooerr_`, pour obtenir l'arbre d'appel des SUBROUTINES, faire :
  - En monothread : **`where`**
  - En multithreads : **`thread apply all where`**

Erreurs de manipulation des ARGUMENTS de SEGMENTS

ARGUMENT(S) ELEMENT(S) DE SEGMENT ...

- Eléments de SEGMENT en arguments d'une SUBROUTINE ➔ **SEGXXX** plus autorisés (FORTRAN pur)
- Protection de GEMAT - ESOPE : retassement mémoire ➔ adresse transmise plus valide !
  - ➔ Trouver l'erreur : en « Debug », faire un break `oooerr_` et trouver les `OOx` dans les ARGUMENTS
  - ➔ Corriger l'erreur : passer le POINTEUR en ARGUMENT à la place

```
      SEGMENT SAMOI
        REAL*8 XTAB(N1)
      ENDSEGMENT

      N1=3
      SEGINI,SAMOI

C     Instructions ESOPE
      CALL SUB1(SAMOI)
      CALL SUB2(SAMOI.XTAB(1))
```

SAMOI est un **POINTEUR** (type INTEGER)

XTAB est un **ARGUMENT du SEGMENT SAMOI**

```
C     Traduction en F77
      CALL SUB1(SAMOI)
C     CALL SUB2(XTAB(1))
      OO5=OOA(OOT+SAMOI)
      OOW(1)=OOW(1)+1
      CALL SUB2(XTA_01(-008+OO5+8+1))
      OOW(1)=OOW(1)-1
```

Appel à SUB1 inchangé

Appel à SUB2 commenté

Indicateur ARGUMENT ELEMENT DE SEGMENT

L'appel a SUB2 contient des OOx

R.A.Z Indicateur ARGUMENT ELEMENT DE SEGMENT

Erreurs de programmation : écrasement de mémoire

`DESTRUCTION MÉMOIRE` : boucle trop grande quelque part !

➔ Ecrasement en **FORTRAN** : Désactiver la mise en queue des SEGDES/SEGSUP

```
'OPTI' 'SURV' 1;
```

Ajouter des **SEGACT**/**SEGDES** progressivement dans le code source

Déterminer quelle SUBROUTINE déborde…

➔ ***Conseil*** : Programmez en ESOPE et pas en FORTRAN pur

➔ Ecrasement en **ESOPE** : `compilcast19 -cd` de l'arbre d'appel

Erreur trouvée en 1 étape grâce à ESOPE (`SEGSEV` lors du dépassement)

Les différents SIGNAUX du système

SIGSEV : Tentative d'accès invalide à une adresse mémoire valide (ex : essayer d'écrire là où on ne peut que lire)

SIGFPE : Erreur arithmétique fatale (division par zéro, `overflow`, utilisation d'un `Nan`, etc.)

SIGABRT : Renvoyé par `malloc()` ou `free()` lors de troubles sérieux (comme un overflow dans la mémoire demandée). La fonction invoquée est `abord()`

SIGBUS : Tentative d'accès à une adresse mémoire invalide (overflow) (différent de **SIGSEV**)

SIGILL : Tentative d'exécution d'une instruction illégale. Le CPU tente d'exécuter une instruction qu'il ne connait pas. Saut à une adresse qui n'est pas dans l'aire définie par le programme. Compilation pour une mauvaise architecture. Ne rien retourner alors qu'on a spécifié un retour, etc.

Utilisation suffisante de gdb

run :	Lance l'exécution dans l'environnement **gdb**
break :	Permet l'arrêt de l'exécution au niveau d'un symbole, d'une ligne ou d'une instruction
where :	Affiche l'arbre d'appel jusqu'au symbole courant
up :	Remonte d'un symbole dans l'arbre d'appel
down :	Descend d'un symbole dans l'arbre d'appel
l :	Liste le `FORTRAN` (si compilé en DEBUG) à la ligne courante ou la ligne demandée
c :	Reprend l'exécution du programme

CRÉER UN NOUVEL OPÉRATEUR

CRÉER UN NOUVEL OPÉRATEUR

Ecrire la spécification ***avant*** de programmer en ESOPE (`.notice`)

Exemple de syntaxe : opérateur de calcul statistique sur les objets `'LISTREEL'`

```
* Syntaxe GIBIANE (Notice de l'opérateur)
  FLOT1='STAT' LREEL1 | ('MOYE') | ;  //Calcule la moyenne
                      |  'MEDI'  |    //Calcule la médiane
                      |  'VARI'  |    //Calcule la variance

* Par défaut (en l'absence de MOT CLE), on souhaite obtenir la moyenne
```

Ajout du MOT du nouvel opérateur dans la SUBROUTINE `pilot.eso`

- ***1ère manière*** : Chercher la chaine `'....'` (ancien opérateur) dans **`MDIR1`**, **`MDIR2`** ou **`MDIR3`**

  ➔ Dans notre cas c'est dans **`MDIR1`** entre `'SUIT'` et `'VALP'` **`4 caractères à changer`**

- ***2ème manière*** : Aucun ancien opérateur restant ➔ agrandir **`MDIR3`**

```
C     Agrandissement du DATA MDIR3
      NDIR3=152
      NDIR3=153

      ...instructions...

       DATA MDIR3/'NUAG','WEIP','KHIS','KOPS','FSUR','FLAM','ELNO',
     > ...
     > 'ANNO'/
     > 'ANNO','STAT'/
```

CRÉER UN NOUVEL OPÉRATEUR

Ajout de l'appel a la nouvelle SUBROUTINE dans `pilot.eso`

- ***Attention*** : Le nom de la SUBROUTINE doit idéalement correspondre à l'opérateur

  Nom retenu : `stati.eso`

  Arguments : `aucun` ➔ Ils seront lus dans `stati.eso`

```
C     Ajout de l'appel au nouvel opérateur 'STAT' entre 'SUIT' et 'VALP'
 260  CALL SUITE
      GOTO 1

 261  CONTINUE
C     ANCIEN APPEL A L'OPERATEUR
 261  CALL STATI
      GOTO 1

 262  CALL VALPRO
      GOTO 1
```

