Lecture des données par directives

L'introduction des données dans une application de MÉLINA se fait à l'aide d'un vocabulaire de *mots-clés* restreint et de *mnémoniques* ou *noms propres* définis par l'utilisateur ou le développeur de l'application. Certains *mots-clés* ou *directives générales* déclenchent la prise en compte d'une série de commandes ou *sous-directives*. Les directives générales sont les suivantes :

• Directive ALLOCATION [DYNAMIQUE]

qui permet de modifier les valeurs par défaut d'initialisation de la librairie d'allocation dynamique allodyn.

• Directive STRUCTURE

qui permet de modifier les tailles initiales allouées par défaut aux structures de données.

• Directive GEOMETRIE / MAILLAGE ou MESH

qui déclenche la lecture du fichier de maillage.

• Directive INCONNUE ou UNKNOWN

qui déclenche la lecture du nom des inconnues du problème et de leurs caractéristiques.

• Directive SYMETRIE / ANTISYMETRIE ou SYMMETRY / SKEWSYMMETRY

qui fixe les (anti)symétries du problème ; ces notions n'ont actuellement d'intérêt que pour les calculs des termes liés aux noyaux de Green : termes de couplage, calcul de représentation intégrale, et pour certains calculs relevant de la physique du problème.

• Directive QUADRATURE

qui permet de modifier le degré de précision de la formule d'intégration numérique par défaut pour les calculs des intégrales sur tous les domaines de calculs ; voir aussi le mot-clé QUADRATURE de la directive suivante.

• Directive CALCUL [SUR LE DOMAINE] ou COMPUTE [OVER/ON DOMAIN]

qui produit les structures de données qui pilotent les calculs sur les domaines définis par le maillage : ces structures contiennent les informations relatives aux domaines de calcul et aux termes (ceux, par exemple, de la formulation variationnelle).

• Directive AUTRES CALCULS ou OTHER COMPUTATION

qui permet d'enrichir les structures de données pour des termes autres que ceux définis à l'aide de la directive CALCUL, et dont le calcul est de la responsabilité de l'utilisateur.

• Directive ASSEMBLAGE ou ASSEMBLY

qui produit les structures d'assemblage des termes. Elles contiennent les données définissant de nouveaux termes résultats d'une combinaison linéaire de termes produits par exploitation des autres directives de calcul.

• Directive DONNEE ou DATA

qui permet de définir les noms et caractéristiques des données et d'affecter une valeur aux données de type constante ou tableau.

• Directive FIN ou END

1) qui met fin à la lecture du fichier de maillage ;
2) qui met fin à la lecture du groupe de *directives générales* INCONNUE, SYMETRIE, CALCUL, AUTRES CALCULS, ASSEMBLAGE ;
3) qui termine la lecture d'une série de données de la directive DONNEE.

1. Format des données

Les données sont fournies sur un fichier formatté à accès séquentiel dont les lignes ne sont exploitées que sur leurs **80 premiers caractères**.
Les **articles** de données d'un fichier de données sont :

- ▷ des **mots**, c'est-à-dire des chaînes de caractères *non encadrées par des apostrophes*, comme par exemple les directives générales, et ne contenant pas de caractère séparateur (voir leur liste ci-dessous). Dans les **mots** une majuscule et la minuscule correspondante sont interprétées de la même façon ;
- ▷ des **chaînes**, c'est-à-dire des chaînes de caractères *encadrées d'apostrophes*, comme les noms propres (noms de domaines géométriques, de termes, de données, etc.) définis par le développeur de l'application, les noms des intégrands ou des conditions aux limites essentielles, ou encore les valeurs des données constantes chaînes de caractères. Dans les **chaînes** les majuscules et minuscules sont considérées de manière distincte ;
- ▷ des **valeurs scalaires**, entières, réelles ou réelles double précision[(1)] ;
- ▷ des **lignes 'commentaire'**, c'est-à-dire des lignes contenant `!`, `%` ou `*` en colonne 1 : ces lignes sont ignorées lors de la lecture et du décodage des données ;
- ▷ des **commentaires** sur des lignes de données non commentaire, i.e. des ensembles de caractères suivant `!` ou `%` ou encadrées de parenthèses **( )** ou de crochets <> ; leur contenu est ignoré lors de la lecture et du décodage des données.

Les articles de données (mots, chaînes, valeurs scalaires) sont séparés dans les lignes du fichier par des séparateurs qui sont

- le caractère *blanc* et les caractères suivants `# : ; =`
- le caractère rencontré en fin de ligne $\llcorner_F$
- les 'petits mots' DE, DES, DU, EN, ET, LA, LE, LES, PAR, SUR, UN, UNE et leur déclinaison anglaise AND, BY, FROM, OF, ON, ONTO, OVER, THE, TO.

2. Légende des spécifications

Les notations utilisées dans la description des données sont les suivantes :

`[XXX]`	signifie que `XXX` est un mot-clé ou un groupe de mots-clés optionnel.
`<  >`	indique un groupe de directives ou mots-clés.
$\wedge$	indique un choix exclusif entre plusieurs (groupes de) mots-clés ou directives.
$\oplus$	indique une répétition possible à partir du mot-clé qui suit ce symbole.
$\|\|$`I` ou $\|\|I_k$	est une donnée **entière** en format libre, c'est-à-dire une suite de chiffres (limitée à 12) précédée ou non d'un signe (`100, -30`).
$\|\|$`R` ou $\|\|R_k$	est une donnée **réelle** correspondant au format F*x.y* ou E*x.y* où *x* et *y* sont des entiers quelconques (Exemples : `1., 3.14159, -1.3E-06, 1.D-8`), ou encore d'une expression encadrée de `$`, de la forme `$` e_1 `$` ou `$` e_1 *op* e_2 `$` où *op* est l'un des opérateurs `+, -, *, /, **` ou `^` et e_i est soit un nombre entier ou réel au sens ci-dessus, soit le mot `PI`, soit une expression de la forme *f(n)* où *n* est un nombre, soit `PI` et *f* une fonction choisie dans la liste `ACOS, ARCCOS, ARCSIN, ARCTG, ASIN, ATAN, CH, COS, COSH, EXP, LN, LOG, LOG10, SH, SIN, SINH, SQRT, TAN, TANH, TG, TH`. Remarque : Dans ces expressions, seuls les *blancs* sont ignorés. Exemples : `$PI/2$`, `$LOG(2)$`, `$3 + SQRT(PI)$`.
$\|\|$`A` ou $\|\|A_k$	est un **mot** c'est-à-dire une chaîne de caractères de longueur quelconque commençant par une des lettres majuscules ou minuscules de l'alphabet. Dans un mot les caractères minuscules ou majuscules sont interprétées de la même façon, par exemple les mots `reel`, `Reel` et `REEL` ont la même signification. Dans l'exposé qui suit, ces mots apparaîtront systématiquement en majuscules.

(1) Les données complexes seront, on le verra plus loin, fournies comme un couple de valeurs réelles.

`||C` ou `||C`$_k$ est une **chaîne de caractères** placée entre apostrophes et de longueur quelconque, néanmoins limitée à 78 caractères : 80 caractères sur une ligne -2 caractères pour les 2 apostrophes d'encadrement.

`||C*I` est une chaîne de caractères placée entre apostrophes de longueur maximum `I`.

3. Quelques directives utilitaires

Ces directives peuvent être placées à un endroit quelconque du fichier des directives ; elles sont en effet interceptées par la procédure `REDLE` 'interpréteur' des chaînes de caractères de la librairie de lecture et de décodage des données redlec.

