Procédure TEST_FICHIER

1 But

Cette macro-commande permet de tester la non régression de fichiers produits par les commandes Code_Aster, principalement IMPR_RESU, mais on peut l'utiliser sur n'importe quel fichier texte.

Pour tester un fichier, on extrait la liste des nombres réels et entiers présents dans le fichier, le cardinal de cette liste, et l'ensemble du texte restant (une fois les nombres extraits).

Pour les nombres, on vérifie la non régression de la somme, de la somme absolue ou du min/max des valeurs à une précision près.
Pour tester le texte du fichier, on utilise le module Python md5 qui fournit une « signature » du texte (appelé md5sum).

Retourne OK si le fichier est identique au fichier qui a servi de référence, NOOK sinon.

Cette commande est utilisée essentiellement par les cas tests pour valider la non-régression des résultats.

2 Syntaxe

```
TEST_FICHIER (
      ♦  FICHIER   =  fich,                                    [Kn]
      ◊  EXPR_IGNORE = regexp,                                 [l_Kn]
         TYPE_TEST =  / 'SOMM',                                [DEFAUT]
                      / 'SOMM_ABS',
                      / 'MAXI',
                      / 'MINI',
                      / 'MAXI_ABS',
                      / 'MINI_ABS',
      ♦  NB_VALE =  nbval ,                                    [I]

      # Définition de la valeur de référence :
      # voir TEST_RESU [u4.92.01]

      ◊  INFO =    /  1 ,                                      [DEFAUT]
                   /  2 ,

           )
```

3 Généralités

Cette procédure permet de tester la non régression de fichiers produits par les commandes Aster, principalement IMPR_RESU, mais on peut l'utiliser sur n'importe quel fichier texte.

Pour tester un fichier, on extrait la liste des nombres réels et la liste des entiers présents dans le fichier, le cardinal de ces listes, et l'ensemble du texte restant (une fois les nombres extraits).

Pour les nombres, on vérifie la non régression de la somme, de la somme absolue ou du min/max des valeurs à une précision près. Les nombres réels et entiers sont traités séparément.

On vérifie également (et c'est le seul test obligatoire) le cardinal de la liste des nombres réels identifiés dans le fichier.
Pour tester le texte du fichier, on utilise le module Python md5 qui fournit une « signature » du texte (appelé md5sum). Ainsi, toute variation d'un texte (titre, nom d'une composante) par rapport à un fichier de référence entraînera un changement de la signature du fichier.

Remarque 1 :

> Le fichier doit être fermé pour que la valeur soit fiable (avec par exemple DEFI_FICHIER, ACTION = 'LIBERER').

Remarque 2 :

> La commande ne donne pas d'information pertinente sur des fichiers binaires.

La fonction test_file peut être appelée hors de *Code_Aster* pour obtenir la valeur du md5sum d'un fichier :

```
iret, sum = test_file( filename='nom_fichier',
                       type_test='SOMM',
                       verbose=True)
```

Remarque 3

> *Pour les nombres entiers, la valeur testée sur les entiers est tronquée au plus grand entier qui peut être représenté en 32 bits (environ 2.e9).*

4 Opérandes

4.1 Opérande FICHIER

♦ FICHIER = fich,

On fournit ici le nom du fichier à analyser avec son chemin d'accès en relatif par rapport au répertoire d'exécution ou en absolu. Exemples : 'fort.37' ou './REPE_OUT/resultats.pos'.

4.2 Opérande EXPR_IGNORE

◊ EXPR_IGNORE = regexp

Les lignes du fichier satisfaisant les expressions régulières données derrière ce mot-clé seront ignorées dans l'analyse du fichier.

Exemple :*

```
EXPR_IGNORE = (  'DATE=[0-9]{2}/[0-9]{2}/[0-9]{4}',
                 '^VERSION'    )
```

Ici les lignes contenant DATE=jj/mm/aaaa où j, m, a sont des chiffres **ou** commençant par VERSION sont ignorées.

4.3 Opérande TYPE_TEST

Type du test fait sur les valeurs réelles et entières du fichier. Les valeurs possibles sont :
- SOMM : somme des valeurs
- SOMM_ABS : somme des valeurs absolues
- MAXI : valeur maximale
- MINI : valeur minimale
- MAXI_ABS : maximum des valeurs absolues
- MINI_ABS : minimum des valeurs absolues

4.4 Opérandes NB_VALE et NB_VALE_I

NB_VALE est le nombre de valeurs réelles attendues dans le fichier. Il s'agit du seul test obligatoire.
NB_VALE_I est le nombre de valeurs entières attendues dans le fichier.

4.5 Mots-clés communs aux commandes TEST_XXX

La définition des valeurs de non régression et de référence, ainsi que des tolérances admissibles, les critère de comparaison est détaillée dans la documentation [u4.92.01] de la commande TEST_RESU.

Les spécificités de TEST_FICHIER sont :
- pas de valeurs complexes,
- pas de test en valeur absolue (sauf avec TYPE_TEST),
- VALE_CALC, VALE_CALC_I et VALE_CALC_K peuvent être fournis ensemble,
- pas de tolérance sur la valeur du paramètre (TOLE_MACHINE et CRITERE ne prenne qu'une seule valeur).

- VALE_CALC fournit la valeur de référence attendue pour le test sur les valeurs réelles (la somme ou maximum ou...).
- VALE_CALC_I fournit la valeur de référence attendue pour le test sur les valeurs entières (la somme ou maximum ou...).
- VALE_CALC_K fournit la signature md5 du texte du fichier après que les toutes les valeurs réelles et entières ont été retirées. Pour obtenir la valeur sur le fichier de référence, il suffit de lancer la commande TEST_FICHIER et de relever la valeur affichée (il s'agit d'une suite de 32 caractères hexadécimaux).

4.6 Opérande INFO

◊ INFO = inf

Précise le détail des informations imprimées dans le fichier message.
Si inf=1, on a le résumé suivant :

(extrait de ssls108a)

```
Nom du fichier   : ./REPE_OUT/DEP12.pos

                      Calculé                              Référence
Entiers :
Nombre de valeurs : 176                                      176
Somme des valeurs : 878                                      878
Réels :
Nombre de valeurs : 10375                                    10375
Somme des valeurs :  1.5553683808e+04                     1.5553683808e+04
Somme de contrôle : e5050b2a3517728c4cc0e23af2b16ba5  not_tested
```

Si inf=2, on a la liste des toutes valeurs relevées dans le fichier, lu par blloc (5897 valeurs réelles et 118 valeurs entières dans le 3ème bloc de cet exemple) :

```
Nombres réels : 5897
['2.1774365E-003', '2.4554530E-003', '2.4552944E-003', '7.0000000E+000', ...]
Nombres entiers : 118
['1', '0', '0', '0', '0', '0', '0', '144', '0', '0', '0', '0', '0', '0', ...]
```

ainsi que l'ensemble du texte restant une fois tous les nombres extraits :

```
Texte :
$EndView$ViewDEFZ_DRX
```