2 lignes à effacer

2 lignes à ajouter

CRÉER UN NOUVEL OPÉRATEUR

Lire des arguments GIBIANE en ESOPE

```
C Remarques générales : ICODE = 1 ➔ lecture OBLIGATOIRE (Erreur sinon)
C                       IRETOU= 0 ➔ Lecture infructueuse

C Après chaque LIRXXX : IF(IERR .NE. 0) RETURN (avec -INC CCOPTIO)

C     Lecture d'un 'FLOTTANT'
      CALL LIRREE(XLU,ICODE,IRETOU)

C     Lecture d'un 'ENTIER'
      CALL LIRENT(ILU,ICODE,IRETOU)

C     Lecture d'un 'LOGIQUE'
      CALL LIRLOG(BLU,ICODE,IRETOU)

C     Lecture d'un 'MOT'    : Chaine de caractères "libre"
      CALL LIRCHA(CLU,ICODE,IRETOU)

C     Lecture d'un 'MOT'    : MOT CLE contenus dans une liste connue
      CALL LIRMOT(MOLIST,ITAILL,IRETOU,ICODE)
C       MOLIST: DATA ou Tableau contenant les mots-cles
C       ITAILL: Nombre d'éléments a considérer dans la liste
C       IRETOU: Position dans la liste (0 si absent)

C     Lecture d'un 'OBJET' : Autres types que précédents
      CALL LIROBJ(CTYPE,IPLU,ICODE,IRETOU)
C       CTYPE : Chaine du type de l'objet à lire (ex:'MAILLAGE')
C       IPLU  : POINTEUR vers l'objet lu (si lecture fructueuse)
```

CRÉER UN NOUVEL OPÉRATEUR

Ecrire des arguments GIBIANE en ESOPE

```
C     Ecriture d'un 'FLOTTANT'
      CALL ECRREE(XECR)

C     Ecriture d'un 'ENTIER'
      CALL ECRENT(IECR)

C     Ecriture d'un 'LOGIQUE'
      CALL ECRLOG(BECR)

C     Ecriture d'un 'MOT'
      CALL ECRCHA(CECR)

C     Ecriture d'un 'OBJET' : Autres types que précédents
      CALL ECROBJ(CTYPE,IPECR)
C       CTYPE : Chaine du type de l'objet à écrire (ex:'MAILLAGE')
C       IPECR : POINTEUR vers l'objet
```

CRÉER UN NOUVEL OPÉRATEUR

- **Niveau 1 : Détecter TOUTES les syntaxes spécifiées : *aucune autre***

```
      SUBROUTINE STATI
      IMPLICIT INTEGER(I-N)
      IMPLICIT REAL*8(A-H,O-Z)
C     Déclarations (includes, MOTS-CLES)

C     Lecture d'un 'LISTREEL' obligatoire et activation

C     Lecture d'un 'MOT-CLE' optionnel ==> Défaut = 1 (Moyenne)

C     Appel a une SUBROUTINE qui fait le travail avec des ARGUMENTS

C     Ecriture du résultat en GIBIANE

      END
```

CRÉER UN NOUVEL OPÉRATEUR

Niveau 2 : Appel a une SUBROUTINE avec des ARGUMENTS

Intérêt : Pouvoir appeler l'opérateur directement en ESOPE
Travaille à SEGMENTS activés pour des questions de performance

```
      SUBROUTINE STATI1(MLREEL,IPOSI,XECR)
      IMPLICIT INTEGER(I-N)
      IMPLICIT REAL*8(A-H,O-Z)
C     Déclarations (includes)

C     Test de validité des données

C     Initialisation

C     Différents cas de figure
```

LES MESSAGES DANS CAST3M

LES MESSAGES DANS CAST3M

- **Tout dans un seul fichier :** …/data/GIBI.ERREUR

  Clarté pour l'utilisateur devant le message

  Gestion multi-langue évidente

  erreur.eso : La seule **SUBROUTINE** à invoquer pour le développeur

  Débogage évident pour stopper Cast3M sur un message

- **Qu'est-ce qu'un message dans Cast3M**

  Une langue : 'OPTI' 'LANG' MOT1; en GIBIANE ou MOT1 peut valoir :

  - 'FRAN' français par défaut (le défaut se change en tête de GIBI.ERREUR)
  - 'ANGL' anglais disponible 'OPTI' 'LANG' 'ANGL'; en GIBIANE
  - 'ESPA' espagnol libre d'être ajouté (traduire tous les messages existants)

  Un numéro de message : type **ENTIER**

  Un niveau d'erreur : type **ENTIER**

  - Niveau d'erreur 0 : Message affiché, Cast3M poursuit (ce n'est pas une erreur)
  - Niveau d'erreur 1 : Message affiché, ERREUR émise, Cast3M rend la main, en sortie STOP 4
  - Niveau d'erreur 2 : Message affiché, ERREUR émise, Cast3M rend la main, en sortie STOP 8
  - Niveau d'erreur 3 : Message affiché, ERREUR émise, Cast3M quitte, en sortie STOP 12

  Un texte (sur 2 lignes maximum) pouvant contenir des variables de type :

  - MOT : %m1:8 contenu de MOTERR(1:8) (Voir CCOPTIO.INC)
  - ENTIER : %i1 à %i10 contenus dans le tableau INTERR (Voir CCOPTIO.INC)
  - FLOTTANT : %r1 à %r10 contenus dans le tableau REAERR (Voir CCOPTIO.INC)
  - LOGIQUE : %b1 à %b10 contenus dans le tableau BOOERR (Voir CCOPTIO.INC)

LES MESSAGES DANS CAST3M

Ajouter un message dans `GIBI.ERREUR`

1- Vérifier les messages existants

2- Ajouter le message dans chacune des langues (au bon endroit)

- début de chaque langue pour les numéros de message négatifs (rechercher la balise `9997 0`)
- fin de chaque langue pour les numéros de message positifs

Exemple de message

```
1113 2
Ligne1 : Le mot '%m1:8', l'entier %i1
1113 2
Ligne2 : le flottant %r2, le logique %b1
```

Numéro de message : **1113** | niveau d'ERREUR : **2**

Message sur 2 lignes :
Répéter Numéro de message et niveau d'ERREUR

Invoquer le message 1113 en ESOPE

En ESOPE :

```
      MOTERR(1:8) ='MON__MOT'
      INTERR(1)    = 57
      REAERR(2)    = 5.D2
      BOOERR(1)    =.TRUE.
      CALL ERREUR(1113)
      RETURN
```