• Directive `ECRITURE` ou `WRITE`

Cette directive permet de changer le nom du fichier d'impression principal où sont reportés toutes les macro-opérations en cours d'une exécution ; ce fichier est par défaut le fichier de sortie standard (`stdout : standard output`) du système, de numéro d'unité logique `IMPPAL` (6 en général). La syntaxe est

```
ECRITURE [ SUR ] [ FICHIER ] ||C
```

```
WRITE [ ON / TO / ONTO ] [ FILE ] ||C
```

où `||C` est (étonnant non !) le nom du fichier.

Procédures concernées, librairie redlec

`REDLE` procédure principale d'interprétation des données.
`REDOUT` lecture du nom et ouverture d'un fichier.

• Directive `TITRE` ou `TITLE`

Il est possible d'insérer des titres sur le fichier d'impression principal en utilisant la directive utilitaire `TITRE` (ou `TITLE`) suivi du nombre de lignes du titre puis des lignes contenant le titre :

```
TITRE ||I
                ⋮
     ''||I lignes du titre''
                ⋮
```

Les lignes du titre sont imprimées, au moment de leur lecture, centrées sur les 80 premières colonnes du fichier principal d'impression `IMPPAL`.

Exemple

```
TITLE 2
     Quel mauvais maillage pour la saison
     Les elements sont dechaines
```

Procédures concernées, librairie redlec

`REDLE` procédure principale d'interprétation des données.
`REDTIT` lecture d'un titre et impression sur fichier de sortie.

Remarque

La procédure `PTITRE`, librairie utiliter réalise la même fonction, elle permet d'insérer des titres dans le fichier `IMPPAL` et peut être appelée du programme principal.

• Directive `LECTURE` ou `READ`

Cette directive permet de changer le nom du fichier d'entrée des directives ; le fichier d'entrée par défaut étant le fichier d'entrée standard (`sdtin : standard input`) du système, de numéro d'unité logique `USYSIN` (5 en général), sous la forme :

```
LECTURE [ AVEC ECHO ] [ SUR FICHIER ] ||C
```

```
WRITE [ WITH ECHO ] [ FROM FILE ] ||C
```

où

`AVEC ECHO` indique que toutes les 'lignes' du fichier seront imprimées sur le fichier d'impression secondaire `IMPSDR`.

△ Le fichier d'entrée standard est désigné exclusivement dans le code sous le nom '`stdin`' ou '`STDIN`'. A l'instant initial d'une exécution, c'est ce fichier qui doit contenir la première directive, qui peut être par exemple :

```
LECTURE SUR FICHIER 'mes-directives'
```

Procédures concernées, librairie redlec

`REDLE` procédure principale d'interprétation des données.
`REDUNI` lecture d'un nom et ouverture d'un fichier d'entree.
`REDINI` initialisation de la lecture sur fichier.

• Directive `STOP` ou `ARRET`

Cette directive permet de mettre un point d'arrêt à l'exécution du programme lors de la lecture des directives. Elle est utile lorsque l'on veut seulement vérifier un fichier de directives :

`STOP` ou `ARRET`

Procédure concernée, librairie redlec

`REDLE` procédure principale d'interprétation des données.

1. Initialisation

1.1. Initialisation de l'allocation dynamique

Directive ALLOCATION DYNAMIQUE

Ce paragraphe, relativement technique, peut être omis en première lecture. Il permet dans le cas de 'grosses' applications de MÉLINA d'améliorer les performances de la librairie d'allocation dynamique. Il s'agit de fournir par donnée des informations concernant le nombre initial maximum de tableaux gérés, le nombre et la taille des buffers tampons servant aux transferts entre la mémoire centrale et le système de fichiers à accès direct constituant la mémoire secondaire paginée, ainsi que la taille maximale de ces fichiers. Une commande (mot-clé MOUCHARD) permet d'imprimer sur un fichier, à des fins de *debug*, toutes les opérations effectuées par les procédures d'allocation dynamique, ce fichier peut alors être consulté après l'exécution du programme ; une seconde commande (mot-clé RUMEURS) permet de stocker dans le tableau 'RUMEUR' inclus dans le super-tableau de caractères (AST) l'état des tableaux gérés ; ce tableau peut être affiché au cours de l'exécution par appel à la procédure PRRUME, librairie allodyn.

La directive ALLOCATION DYNAMIQUE qui active cette lecture de données est optionnelle, sa syntaxe est la suivante : (2)

```
ALLOCATION [ DYNAMIQUE ]
           [ QUANTITE [DE] [TABLEAUX] INITIALE ||I ]
           [ TABLEAUX TAMPONS : ⊕  < NOMBRE ||I ∧  LONGUEUR ||I > ]
           [ [SYSTEME] [DE] FICHIERS : [NOMBRE] [D''] ENREGISTREMENTS ||I ]
           [ RUMEURS < ACTIVES ∧  INACTIVES > ]
           [ MOUCHARD < BAVARD ∧ MUET > ]
```

Valeurs par défaut : Elles correspondent à la syntaxe suivante

```
QUANTITE INITIALE 300(*)
TABLEAUX TAMPONS NOMBRE 5(*)
RUMEURS INACTIVES
MOUCHARD MUET
```

(*) Ces valeurs non contractuelles peuvent être modifiées dans le source de la procédure STINIT, librairie allodyn appelée dans la procédure INITIE, module initial.

Les valeurs par défaut correspondant à la longueur des tableaux tampons et à la sous-directive SYSTEME DE FICHIERS dépendent du système d'exploitation.

Procédures concernées, module initial

INITIE Initialisation de l'allocation dynamique et des structures.
INITIA Initialisation (valeurs par défaut) des paramètres.
INIFIC Initialisation des fichiers.
LCALDY Lecture des mots-clés de la directive ALLOCATION DYNAMIQUE.
STINIT Initialisation des super-tableaux.

(2) Le format de présentation des données est totalement libre, pourvu que les articles soient séparés et que les lignes ne dépassent pas 80 caractères ; la présentation indentée des directives faite dans ce document n'a donc d'autre but que la lisibilité.

Exemple

```
ALLOCATION DYNAMIQUE
           QUANTITE INITIALE 300(i)
           TABLEAUX TAMPONS NOMBRE 2 LONGUEUR 5120(ii)
           FICHIERS NOMBRE D''ENREGISTREMENTS 1000(ii)
           RUMEURS INACTIVES(iii) MOUCHARD MUET(iv)
```

(i) Ce nombre maximum initial de tableaux – il s'agit de **tous** les tableaux du code gérés par la librairie d'allocation dynamique, et non pas des seuls tableaux déclarés de l'application – est remis à jour en cours de l'exécution ; s'il est insuffisant il est augmenté, selon la place disponible dans les super-tableaux, de 64%, 32%, 16%, 8%, 4% ou 2%. Cette augmentation provoque néanmoins en général un déplacement des tables de gestions des tableaux dans les super-tableaux, qui peut être fort coûteux. Une estimation du nombre de fichiers gérés peut être obtenue par un appel à la procédure STIMPR, librairie allodyn.

(ii) Sur certains systèmes, la taille d'un fichier est limitée. Le produit de ses 2 nombres définit cette taille, comptée en mots. Un bon choix pour LONGUEUR est un multiple de la taille d'un secteur de disque.

(iii) Le tableau des rumeurs ne peut que grossir. Étant donné qu'il est implanté en mémoire centrale dans le super-tableau de caractères AST, il est préférable de ne l'utiliser que pour tester de *'petites exécutions'*.

(iv) En cas d'activation, le mouchard, c'est-à-dire le fichier de numéro d'unité logique IMPALO, contient une quantité d'informations qui peut être gigantesque, puisque toutes les actions des procédures de l'allocation dynamique y sont reportées. On aura donc intérêt à ne l'activer (BAVARD) qu'en cas de nécessité. Dans la future version de la librairie allodyn, le nombre d'informations reportées pourra être limité à un nombre fixé de messages les plus récents. L'activation ou la désactivation de ce mouchard peut aussi être effectuée en cours de l'exécution du code par appel à la procédure MOUCHA, librairie allodyn dont le seul argument est l'une des chaînes de caractères 'BAVARD' ou 'ON' et 'MUET' ou 'OFF'. Dans le cas d'activation du mouchard par procédure seules les actions des procédures appelées postérieurement à l'activation sont reportées.

1.2. Tailles initiales des structures

Directive STRUCTURE

Les structures de données contenant la description des **objets** (inconnues, domaines, données, termes, constantes, etc.) utilisés dans le code MÉLINA sont initialement créées avec une taille maximale par défaut (voir plus bas). Lorsqu'en cours d'exécution cette taille s'avère insuffisante, son augmentation automatique peut provoquer une importante remise en cause de l'allocation des tableaux en mémoire centrale et ainsi influer sur le temps de calcul. Le contenu de ces structures dépendant pour une large part du problème traité, l'utilisateur peut a priori connaître (une approximation de) leur taille et la communiquer au programme en utilisant la directive optionnelle STRUCTURE dont la syntaxe est la suivante :

```
STRUCTURE [DE] [ DONNEES ]
          ⊕ < INCONNUE ∧ DOMAINE ∧ DONNEE ∧ TERME ∧ CONSTANTE
              < ENTIER[E] ∧ REEL[LE] ∧ COMPLEXE ∧ CARACTERE > >
              NOMBRE ‖I1 [ LONGUEUR ‖I2 ]
```

Le mot-clé

- INCONNUE permet de définir le nombre maximal initial d'inconnues ainsi que le nombre maximal initial de caractères par nom d'inconnues ;
- DOMAINE le nombre maximal initial de domaines géométriques ainsi que le nombre maximal initial de caractères par nom de domaine ;

- DONNEE idem pour les données. Toutes les données qu'elles soient définies par directives ou créées lors du déroulement du programme doivent être compatibilisées.
- TERME idem pour les termes. Tous les termes qu'ils soient définis par directives ou créés lors du déroulement du programme doivent être compatibilisés.

Le mot-clé

- CONSTANTE permet de définir le nombre maximal initial de données constantes, et l'un des mots clés ENTIER, REEL, COMPLEXE ou CARACTERE, le type du tableau des constantes voulu. Pour les types entier, réel et complexe, seul le nombre (maximal initial) de constantes peut être spécifié ; pour le type caractère on peut aussi donner le nombre (maximal initial) de caractères par nom de donnée.

La valeur entière

- $\|I_1$ représente le nombre maximal initial d'items de la structure.
- $\|I_2$ représente le nombre maximal initial de caractères des noms du tableau de noms des objets liés à la structure.

Les valeurs par défaut (définies à l'aide de l'instruction DATA dans la procédure INITIE, module initial) correspondent à la syntaxe

```
STRUCTURE(DE DONNEES)
          INCONNUE             NOMBRE 5   LONGUEUR 6
          DOMAINE              NOMBRE 10  LONGUEUR 6
          DONNEE               NOMBRE 50  LONGUEUR 6
          TERME                NOMBRE 100LONGUEUR 6
          CONSTANTE ENTIERE    NOMBRE 10
          CONSTANTE REELLE     NOMBRE 10
          CONSTANTE COMPLEXE   NOMBRE 10
          CONSTANTE CARACTERE NOMBRE 10 LONGUEUR 8
```

Procédures concernées, module initial

INITIE Initialisation de l'allocation dynamique et des structures.
INITIA Initialisation (valeurs par défaut) des paramètres.
LCSTRU Lecture des valeurs d'initialisation d'une structure.
CREESD Création des tableaux contenant les structures de données.

Structures de données concernées

#OMINC , #NCONU noms et description des inconnues, mot-clé INCONNUE,
#OMDOM , #TERDO noms et description des domaines géométriques, mot-clé DOMAINE,
#OMDON , $DONNE noms et caractéristiques des données, mot-clé DONNEE,
#OMTRM , $SDTRM noms et caractéristiques des termes, mot-clé TERME,
$ECSTE tableau des constantes entières, mot-clés CONSTANTE ENTIERE,
$RCSTE tableau des constantes réelles, mot-clés CONSTANTE REELLE,
$CCSTE tableau des constantes complexes, mot-clés CONSTANTE COMPLEXE,
$ACSTE , $LGCSA tableau des constantes chaînes de caractères et tableau des longueurs de ces chaînes, mot-clés CONSTANTE CARACTERE

2. Maillage

Directive GEOMETRIE ou MAILLAGE

Le code MÉLINA ne contient pas de module de maillage. Le maillage est fourni au code par lecture d'un fichier contenant la définition du maillage selon le format MÉLINA. La lecture du fichier de maillage est déclenchée par la lecture de la directive GEOMETRIE ou MAILLAGE dans le module de lecture lecgeom.
Un tel fichier peut provenir d'un mailleur quelconque mais doit alors être transformé en fichier au format MÉLINA(une interface MÉLINA–MODULEF a été développée, voir plus bas).
Les renseignements contenus dans un fichier de maillage au format MÉLINA, que tout programme de maillage peut fournir sous une forme ou une autre, sont exclusivement géométriques. Ils permettent de définir la dimension d'espace, le nombre des éléments du maillage, les types (géométriques) des éléments, les coordonnées et la numérotation des *points* élément par élément, et la définition des domaines géométriques. Le fichier de maillage peut en outre contenir quelques renseignements supplémentaires définissant par exemple le niveau d'impression du maillage ou les formats selon lesquels sont lues les coordonnées des points et leur numérotation globale.

La syntaxe de la directive GEOMETRIE ou MAILLAGE est la suivante :

```
< GEOMETRIE ∧ MAILLAGE > [ LECTURE [SUR] < UNITE ∧ FICHIER > ||C ]
```

où ||C est le nom du fichier contenant le maillage au format MÉLINA ; par défaut le fichier est celui contenant les directives (qui est lui même par défaut le fichier des entrées standard).

Exemple
```
GEOMETRIE SUR LE FICHIER 'mon-maillage'
```

2.1. Structure d'un fichier de maillage

Le fichier de maillage est constitué de 4 parties :

• **Introduction – Description du format du fichier**

Le format de lecture du fichier est fourni par la donnée des mots-clés suivants :

```
[ FORMAT [DE] LECTURE       [ [DES] COORDONNEES ||C*50 ]
                            [ [DE LA] NUMEROTATION [GLOBALE] ||C*50 ]
                            [ < SANS ∧ AVEC > COMMENTAIRE] ]
[ IMPRESSION [DE] [NIVEAU] ||I ]
```

qui permet de définir le format de lecture des coordonnées des points du maillage, de la numérotation des points du maillage, et d'indiquer si le fichier contient ou non des lignes commentaires (cf. plus bas). Les chaînes ||C*50 doivent être des champs valides au sens de l'instruction FORMAT de Fortran77 ou bien la chaîne '*' pour une lecture en format libre. La commande IMPRESSION, dont la valeur par défaut 0 correspond à *'aucune impression'*, et que l'on retrouvera partout, est utilisée ici pour l'impression (!) sur le fichier d'impression secondaire IMPSDR de renseignements concernant le maillage, renseignements d'autant plus abondants que la valeur ||I suivant le mot-clé NIVEAU est élevée.
Ces sous-directives sont optionnelles, les valeurs par défaut associées à cet ensemble de mots-clés correspondent à la syntaxe suivante :

```
FORMAT DE LECTURE DES COORDONNEES '6E12.4'
       DE LA NUMEROTATION GLOBALE '18I4'
       AVEC COMMENTAIRE
```

Procédures concernées, module lecgeom

LCGMAI Lecture du maillage proprement dit ;
LCFORM Lecture des formats de lecture des coordonnées des points et de la numérotation globale des points.