Place `IERR` au niveau d'erreur `2`

```
 *****  ERREUR 1113 ***** dans l'operateur ....
 Ligne1 : Le mot 'MON__MOT', l'entier 57
 Ligne2 : le flottant 0.05 , le logique VRAI
```

CRÉER UN NOUVEAU MODÈLE MÉCANIQUE

CRÉER UN NOUVEAU MODÈLE MÉCANIQUE

Les étapes à suivre pour créer un nouveau modèle

1- Spécification technique (Notice)

2- Donner les noms des composantes associées au modèle

- Nom des composantes **PRIMALES** (déplacement) Défaut : `'UX'`, `'UY'`, `'UZ'`
  ➔ `iddepl.eso`

- Nom des composantes **DUALES** (forces) Défaut : `'FX'`, `'FY'`, `'FZ'`
  ➔ `idforc.eso`, `iddual.eso`

- Nom des composantes du **GRADIENT** Défaut : `'UX,X'`, `'UX,Y'`, `'UX,Z'`
  `'UY,X'`, `'UY,Y'`, `'UY,Z'`
  `'UZ,X'`, `'UZ,Y'`, `'UZ,Z'`
  ➔ `idgrad.eso`

- Nom des composantes des **CONTRAINTES** Défaut : `'SMXX'`, `'SMYY'`, `'SMZZ'`
  `'SMXY'`, `'SMXZ'`, `'SMYZ'`
  ➔ `idcont.eso`

- Nom des composantes des **DEFORMATIONS** Défaut : `'EPXX'`, `'EPYY'`, `'EPZZ'`
  `'GAXY'`, `'GAXZ'`, `'GAYZ'`
  ➔ `iddefo.eso`

- Nom des composantes du **MATERIAU**
  ➔ `nomate.eso`, `idmatr.eso`, `idcara.eso`

```
modpla.eso/idplas.eso (PLASTIQUE)
modflu.eso/idflua.eso (FLUAGE)
modvis.eso/idvisc.eso (VISCOPLASTIQUE)
modend.eso/idendo.eso (ENDOMMAGEMENT)
modple.eso/idplen.eso (PLASTIQUE ENDOMMAGEMENT)
modenl.eso/idelnl.eso (ELASTIQUE NON-LINEAIRE)
```

CRÉER UN NOUVEAU MODÈLE MÉCANIQUE

- Nom de la composante pour la **TEMPERATURE** Défaut : `'T'`
  ➔ `idtemp.eso`

- Nom de la composante des **CONTRAINTES PRINCIPALES**
  ➔ `idprin.eso` Défaut : `'SI11'`, `'SI22'`, `'SI33'`
  `'COX1'`, `'COX2'`, `'COX3'`
  `'COY1'`, `'COY2'`, `'COY3'`
  `'COZ1'`, `'COZ2'`, `'COZ3'`

- Nom des composantes des **VARIABLES INTERNES**
  ➔ `idvari.eso`

- Nom des composantes des **DEFORMATIONS INELASTIQUES**
  ➔ `iddein.eso` Défaut : `'EIXX'`, `'EIYY'`, `'EIZZ'`
  `'GIXY'`

Ajout d'un nouveau modèle : opérateur 'MODE'

Description théorique de la loi visée : Fluage de Norton

$$\dot{\varepsilon}_{eq} = \left(\frac{\sigma_{eq}}{C_1}\right)^{C_2}$$

$$\underline{\underline{\delta\varepsilon_{el}}} = \underline{\underline{\delta\varepsilon_{tot}}} - \delta t \cdot \frac{3}{2} \cdot \dot{\varepsilon}_{eq} \cdot \frac{\underline{\underline{\sigma_D}}}{\sigma_{eq}}$$

$$\underline{\underline{\delta\sigma}} = 2 \cdot \mu \cdot \underline{\underline{\delta\varepsilon_{el}}} + \lambda \cdot Tr\left(\underline{\underline{\delta\varepsilon_{el}}}\right)$$

Définition du MODELE et du MATERIAU en GIBIANE

```
MOD1='MODE' MASSIF 'MECANIQUE' 'ELASTIQUE' 'FLUAGE' 'CAST3M_FOR_DEV' ;
MAT1='MATE' MOD1   'YOUN' YOU1 'NU' NU1 'C1' COE1 'C2' COE2 ;
```

Incrémenter le numéro de modèle : `nomate.eso`

```
      ELSE IF (IMOD.EQ.19) THEN
C       KELVIN
C       INATU = 174
C       AM 3/3/17 MODELE INDISPONIBLE
        CMATE=' '
        IMATE=0
        INATU=0

      ELSE IF (IMOD.EQ.20) THEN
C       CAST3M_FOR_DEV
        INATU = 186
      ENDIF
```

Ajout du nouveau numéro de matériau

CRÉER UN NOUVEAU MODÈLE MÉCANIQUE

Nommer la loi de fluage CAST3M_FOR_DEV : à ajouter dans modflu.eso

```
      ~~NMOD=19~~
      NMOD=20

      MOMODL(1)='NORTON'
      ...autres modèles de fluage...
      MOMODL(19)='KELVIN'
      MOMODL(20)='CAST3M_FOR_DEV'
      END
```

Modification du nombre de lois de fluage

Ajout du nom de notre modèle

Composantes matériaux : C_1 et C_2 à ajouter dans idflua.eso

```
      ...caractéristiques des modèles de fluage existant...
      ELSE IF (IPLAC.EQ.20)THEN
C       CAST3M_FOR_DEV
        JGM0=JGOBL
        JGOBL=JGM0+2
        TABOBL(JGM0+1)='C1'
        TABOBL(JGM0+2)='C2'
        GOTO 9999
```

Noms des composantes du matériaux

CRÉER UN NOUVEAU MODÈLE MÉCANIQUE

Ajouter les noms des variables internes : `idvari.eso`

```
      ...Variables internes des autres modèles
      ELSE IF (MATEPL .EQ. 186)THEN
C       CAST3M_FOR_DEV
        NBROBL=0
        SEGINI,NOMID
        IPCOMP=NOMID
```