• **Description globale du maillage**

Il s'agit ici d'une sous-directive obligatoire : elle permet de définir de manière implicite la dimension d'espace par le nombre de variables d'espace ainsi que le nombre d'éléments du maillage. La syntaxe est la suivante[3] :

```
DESCRIPTION GLOBALE [DU] [MAILLAGE]
            [NOM] [DES] VARIABLES[D''']ESPACE : ⊕ ||C
            NOMBRE[D''']ELEMENTS : ||I
```

Exemple 1 En dimension 3, variables cartésiennes

```
DESCRIPTION GLOBALE DU MAILLAGE
            NOMS DES VARIABLES D''ESPACE : 'X' 'Y' 'Z'
            NOMBRE D''ELEMENTS 100
```

Exemple 2 En dimension 2, variables polaires

```
DESCRIPTION GLOBALE DU MAILLAGE
            VARIABLES D''ESPACE : 'RHO' 'THETA'
            NOMBRE D''ELEMENTS : 857
```

Procédures concernées, module lecgeom

LCDESG Lecture de la description globale du maillage : noms des variables d'espace (définition de la dimension d'espace NDIM du COMMON/GLOBAL/ et création du tableau #ARIAB), nombre d'éléments (variable NBTEL du COMMON/GLOBAL/).

Structures de données mises à jour

#ARIAB tableau des noms des variables d'espace.
NDIM dimension d'espace (COMMON/GLOBAL/).
NBTEL nombre d'éléments (COMMON/GLOBAL/).

• **Description élémentaire du maillage**

L'ensemble des éléments du maillage est partitionné en *blocs d'éléments consécutifs de même type géométrique*. La sous-directive BLOC permet de définir cette partition sous la forme[3] :

```
⊕ BLOC [DE] [TYPE] [GEOMETRIQUE] ||A : ||I ELEMENTS
```

où ||A est un mot-clé définissant le type des éléments du bloc et ||I leur nombre.
Par exemple le mot-clé HE02 représente les hexaèdres de Lagrange de type 2 (à 27 nœuds), TE01 les tétraèdres de Lagrange de type 1 (à 4 nœuds), TR01 les triangles de Lagrange de type 1 (à 3 nœuds), etc. La liste complète des éléments reconnus dans MÉLINA se trouve dans la documentation de la librairie ef3d.

⚠ Le mot-clé ELEMENTS est obligatoire, il indique la fin d'un bloc et le dernier mot-clé ELEMENTS du groupe de mots-clés BLOC provoque la lecture des données définissant les éléments (cf. ci-dessous), il ne doit être suivi d'aucun commentaire dans le fichier.

Exemple En dimension 2, un maillage constitué de 10 triangles numérotés de 1 à 10, 5 quadrangles numérotés de 11 à 15 et 15 triangles numérotés de 16 à 30, de type Lagrange d'ordre 2

(3) Le caractère : n'est utilisé ici que pour faire joli ; il s'agit d'un caractère séparateur de données qui peut être remplacé par un blanc.

```
BLOC DE TYPE GEOMETRIQUE TR02 : 10 ELEMENTS
BLOC                     QU02 : 5 ELEMENTS
BLOC                     TR02 : 15 ELEMENTS
```

Cette description est suivi du corps du fichier de maillage qui définit la liste des éléments du maillage. Ce corps est constitué des coordonnées des *points* et de leur numérotation globale. Étant donné le volume que ces informations peuvent représenter, ces données sont lues directement à l'aide de l'ordre de lecture sur fichier séquentiel formatté READ (et donc pas comme des directives interprétées).
Les coordonnées et les numéros globaux des points sont lus élément par élément selon le format par défaut ou celui défini par la directive FORMAT (cf. plus haut). La liste des coordonnées des points d'un élément peut être précédée d'une ligne de commentaire, ainsi que la liste de leurs numéros globaux, voir la sous-directive AVEC ∧ SANS COMMENTAIRE du mot-clé FORMAT de la directive GEOMETRIE et les exemples complets de fichiers de maillage 2D et 3D plus bas. Ce corps n'étant pas lu par directive, aucune ligne commentaire autre que celles prévues par le mot-clé COMMENTAIRE ne doit y apparaître.

Procédures concernées, module lecgeom

LCDESL Lecture de la description des blocs d'éléments du maillage (cf. mot-clé BLOC) et création de la structure #LOKEL ou de son ersatz du COMMON/GLOBAL/ (variables NBBBLK et NU1BLK).

RDELEM Lecture des données définissant un élément.

RDCOOR Lecture des coordonnées des points d'un élément selon le format de lecture par défaut ou le format déclaré par directive, calcul de l'orientation de l'élément et construction de la structure #ORPTL, tableau de niveau 0, contenant les coordonnées des points du maillage distribuée élément par élément.

RDNUGL Lecture de la numérotation globale des points d'un élément et construction de la structure #GNEEL contenant la numérotation globale des points du maillage, distribuées élément par élément. Le niveau (NIVENG) du tableau contenant la structure est le numéro de type de l'interpolation géométrique : 1 pour les éléments de Lagrange de type 1, 2 pour les éléments de Lagrange de type 2, *etc.*

Structures de données mises à jour

#LOKEL Structure décrivant les types des éléments.

#ORPTL Tableau des coordonnées des points des éléments.

#GNEEL Tableau des numéros des points des éléments.

- **Définition des domaines géométriques**

Le maillage complet étant défini par la liste des éléments ci-dessus, la dernière partie du fichier de maillage contient les listes des constituants des différents domaines géométriques, sur lesquels seront définis les calculs, par exemple des termes (intégrales) de la formulation variationnelle.
La définition de chaque domaine débute par le mot-clé DOMAINE :

```
DOMAINE ||C
```

où ||C est la chaîne de caractères représentant le nom du domaine.

La liste des constituants du domaine est défini de la manière suivante :

– pour un domaine volumique en dimension 3 ou surfacique en dimension 2 :

```
⊕ < ELEMENT(S) ∧ E > ⊕ ||I [ / ||J ]
```

où E ||J / ||I désigne tous les éléments numérotés de ||I à ||J.

– pour un domaine de dimension inférieure à la dimension d'espace (domaines de *bord* par exemple) :

```
⊕ < ELEMENT ∧ E > ||I1 < < FACE ∧ F > ||I2
                       ∧ < ARETE ∧ A > ||I2
                       ∧ < POINT ∧ P > ||I2
                       >
```

Les mots-clés ELEMENT et E sont équivalents, et en dimension 2, les mots-clés FACE ou F et ARETE ou A sont équivalents et caractérisent un domaine linéique.

Exemple 1 Domaine Ω volumique en 3D ou surfacique en 2D

```
DOMAINE 'OMEGA'                DOMAINE 'OMEGA'
   ELEMENTS 1 3 5 7 8 9           ELEMENTS 1 3 5 ELEMENTS 7 8 9
```

ou bien de manière équivalente

```
DOMAINE 'OMEGA'                DOMAINE 'OMEGA'
   E 1 E 3 E 5 E 7 E 8 E 9        E 1 3 5 7 ELEMENT 8 9
```

Exemple 2 Domaine Γ surfacique en 3D

```
DOMAINE 'GAMMA' ELEMENT 1 FACE 3 ELEMENT 2 FACE 2 ELEMENT 7 FACE 2
```

ou encore de manière équivalente

```
DOMAINE 'GAMMA'
    E 1 F 3 E 2 F 2 E 7 F 2
```

Procédures concernées, module lecgeom

LCNOMD Lecture des noms des domaines géométriques (construction du tableau des noms de domaine #OMDOM).

LCDOMA Lecture des constituants d'un domaine géométrique (construction d'une structure #ISTEL, dont le niveau est le numéro du domaine géométrique, i.e. le rang de son nom dans le tableau des noms de domaine #OMDOM).

LCELDO Cas d'un domaine *volumique*, i.e. d'un domaine dont les constituants sont des éléments.

LCFADO Cas d'un domaine de bord, i.e. d'un domaine dont les constituants sont les faces, arêtes ou points d'un élément.

Structures de données mises à jour

#OMDOM Noms des domaines géométriques.

#ISTEL Structures décrivant les domaines géométriques.

#TERDO Structure décrivant les calculs sur les domaines.

- **Fin du fichier de maillage**

Le fichier contenant les données du maillage se termine obligatoirement par le mot-clé :

```
FIN
```

qui provoque un retour à la lecture sur le fichier de données des directives.

2.2. Un exemple de fichier de maillage 2D

Nous donnons ci-dessous le fichier complet d'un maillage P_1 de la région bidimensionnelle, constituée de deux couronnes circulaires concentriques adjacentes Ω_1 et Ω_2 (*figure 1*). La frontière commune aux deux domaines sera notée Γ_1 ou Γ_2 selon qu'elle est considérée comme frontière de Ω_1 ou de Ω_2. Ces deux définitions, qui recouvrent toutes deux la ligne brisée dont les sommets sont les points `5,6,7,8,9` de la figure 2, se retrouvent dans le fichier de maillage, les domaines '`Gamma1`' et '`Gamma2`' étant définis par l'union d'arêtes d'éléments différents.

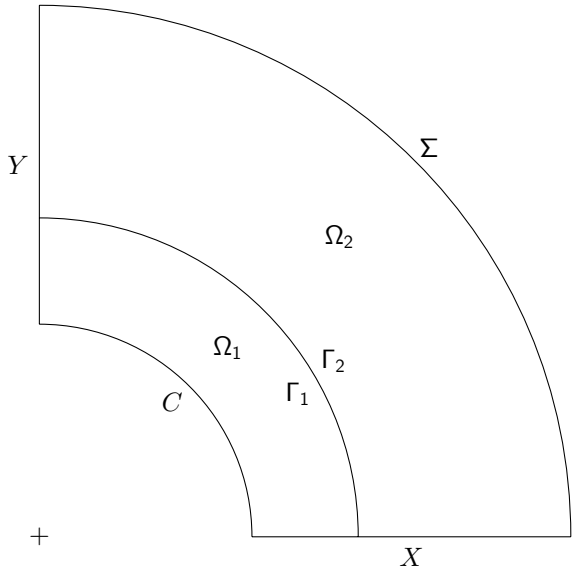

Fig. 1 – La géométrie

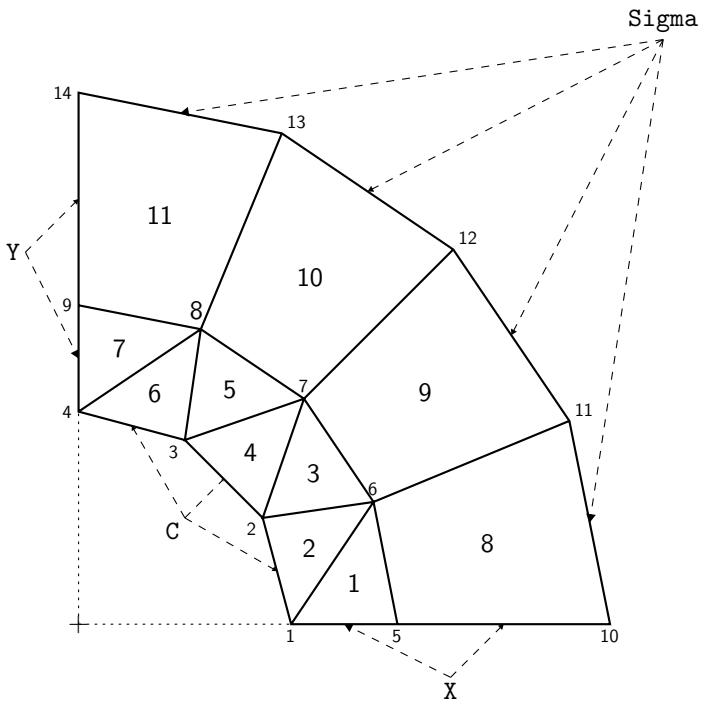

Fig. 2 – Le maillage et les domaines géométriques

Le fichier de maillage au format 'MÉLINA' correspondant

```
 TITRE 2
 Quart de couronne circulaire maille en 2 couronnes concentriques
 contenant respectivement 7 triangles et 4 quadrangles
*
 FORMAT DE LECTURE DES COORDONNEES '8F7.4'
                   DE LA NUMEROTATION GLOBALE '4I3'
                   SANS COMMENTAIRE
*
 DESCRIPTION GLOBALE DU MAILLAGE
    VARIABLES D''ESPACE    'X'  'Y'
    NOMBRE D''ELEMENTS     11
*
***                 Definition des elements du maillage
*
 BLOC DE TYPE GEOMETRIQUE TR01 : 7 ELEMENTS
 BLOC DE TYPE GEOMETRIQUE QU01 : 4 ELEMENTS
 1.5000 0.0000 1.3858 0.5740 1.0000 0.0000
  5  6  1
 0.8660 0.5000 1.0000 0.0000 1.3858 0.5740
  2  1  6
 1.3858 0.5740 1.0607 1.0607 0.8660 0.5000
  6  7  2
 0.5000 0.8660 0.8660 0.5000 1.0607 1.0607
  3  2  7
 1.0607 1.0607 0.5740 1.3858 0.0000 1.0000
  7  8  4
 0.0000 1.0000 0.5000 0.8660 0.5740 1.3858
  4  3  8
 0.5740 1.3858 0.0000 1.5000 0.0000 1.0000
  8  9  4
 1.3858 0.5740 1.5000 0.0000 2.2500 0.0000 2.0787 0.8610
  6  5 10 11
 1.0607 1.0607 1.3858 0.5740 2.0787 0.8610 1.5910 1.5910
  7  6 11 12
 0.5740 1.3858 1.0607 1.0607 1.5910 1.5910 0.8610 2.0787
  8  7 12 13
 0.0000 1.5000 0.5740 1.3858 0.8610 2.0787 0.0000 2.2500
  9  8 13 14
*
***                 Definition des domaines geometriques
*
DOMAINE 'Omega1' (Elements triangulaires)
ELEMENTS 1 / 7
*
DOMAINE 'Omega2' (Elements quadrangulaires)
ELEMENTS 8 / 11
*
DOMAINE 'C' (Cercle interieur)
ELEMENT 2 ARETE 1 ELEMENT 4 ARETE 1 ELEMENT 6 ARETE 1
*
DOMAINE 'Sigma' (Cercle exterieur)
ELEMENT 8 ARETE 3 E 9 A 3 E 10 A 3 E 11 A 3
*
DOMAINE 'X' (Axe Ox)
E 1 A 3 E 8 A 2
*
DOMAINE 'Y' (Axe Oy)
E 7 A 2 E 11 A 4
*
DOMAINE 'Gamma1' (Frontiere intermediaire vue de Omega1)
E 1 A 1 E 3 A 1 E 5 A 1 E 7 A 1
*
DOMAINE 'Gamma2' (Frontiere intermediaire vue de Omega2)
E 8 A 1 E 9 A 1 E 10 A 1 E 11 A 1
FIN (du fichier de maillage)
```