Listing du modèle en GIBIANE

```
   POINTEUR SUR L'OBJET MAILLAGE :   2121127
   TYPE D'ELEMENT FINI:     QUA4
   FORMULATION          : MECANIQUE
   MODELE DE MATERIAU : ELASTIQUE  ISOTROPE  FLUAGE  CAST3M_FOR_DEV
   Hypothèse de déformations : Quadratique
   Liste des noms de composantes de DEPLACEM
          UX   UY
   Liste des noms de composantes de FORCES
          FX   FY
   ...
   Liste des noms de composantes de MATERIAU
          YOUN NU   C1   C2
          RHO  ALPH VISQ
```

En **BLEU** : exigé par l'élasticité
En **ROUGE** : exigé par le fluage

Ajout de la loi de comportement : opérateur 'COMP'

Ajouter le numéro de loi dans le **GOTO** de coml6.eso (ligne 680)

```
      GOTO( 8, 8,
C inplas  0 1  2  3  4  5   6  7   8  9  10  11 12  13 14 15
     &    7,7, 8, 7, 7, 7,111, 7,111, 8,111,111, 7,111, 8, 7,
        ...
C          180 181 182 183 184 185 186
     &     111,111,111,111,111,111,12 )jnppla
```

Ajout du numéro de loi pour aller au Label 12 (coml12.eso)

Ajouter l'appel a la loi de comportement dans coml12.eso

```
      ELSEIF(INPLAS.EQ.186) THEN
C Modele CAST3M_FOR_DEV
        CALL C3MFDE(NSTRS,TEMP0,TEMPF,DEPST,
     &              XMAT0,SIG0,EPIN0,XMATF,SIGF,EPINF)


      ELSE
        WRITE(IOIMP,*) 'Branchement incorrect dans COML12'
        CALL ERREUR(5)
      ENDIF
```

Appel au calcul de la loi de comportement

Matériau, contraintes, déformation inélastique initial & final

CRÉER UN NOUVEAU MODÈLE MÉCANIQUE

Intégrer la loi de comportement

```
      SUBROUTINE C3MFDE(NSTRS,TEMP0,TEMPF,DEPST,
     &                  XMAT0,SIG0,EPIN0,XMATF,SIGF,EPINF)
      IMPLICIT INTEGER(I-N)
      IMPLICIT REAL*8(A-H,O-Z)

C Declarations : ARGUMENTS & TABLEAUX de TRAVAIL
      REAL*8 DEPST(*)
      REAL*8 XMAT0(*) ,SIG0(*),EPIN0(*)
      REAL*8 XMATF(*) ,SIGF(*),EPINF(*)
      REAL*8 XDEVIF(6),DEPSEL(6),SIGFP(6)

C Initialisations : Coefficients de Lame, DEPS_EL=DEPS_TOT
      XMU0 = 0.5D0*XMAT0(1)/(1.0D0 + XNU0)
      XLA0 = XMAT0(1)*XNU0/((1.0D0 + XNU0)*(1.0D0 - 2.D0*XNU0))

      DO ii=4,NSTRS
C       Attention : composante GAXY ==> GAXY=2.D0 * EPXY
        DEPSEL(ii)=DEPST(ii)/2.D0
      ENDDO
      DO ii=1,3
        DEPSEL(ii)=DEPST(ii)
      ENDDO

C Boucle de résolution : SIGF(Hooke isotrope), SIGDEV, VMIS, DEPS_EL
C Test que SIGF sont REELLES
C Restitution des résultats convergés : SIGF (Hooke isotrope), EPINF
      END
```

CRÉER UN NOUVEL ÉLÉMENT FINI

CRÉER UN NOUVEL OBJET

CRÉER UN NOUVEL OBJET

- **1- Créer la structure de l'objet dans un fichier include : `SMOBJ.INC`**
- **2- Utiliser & manipuler l'objet en ESOPE : `-INC SMOBJ`**
- **3- Créer/modifier les opérateurs pour manipuler l'objet**

Nommer l'objet dans un opérateur pour y accéder en GIBIANE :

```
      CALL ECROBJ('LE_TYPE',LE_POINTEUR)
```

Récupérer l'objet dans un opérateur/directive :

```
      CALL LIROBJ('LE_TYPE',LE_POINTEUR,ICODE,IRETOU)
      IF(IERR .NE. 8)RETURN
```

Directive **`LIST`** : Il faut pouvoir lister l'objet (modifier `prlist.eso`)

```
      PARAMETER (NMO=38)
      DATA LISMO / 'MOT       ','ENTIER  ','FLOTTANT','LOGIQUE ',
     $             ...
     $             'ANNOTATI','NEW_OBJE'/
       ...
GOTO ( 50, 60, 70, 80, 90,100,110,120,130,140,150,160,170,180, .
190,200,210,220,230,240,250,260,270,280,290,300,310,320, .
330,340,350,360,370,380),IPP
       ...
C LISTE D'UN OBJET 'NEW_OBJE'
380   CALL ECOBJE(IRET,jentet)
      GOTO 50000
```

Agrandissement du DATA

Ajout au **`DATA`** de l'élément **`38`**
Renvoi en **`380`** dans le **`GOTO`**

Invoquer et programmer la SUBROUTINE qui liste
`IRET` : POINTEUR sur l'objet
`jentet` : Vaut 1 si résumé (`'LIST' 'RESU' OBJE1;`)
Mettre les messages dans `GIBI.ERREUR` (**FRAN** & **ANGL**)

CRÉER UN NOUVEL OBJET

Liste minimale des opérateurs à modifier pour un nouvel OBJET

- **LIST** : Liste le contenu des OBJETS
- **MENA** : Suppression des SEGMENTS non reliés à des OBJETS nommés
- **SAUV** : Sauvegarde d'une session Cast3M
- **REST** : Restitution d'une sauvegarde

Liste des opérateurs auxquels on peut avoir recours

- **EXTR** : Idéalement, doit pouvoir extraire tout le contenu de l'OBJET
- **CHAN** : Changer une des grandeurs de l'OBJET (Crée un nouvel OBJET)
- **ET** : Concaténation des OBJETS
- **MANU** : Création d'OBJETS « à la main »
- **VIDE** : Création d'OBJETS « vides »
- **COPI** : Duplique entièrement un OBJET