2.3. Un exemple de fichier de maillage 3D

Il correspond au maillage d'un quart de coque sphérique

$$\{1.0 \leq x^2 + y^2 + z^2 \leq 1.75;\ y \geq 0;\ z \leq 0\}$$

à l'aide de 8 éléments prismatiques d'ordre 2, chaque huitième de sphère étant maillés à l'aide de 4 triangles 'sphériques' P_2 'presqu'égaux'.

```
 TITRE 1
 Quart de Coque Spherique : 8 Elements Prismatiques d'ordre 2, 75 points
*
* IMPRESSION DE NIVEAU 1
 FORMAT DE LECTURE DES COORDONNEES '6E12.5'
                   DE LA NUMEROTATION GLOBALE '18I4'
                   AVEC COMMENTAIRE
*
 DESCRIPTION GLOBALE DU MAILLAGE
    VARIABLES D'' ESPACE    'X'  'Y'  'Z'
    NOMBRE D'' ELEMENTS     8
*
***                  Definition des elements du maillage
*
 BLOC DE TYPE GEOMETRIQUE PR02 : 8 ELEMENTS
   Element 1 : Coordonnees des points (Ligne commentaire ignoree)
  .10000+001  .00000+000  .00000+000  .70711+000  .00000+000 -.70711+000
  .70711+000  .70711+000  .00000+000  .17500+001  .00000+000  .00000+000
  .12374+001  .00000+000 -.12374+001  .12374+001  .12374+001  .00000+000
  .92388+000  .00000+000 -.38268+000  .70711+000  .50000+000 -.50000+000
  .92388+000  .38268+000  .00000+000  .16168+001  .00000+000 -.66970+000
  .12374+001  .87500+000 -.87500+000  .16168+001  .66970+000  .00000+000
  .13750+001  .00000+000  .00000+000  .97227+000  .00000+000 -.97227+000
  .97227+000  .97227+000  .00000+000  .12220+001  .00000+000 -.50616+000
  .97227+000  .58682+000 -.58682+000  .12220+001  .50616+000  .00000+000
   Element 1 : Numerotation globale des points (Ligne ignoree)
   1   4   6  51  54  56   2   5   3  52  55  53  26  29  31  27  30  28
   Element 2 : Coordonnees des points (Ligne ignoree)
  .70711+000  .00000+000 -.70711+000  .00000+000  .00000+000 -.10000+001
  .00000+000  .70711+000 -.70711+000  .12374+001  .00000+000 -.12374+001
  .00000+000  .00000+000 -.17500+001  .00000+000  .12374+001 -.12374+001
  .38268+000  .00000+000 -.92388+000  .00000+000  .38268+000 -.92388+000
  .38268+000  .46194+000 -.80010+000  .66970+000  .00000+000 -.16168+001
  .00000+000  .66970+000 -.16168+001  .66970+000  .80839+000 -.14002+001
  .97227+000  .00000+000 -.97227+000  .00000+000  .00000+000 -.13750+001
  .00000+000  .97227+000 -.97227+000  .50616+000  .00000+000 -.12220+001
  .00000+000  .50616+000 -.12220+001  .50616+000  .56065+000 -.10362+001
   Element 2 : Numerotation globale des points (Ligne ignoree)
   4  11  13  54  61  63   7  12   8  57  62  58  29  36  38  32  37  33
   Element 3 : Coordonnees des points (Ligne ignoree)
  .00000+000  .70711+000 -.70711+000  .70711+000  .70711+000  .00000+000
  .70711+000  .00000+000 -.70711+000  .00000+000  .12374+001 -.12374+001
  .12374+001  .12374+001  .00000+000  .12374+001  .00000+000 -.12374+001
  .38268+000  .80010+000 -.46194+000  .70711+000  .50000+000 -.50000+000
  .38268+000  .46194+000 -.80010+000  .66970+000  .14002+001 -.80839+000
  .12374+001  .87500+000 -.87500+000  .66970+000  .80839+000 -.14002+001
  .00000+000  .97227+000 -.97227+000  .97227+000  .97227+000  .00000+000
  .97227+000  .00000+000 -.97227+000  .50616+000  .10362+001 -.56065+000
  .97227+000  .58682+000 -.58682+000  .50616+000  .56065+000 -.10362+001
   Element 3 : Numerotation globale des points (Ligne ignoree)
  13   6   4  63  56  54   9   5   8  59  55  58  38  31  29  34  30  33
   Element 4 : Coordonnees des points (Ligne ignoree)
  .70711+000  .70711+000  .00000+000  .00000+000  .70711+000 -.70711+000
  .00000+000  .10000+001  .00000+000  .12374+001  .12374+001  .00000+000
  .00000+000  .12374+001 -.12374+001  .00000+000  .17500+001  .00000+000
  .38268+000  .80010+000 -.46194+000  .00000+000  .92388+000 -.38268+000
  .38268+000  .92388+000  .00000+000  .66970+000  .14002+001 -.80839+000
  .00000+000  .16168+001 -.66970+000  .66970+000  .16168+001  .00000+000
  .97227+000  .97227+000  .00000+000  .00000+000  .97227+000 -.97227+000
  .00000+000  .13750+001  .00000+000  .50616+000  .10362+001 -.56065+000
  .00000+000  .12220+001 -.50616+000  .50616+000  .12220+001  .00000+000
   Element 4 : Numerotation globale des points (Ligne ignoree)
   6  13  15  56  63  65   9  14  10  59  64  60  31  38  40  34  39  35
```

```
  Element 5 : Coordonnees des points (Ligne ignoree)
-.70711+000  .00000+000 -.70711+000  .00000+000  .70711+000 -.70711+000
 .00000+000  .00000+000 -.10000+001 -.12374+001  .00000+000 -.12374+001
 .00000+000  .12374+001 -.12374+001  .00000+000  .00000+000 -.17500+001
-.38268+000  .46194+000 -.80010+000  .00000+000  .38268+000 -.92388+000
-.38268+000  .00000+000 -.92388+000 -.66970+000  .80839+000 -.14002+001
 .00000+000  .66970+000 -.16168+001 -.66970+000  .00000+000 -.16168+001
-.97227+000  .00000+000 -.97227+000  .00000+000  .97227+000 -.97227+000
 .00000+000  .00000+000 -.13750+001 -.50616+000  .56065+000 -.10362+001
 .00000+000  .50616+000 -.12220+001 -.50616+000  .00000+000 -.12220+001
  Element 5 : Numerotation globale des points (Ligne ignoree)
 20  13  11  70  63  61  17  12  16  67  62  66  45  38  36  42  37  41
  Element 6 : Coordonnees des points (Ligne ignoree)
 .00000+000  .70711+000 -.70711+000 -.70711+000  .00000+000 -.70711+000
-.70711+000  .70711+000  .00000+000  .00000+000  .12374+001 -.12374+001
-.12374+001  .00000+000 -.12374+001 -.12374+001  .12374+001  .00000+000
-.38268+000  .46194+000 -.80010+000 -.70711+000  .50000+000 -.50000+000
-.38268+000  .80010+000 -.46194+000 -.66970+000  .80839+000 -.14002+001
-.12374+001  .87500+000 -.87500+000 -.66970+000  .14002+001 -.80839+000
 .00000+000  .97227+000 -.97227+000 -.97227+000  .00000+000 -.97227+000
-.97227+000  .97227+000  .00000+000 -.50616+000  .56065+000 -.10362+001
-.97227+000  .58682+000 -.58682+000 -.50616+000  .10362+001 -.56065+000
  Element 6 : Numerotation globale des points (Ligne ignoree)
 13  20  22  63  70  72  17  21  18  67  71  68  38  45  47  42  46  43
  Element 7 : Coordonnees des points (Ligne ignoree)
-.70711+000  .70711+000  .00000+000  .00000+000  .10000+001  .00000+000
 .00000+000  .70711+000 -.70711+000 -.12374+001  .12374+001  .00000+000
 .00000+000  .17500+001  .00000+000  .00000+000  .12374+001 -.12374+001
-.38268+000  .92388+000  .00000+000  .00000+000  .92388+000 -.38268+000
-.38268+000  .80010+000 -.46194+000 -.66970+000  .16168+001  .00000+000
 .00000+000  .16168+001 -.66970+000 -.66970+000  .14002+001 -.80839+000
-.97227+000  .97227+000  .00000+000  .00000+000  .13750+001  .00000+000
 .00000+000  .97227+000 -.97227+000 -.50616+000  .12220+001  .00000+000
 .00000+000  .12220+001 -.50616+000 -.50616+000  .10362+001 -.56065+000
  Element 7 : Numerotation globale des points (Ligne ignoree)
 22  15  13  72  65  63  19  14  18  69  64  68  47  40  38  44  39  43
  Element 8 : Coordonnees des points (Ligne ignoree)
-.10000+001  .00000+000  .00000+000 -.70711+000  .70711+000  .00000+000
-.70711+000  .00000+000 -.70711+000 -.17500+001  .00000+000  .00000+000
-.12374+001  .12374+001  .00000+000 -.12374+001  .00000+000 -.12374+001
-.92388+000  .38268+000  .00000+000 -.70711+000  .50000+000 -.50000+000
-.92388+000  .00000+000 -.38268+000 -.16168+001  .66970+000  .00000+000
-.12374+001  .87500+000 -.87500+000 -.16168+001  .00000+000 -.66970+000
-.13750+001  .00000+000  .00000+000 -.97227+000  .97227+000  .00000+000
-.97227+000  .00000+000 -.97227+000 -.12220+001  .50616+000  .00000+000
-.97227+000  .58682+000 -.58682+000 -.12220+001  .00000+000 -.50616+000
  Element 9 : Numerotation globale des points (Ligne ignoree)
 25  22  20  75  72  70  24  21  23  74  71  73  50  47  45  49  46  48
*
***                 Definition des domaines geometriques
*
DOMAINE 'OMEGA'
E 1 / 8
*
DOMAINE 'GAMMA'
E 1 F 1 E 2 F 1 E 3 F 1 E 4 F 1 E 5 F 1 E 6 F 1 E 7 F 1 E 8 F 1
*
DOMAINE 'SIGMA'
E 1 F 5 E 2 F 5 E 3 F 5 E 4 F 5 E 5 F 5 E 6 F 5 E 7 F 5 E 8 F 5
*
DOMAINE 'XOY'
E 1 F 4 E 4 F 4 E 7 F 2 E 8 F 2
*
DOMAINE 'XOZ'
E 1 F 2 E 2 F 2 E 5 F 4 E 8 F 4
FIN
```

Noter que l'on utilise dans ce deuxième exemple, la version `'AVEC COMMENTAIRE'` de lecture des données des éléments ; les lignes du type :

`Element x : Coordonnees des points`

`Element y : Numerotation globale des points`

ne sont donc que des lignes de commentaires et sont ignorées lors de la lecture du fichier : elles sont lues à l'aide d'un `READ(UNIT,*)` sans liste de variables.

2.4. MOME : Interface Modulef–MÉLINA

Une interface entre les structures de données de maillage issues de MODULEF et un fichier de maillage au format MÉLINA a été développée. Elle permet d'utiliser les processeurs graphiques de MODULEF. Il est ainsi possible de transformer une structure NOPO ou des structures MAIL et COOR en un fichier de maillage au format MÉLINA. Il est aussi possible de transformer un fichier de maillage MÉLINA en structure NOPO.
De même un fichier de résultats issu d'une exécution de MÉLINA à l'aide des procédures `OPTERM`, `ECTERM`, `FETERM` peut être transformé en une structure de donnée B de MODULEF.
Ces transformations sont réalisées à l'aide du programme MOME qui utilise les librairies NOPO, UTSD, UTII, UTIL_XXX de MODULEF et les librairies initial, lecgeom, sdexplo, alodyn, redlec, ef3d, utiliter, util_xxx de MÉLINA et fait cohabiter les super-tableaux et les procédures de gestion dynamique de la mémoire des deux codes.

3. Déclaration des inconnues du problème

Directive INCONNUE

La notion d'inconnue dans MÉLINA est comprise au sens du problème variationnel ou de sa formulation discrétisée ; elle représente à la fois le nom d'une (fonction) inconnue et l'espace fonctionnel à laquelle elle appartient (compte non tenu d'éventuelles conditions essentielles). Dans la terminologie MÉLINA, à chaque 'terme' matriciel est affecté un couple d'inconnues, dites respectivement *inconnue en colonne* (la 'vraie' inconnue) et *inconnue en ligne* (représentant une quelconque des fonctions-test). Dans le cas des termes vectoriels (second membre d'un système linéaire), une seule inconnue est affectée au terme (en ligne ou colonne selon qu'il s'agisse d'un terme uni-inconnue en ligne ou colonne). Nous illustrerons les données par directives à l'aide d'exemples issus des problèmes suivants.

Exemple 1 Formulation variationnelle d'un problème de Helmholtz en domaine borné

$$(H)\qquad \left\{\begin{array}{l}\text{Trouver } \varphi\in \mathrm{H}^1(\Omega) \text{ tel que}\\ \displaystyle\int_\Omega \overrightarrow{\mathrm{grad}}\,\varphi\cdot\overrightarrow{\mathrm{grad}}\,\psi\,dx - k^2\int_\Omega \varphi\,\psi\,dx - ik\int_\Sigma \varphi\,\psi\,d\sigma = \int_\Gamma g\psi\,d\gamma\,,\forall\psi\in\mathrm{H}^1(\Omega)\end{array}\right.$$

La seule inconnue du problème est φ puisque la solution et les fonctions-test ψ sont toutes dans le même espace $\mathrm{H}^1(\Omega)$. L'inconnue φ est de type valeurs nodales et scalaire.

△ La plupart des exemples illustrant ce chapitre font directement référence à ce problème ou au problème analogue en domaine non borné. On supposera parfois dans les exemples que g est donnée par $\partial\varphi_w/\partial n_\Gamma$ où φ_w est le potentiel d'une onde incidente.