CRÉER UN NOUVEL OBJET

Directive **`LIST`** : Lister le contenu d'un OBJET en GIBIANE

modifier `prlist.eso` : Ajouter son objet au **`DATA`**
Ajouter le nouvel objet dans le **`GOTO`**
Ajouter l'appel a la SUBROUTINE qui va lister

CRÉER UN NOUVEL OBJET

Directive **MENA** : Conserver l'objet et son contenu après le ménage

modifier `menag6.eso` : protègent (avec `menage2.eso`) les SEGMENTS qui ne doivent pas être supprimés par le ménage
NPGCD est la plus petite différence entre 2 descripteurs ESOPE successifs (8 à priori, voir `IOODES.INC`).
NPGCD est calculé dans `crepil.eso`

modifier `typfil.eso` : Renvoie la file si on donne le type et inversement
Ajouter une nouvelle file avec l'objet

modifier `expil.eso` : Ajouter le nouvel objet dans le **GOTO**
➔ **IIICHA=0** appel a `expil.eso` depuis ménage
➔ **IIICHA=1** appel a `expil.eso` depuis sauvegarde
Pour la sauvegarde, si des POINTEURS sont partagées entre plusieurs OBJETS (ex : LISTREEL) : passer les POINTEURS en question en négatif (sinon ils seront sauvegardes plusieurs fois)
Ajouter (`ajoun.eso`) les refs vers d'autres OBJETS aux piles correspondantes

CRÉER UN NOUVEL OBJET

Directive **SAUV** : Sauvegarder les objets dans la session Cast3M courante

modifier `wrpil.eso` : Permet d'aiguiller vers la bonne pile pour la sauvegarde
Ajouter le nouvel objet dans le **GOTO**
Ajouter l'appel a la SUBROUTINE qui va travailler

modifier `restpi.eso` : Restaures les POINTEURS mis négatifs (le cas échéant)
Ajouter le nouvel objet dans le **GOTO** (ou **GOTO** 1099)

CRÉER UN NOUVEL OBJET

Directive **`REST`** : Restituer le nouvel objet dans la session Cast3M courante

modifier `lipil.eso` : Permet d'aiguiller vers la bonne pile pour la restitution
Ajouter le nouvel objet dans le **`GOTO`**
Ajouter l'appel a la SUBROUTINE qui va travailler

PROGRAMMATION PARALLÈLE

MESURE & VISUALISATION DES PERFORMANCES

MESURE & VISUALISATION DES PERFORMANCES

Mesure des durées dans Cast3M (Opérateur TEMP En GIBIANE)

'TEMP' 'IMPR' : Affichage par opérateur et par assistant (Temps horloge, …)

```
Temps Horloge (ms) par OPERATEUR et par ASSISTANT
 OPTI            0            0            0            0            0
 ET              0            0            0            0            0
 TRAC            7            0            0            0            0
 DIFF            0            0            0            0            0
 OUBL            0            0            0            0            0
 COMP            0         2351         2383            0            0
```

'TEMP' 'IMPR' 'BOUC': Affichage par boucle REPE exécutée

Boucle :	Duree Horloge(ms)	Duree CPU (ms)	Efficacte (%)	Nombre d'appels
BEXTERN	31	62	200	2
BO_BOTH	33	15	45	8
BINCO1	9	0	0	8
BH	15	15	100	16

'TEMP' 'IMPR' 'PROC': Affichage par procédure PROC exécutée

Procedure:	Duree Horloge(ms)	Duree CPU (ms)	Efficacte (%)	Nombre d'appels
PASAPAS	40	46	115	1
PAS_DEFA	49	62	126	1
PAS_INIT	16	15	93	1
PAS_ETAT	12	15	125	6
PAS_VERM	0	0	0	6
PAS_MATE	0	0	0	6
UNPAS	2918	5296	181	5
PAS_RESU	15	15	100	5

MESURE & VISUALISATION DES PERFORMANCES

Couverture de code

gcov : Compilation avec `compilcast20 --gcov *.eso`

Couverture : `castem20 *.dgibi`

lcov : Je n'ai pas réussi à le faire fonctionner avec **gcc 9.2**

Analyse de la performance

perf : Performance analysis tools for linux

Tutorial : https://perf.wiki.kernel.org/index.php/Tutorial#Sampling_with_perf_record

Analyse en « *runtime* » : `perf top`

Analyse :

```
perf record --call-graph=lbr -e instructions:u castem20 fichier.dgibi
```

Post-Traitement :

```
perf report --call-graph=fractal,callee     (Qui a appelé)
perf report --call-graph=fractal,caller     (Qui est appelé)
perf report --no-children                   (Durée Self)
```

Post-Traitement graphique

hotspot : https://github.com/KDAB/hotspot

Nombreuses améliorations dans Cast3M depuis mi-2019 grâce à **hotspot**

Lire les tutos pour les intéressés

MESURE & VISUALISATION DES PERFORMANCES

File Settings View Help

Summary | Bottom Up | Top Down | Flame Graph | Caller / Callee

Parser Errors

Failed to find ELF file for an instruction pointer address. This can break stack unwinding and lead to missing symbols.
Could not find ELF file for libdl-2.22.so. This can break stack unwinding and lead to missing symbols.
Module "libdl-2.22.so" is missing (some) debug symbols.
Module "[vdso]" is missing (some) debug symbols.

Summary

Command:	perf record --call-graph=lbr -e instructions:u castem MFRONT_Steel_Creep_test.dgibi
Run Time:	15.806s
Processes:	20
Threads:	132
Total Samples:	74655 (4.723KHz)
instructions:u:	1.182E+11 (7.466E+04 samples, 100% of total, 4.723KHz)
Lost Chunks:	0

Top Hotspots

Event Source:

Symbol	Binary
mamu33_	cast
comval_	cast
comsor_	cast
monde1_	cast
jacobi_	cast

System Information

Host Name:	semtpc6
Linux Kernel Version:	4.18.20-desktop-1.mga6
Perf Version:	4.18.20-1.mga6
CPU Description:	Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
CPU ID:	GenuineIntel,6,79,1
CPU Architecture:	x86_64
CPUs Online:	56
CPUs Available:	56
CPU Sibling Cores:	[0-13,28-41], [14-27,42-55]
CPU Sibling Threads:	[0,28], [1,29], [2,30], [3,31], [4,32], [5,33], [6,34], [7,35], [8,36], [9,37], [10,38], [11,39], [12,40], [13,41], [14,42], [15,43], [16,44], [17,45], [18,46], [19,47], [20,48], [21,49], [22,50], [23,51], [24,52], [25,53], [
Total Memory:	67,4 Go

MESURE & VISUALISATION DES PERFORMANCES

File Settings View Help

Summary | Bottom Up | Top Down | Flame Graph | Caller / Callee

Chercher

Symbol	Binary	instructions:u (incl.)
mamu33_	cast	9.72%
comval_	cast	8.43%
comsor	cast	6.68%
coml6_	cast	6.68%
coml2_	cast	6.35%
coml_	cast	6.13%
pilot_	cast	5.61%
main2_	cast	2.55%
ooostart	cast	0.287%
start_thread	libpthread-2.22.so	0.287%
monde1_	cast	6.56%
jacobi_	cast	3.79%
<.plt+5410>	libm-2.22.so	3.35%
meladd_	cast	3.15%
bst_	cast	3.12%
chame2_	cast	2.95%
chole3_	cast	2.89%
bsig_	cast	2.87%
noepr2_	cast	2.31%
bmatst_	cast	1.83%
ddotpv_	cast	1.82%
castem::CastemBehaviourHandler<(castem::CastemBehaviourType)0, (tfel::material::ModellingHypothesis::Hypothesis)6, tfel::material::SteelCreep>::Integrator<false, false>::exe(...	libUmatSteel.so	1.63%
fuse_	cast	1.59%
__exp1	libm-2.22.so	1.57%
adchve_	cast	1.57%
epsi2_	cast	1.35%
mucpr1_	cast	1.35%
mucpr2_	cast	1.35%
oootdl_	cast	1.22%
crechp_	cast	1.21%
chole3i_	cast	1.03%
bsigm1_	cast	0.999%
coml8_	cast	0.979%
ooogll_	cast	0.865%

Time Line Filter Chercher Event Source: instruction

Source	Events
CPUs	
CPU #0	
Threads	
caste...	
stty (#...	
getcon...	
getcon...	
rm (#3...	
cp (#3...	
chmod...	
#3616...	
df (#3...	
grep (...	
wc (#3...	
arch (...	

MESURE & VISUALISATION DES PERFORMANCES

File Settings View Help

Summary | Bottom Up | Top Down | Flame Graph | Caller / Callee

Chercher

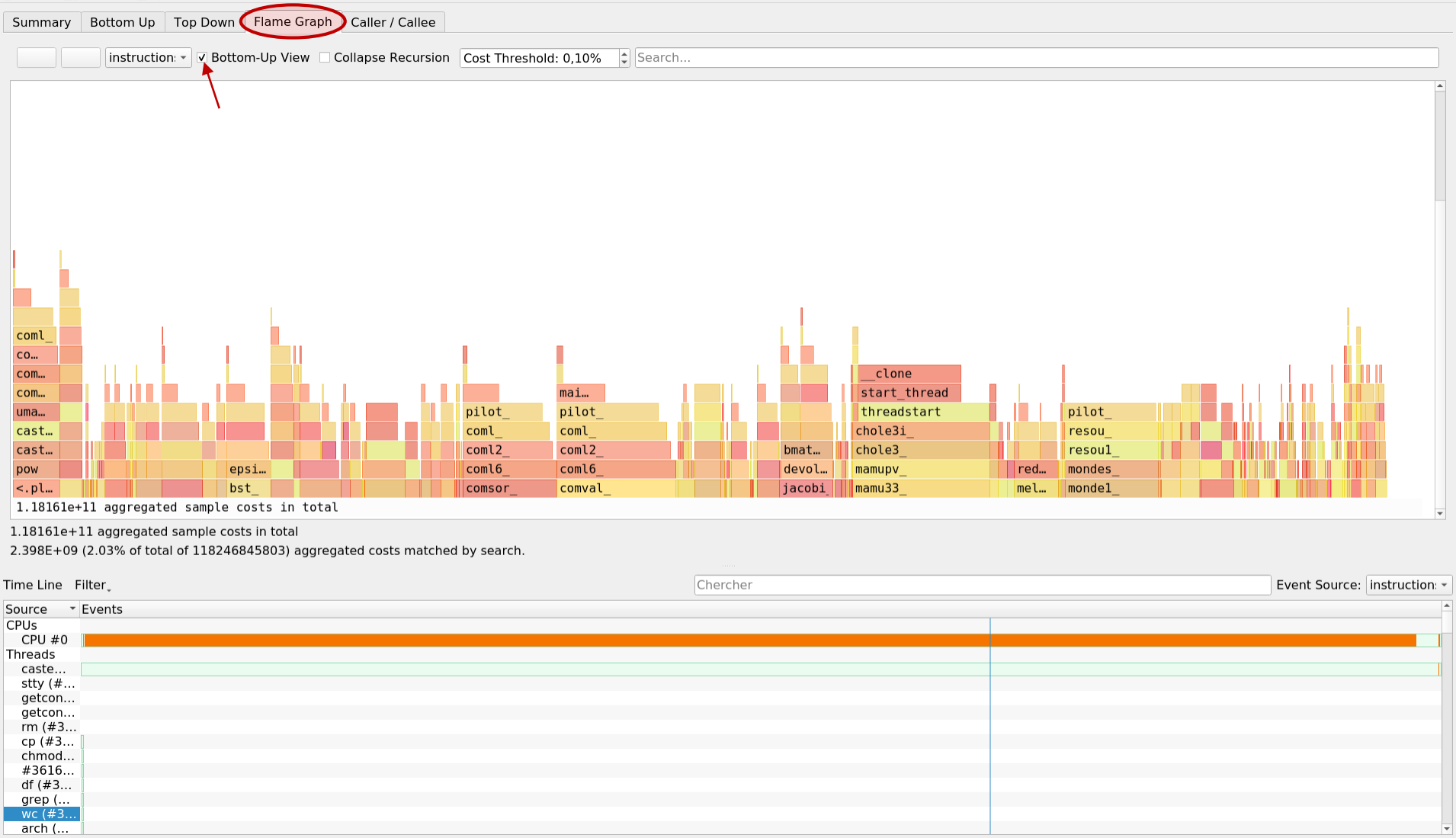
Symbol	Binary	instructions:u (incl.)	instructions:u (self)
pilot_	cast	48.8%	0.016%
main2_	cast	20.4%	0%
pilot_	cast	20.4%	0.000958%
coml	cast	10.1%	0.0115%
coml2_	cast	9.1%	0.00632%
coml6_	cast	9.06%	0.228%
coml8_	cast	3.12%	0.328%
umatsteelcreep	libUmatSteel.so	2.45%	0%
wkuma1_	cast	0.195%	0.195%
umatext_	cast	0.147%	0.147%
comval_	cast	2.99%	2.99%
comsor_	cast	2.26%	2.26%
comara_	cast	0.248%	0.248%
locara_	cast	0.1%	0.1%
comcri_	cast	0.0471%	0%
comouw_	cast	0.0319%	0.0262%
comfin_	cast	0.0152%	0%
ooowsu_	cast	0.00695%	0%
compou_	cast	0.00586%	0.00586%
ooowin_	cast	0.00434%	0.00129%
doxe_	cast	0.000811%	0.000811%
oooprl_	cast	8.84E-07%	0%
ooowin_	cast	0.0158%	0%
ooowsu_	cast	0.00903%	0.00138%
ooowad_	cast	0.00707%	0%
reduaf_	cast	0.746%	0.00165%
comred_	cast	0.0899%	0.0366%
actobj_	cast	0.053%	0.000416%
ooowsu_	cast	0.0417%	0.00709%
ooowin_	cast	0.0216%	0%
ooowad_	cast	0.00764%	0%
memmove@plt	cast	0.00112%	0.00112%
oooprl_	cast	0.000852%	0%
ensi	cast	4.4%	0%

Time Line Filter Chercher Event Source: instruction

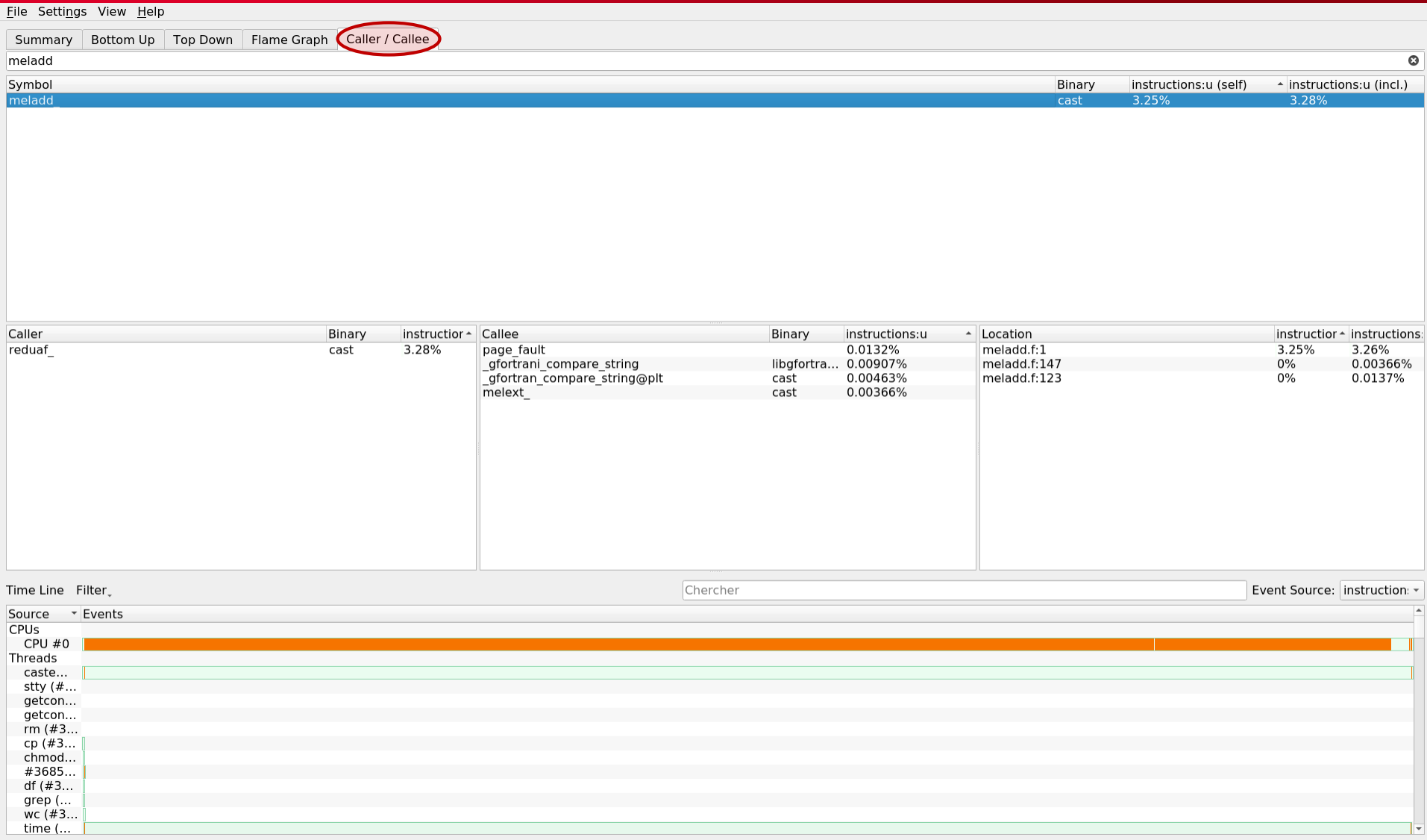
Source	Events
CPUs	
CPU #0	
Threads	
caste…	
stty (#…	
getcon…	
getcon…	
rm (#3…	
cp (#3…	
chmod…	
#3616…	
df (#3…	
grep (…	
wc (#3…	
arch (…	

MESURE & VISUALISATION DES PERFORMANCES

MESURE & VISUALISATION DES PERFORMANCES



MESURE & VISUALISATION DES PERFORMANCES



GLOSSAIRE DE SOURCES UTILES

GLOSSAIRE DE SOURCES UTILES

MANIPULATION du GIBIANE en ESOPE

Lecture d'objets : `lirree, lirent, lirlog, lircha, lirmot, lirobj`
Ecriture d'objets : `ecrree, ecrent, ecrlog, ecrcha, ecrobj`
Position dans certaines piles : `posree, poslog, poscha`

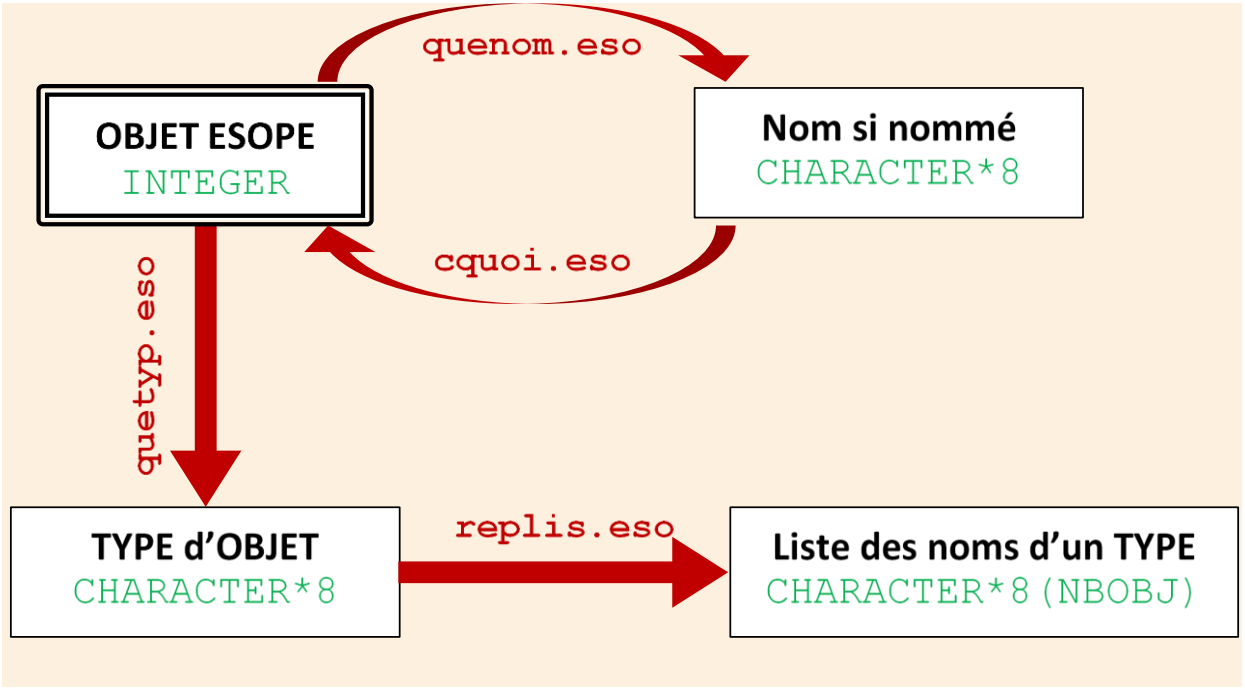
`actobj` : **Activer** un objet et tout son contenu

`prlist` : **Lister** un objet

`acctab` : **Acquérir** le contenu d'une `'TABLE'` par son indice
`ecctab` : **Ecrire** un indice et son contenu dans une `'TABLE'`

MANIPULATION des OBJETS

`quetyp` : **Type** du prochain objet dans la pile **GIBIANE** sans le lire
`quenom` : **Nom** du dernier objet lu
`cquoi` : **Pointeur** d'un objet dont on connait le nom
`replis` : **Tous les noms** d'un type d'objet dans un SEGMENT (`CHARACTER*(8)`)



Appliquer une FONCTION à certains OBJETS

Nativement parallèle (Rien a faire pour le développeur c'est implémenté)
A prévoir : Travailler en directive (Utile en interne opérateur)
23 fonctions disponibles (`+ - / * ** COS SIN TAN` etc. : voir `optabj.eso`)
4 objets prévus

`oplre1` : Appliquer à un objet `'LISTREEL'`
`opevo1` : Appliquer à un objet `'EVOLUTION'`
`opchp1` : Appliquer à un objet `'CHPOINT'`
`opche1` : Appliquer à un objet `'MCHAML'`

Ajouter un élément dans un SEGMENT EXTENSIBLE

`ajou` : Ajoute un élément (**`INTEGER`**) s'il n'y est pas déjà
`ajoun2` : Ajoute un élément (**`INTEGER`**) s'il n'y est pas déjà et renvoie sa position

Trier un TABLEAU

Trier par ordre **croissant**

```
triflo, trifla, trient
```

Trier par **ordre choisi** (**RECURSIVE SUBROUTINE**)

```
ordm01 (R8), ordm02 (I), ordm03 (R8 + Perm.), ordm04 (I + Perm.)
```

Trier par **ordre choisi** des **valeurs absolues** (**RECURSIVE SUBROUTINE**)

```
ordm11 (R8), ordm12 (I), ordm13 (R8 + Perm.), ordm14 (I + Perm.)
```

Position d'un élément dans un TABLEAU

Place d'une **chaine de caractère** :

```
place   CARACTER*(*)
place5  CARACTER*(4)
plamo8  CARACTER*(8)
```

Place d'un **entier** : `place2  INTEGER`

Place d'un **flottant** : `place3  REEL*8`

Autres SUBROUTINES utiles

`vonmis` : Fonction qui calcule la contrainte équivalente de VonMises à partir du tenseur des contraintes directement (pas du déviateur)

`verree` : Subroutine qui renvoie un logique « **VRAI** » si le nombre testé est réel et « **FAUX** » dans le cas contraire (**Nan** ou **Inf**). Utile pour détecter de fausses convergences ou des divergences lors de l'intégration.

`trachp` : Convertit un **`'CHPOINT'`** en SEGMENT de travail **`MTRAV`**
`crechp` : Convertit un SEGMENT de travail **`MTRAV`** en **`'CHPOINT'`** (Chapeau a écrire)