On notera dans ces exemples

`RIGID` la matrice de rigidité discrète correspondant à $\int_\Omega \overrightarrow{\mathrm{grad}}\,w_j\cdot\overrightarrow{\mathrm{grad}}\,w_i\,dx$
`MASSE` la matrice de masse $\int_\Omega w_j\,w_i\,dx$ à laquelle est associée la constante `'-k^2'`
`GDELTA` la matrice de masse sur Σ : $\int_\Sigma w_j\,w_i\,d\sigma$ à laquelle est associée la constante `'-ik'`
`DPHIN` est le tableau des valeurs nodales g_j aux nœuds de Γ
`PDELTA` est la matrice de masse sur Γ : $\int_\Gamma w_j\,w_i\,d\gamma$
`NEUGAM` le second membre $\int_\Gamma g\,w_i\,d\gamma$ qui pourra être calculé comme le produit de la matrice `PDELTA` et du vecteur `DPHIN` :

$$\sum_{j\in\Gamma} g_j\int_\Gamma w_j\,w_i\,d\gamma$$

Exemple 2 Formulation variationnelle régularisée pour l'équation de Maxwell

$$(M)\qquad \left\{\begin{array}{l}\text{Trouver } \vec{e}\in \mathrm{H}(\mathrm{rot},\Omega)\cap\mathrm{H}(\mathrm{div},\Omega) \text{ tel que } \vec{e}\wedge\vec{n}_{|\Gamma_D}=\vec{g}\\ \displaystyle\int_\Omega \overrightarrow{\mathrm{rot}}\,\vec{e}\cdot\overrightarrow{\mathrm{rot}}\,\vec{\xi}\,dx + s\int_\Omega \mathrm{div}\,\vec{e}\;\mathrm{div}\,\vec{\xi}\,dx = 0\,,\\ \qquad\forall\vec{\xi}\in\mathrm{H}(\mathrm{rot},\Omega)\cap\mathrm{H}(\mathrm{div},\Omega) \text{ tel que } \vec{\xi}\wedge\vec{n}_{|\Gamma_D}=0\end{array}\right.$$

La seule inconnue du problème est $\vec{e}$ puisque la solution et les fonctions-test ξ sont toutes dans le même espace $\mathrm{H}(\mathrm{rot},\Omega)\cap\mathrm{H}(\mathrm{div},\Omega)$. L'inconnue $\vec{e}$ est de type valeurs nodales et vectorielle.

Exemple 3 Formulation variationnelle *'vitesse, pression'* du problème de Stokes

$$(S)\qquad \left\{\begin{array}{l}\text{Trouver } (\vec{u},p)\in\left(\mathrm{H}^1_0(\Omega)\right)^d\times\mathrm{L}^2_0(\Omega) \text{ tel que}\\ \nu\displaystyle\int_\Omega \overrightarrow{\mathrm{grad}}\,\vec{u}\cdot\overrightarrow{\mathrm{grad}}\,\vec{v}\,dx - \int_\Omega p\;\mathrm{div}\,\vec{v}\,dx = \int_\Omega \vec{f}\,\vec{v}\,dx\,,\forall\vec{v}\in\left(\mathrm{H}^1_0(\Omega)\right)^d\\ \displaystyle\int_\Omega \mathrm{div}\,\vec{u}\,q\,dx = 0\,,\forall q\in\mathrm{L}^2_0(\Omega)\end{array}\right.$$

Ici apparaissent les 2 inconnues $\vec{u}$ et p qui correspondent respectivement aux espaces $\left(\mathrm{H}_0^1(\Omega)\right)^d$ et $\mathrm{L}_0^2(\Omega)$. Elles sont toutes les deux de type valeurs nodales ; la première est vectorielle et la seconde scalaire.

Exemple 4 Un problème de bifurcation

Étant donné un réel τ, un ouvert borné connexe Ω et un point $p \in \Omega$

$$(B) \qquad \begin{cases} \text{Trouver } (\varphi, \lambda) \in \mathrm{H}_0^1(\Omega) \times \mathbb{R} \text{ tel que} \\ \displaystyle\int_\Omega \overrightarrow{\mathrm{grad}}\,\varphi \cdot \overrightarrow{\mathrm{grad}}\,\psi\, dx + \lambda \int_\Omega \mathrm{e}^{\varphi}\,\psi\, dx = 0\,, \forall \psi \in \mathrm{H}_0^1(\Omega) \\ \varphi(p) = \tau \end{cases}$$

Il apparaît ici une inconnue scalaire φ valeurs nodales et une inconnue scalaire λ dite supplémentaire.
Une fois linéarisé, dans un processus itératif, le problème s'écrit à l'étape n

$$(B^{(n)}) \qquad \begin{cases} \text{Trouver } (\Phi = \varphi^{(n)}, \Lambda = \lambda^{(n)}) \in \mathrm{H}_0^1(\Omega) \times \mathbb{R} \text{ tel que} \\ \displaystyle\int_\Omega \overrightarrow{\mathrm{grad}}\,\Phi \cdot \overrightarrow{\mathrm{grad}}\,\psi\, dx + \int_\Omega a\,\Phi\,\psi\, dx + \Lambda \int_\Omega b\,\psi\, dx = \int_\Omega (c+a)\,\psi\, dx\,, \forall \psi \in \mathrm{H}_0^1(\Omega) \\ \Phi(p) = T \end{cases}$$

avec $b(x) = \mathrm{e}^{\varphi^{(n-1)}}(x)$, $a(x) = \lambda^{(n-1)} b(x)$, $c(x) = -\Delta\varphi^{(n-1)}(x)$ et $T = \tau - \varphi^{(n-1)}(p)$.

3.1. Inconnues en colonne et en ligne

À un terme matriciel est associé un couple d'inconnues, l'une dite 'en colonne', l'autre 'en ligne'. Cette terminologie se rapporte au système linéaire défini par discrétisation d'une formulation variationnelle, ainsi pour le système

$$\sum_{j=1}^{N} A_{ij} x_j = b_i \quad \text{, pour } i = 1, \ldots, N$$

l'indice j se rapporte à l'inconnue en colonne, l'indice i se rapporte aux lignes.

Rappelons que dans MÉLINA la notion d'inconnue se rapporte seulement à la définition de l'espace d'approximation de la fonction inconnue ou des fonctions-test.

Exemple 1 Pour les termes de la formulation (H) $\displaystyle\int_\Omega \overrightarrow{\mathrm{grad}}\,\varphi \cdot \overrightarrow{\mathrm{grad}}\,\psi\, dx$ et $\displaystyle\int_\Omega \varphi\psi\, dx$, les inconnues φ et ψ (fonction-test) se rapportent tous deux au même espace d'approximation (de $\mathrm{H}^1(\Omega)$). Si le nom de l'inconnue φ est `'PHI'` (comme indiqué ci-dessus), le couple d'inconnues colonne×ligne est pour ces termes (`'PHI'`,`'PHI'`).

Exemple 2 Idem pour la formulation (M)

Exemple 3 Problème de Stokes (S)

L'inconnue $\vec{u}$ est notée `'Vitesse'` et l'inconnue p est notée `'Pression'`. Le terme

- $\displaystyle\int_\Omega \overrightarrow{\mathrm{grad}}\,\vec{u} \cdot \overrightarrow{\mathrm{grad}}\,\vec{v}\, dx$ a pour inconnues en colonne et ligne `'Vitesse'`.
- $\displaystyle\int_\Omega p\ \mathrm{div}\,\vec{v}\, dx$ a pour inconnue en colonne `'Pression'` et en ligne `'Vitesse'`.
- $\displaystyle\int_\Omega \mathrm{div}\,\vec{u}\, q\, dx$ a pour inconnue en colonne `'Vitesse'` et en ligne `'Pression'`.

Exemple 4 Problème de bifurcation ($B^{(n)}$)
L'inconnue Φ est notée 'Phi' et l'inconnue Λ est notée 'Lambda'. Le terme

- $\int_\Omega \overrightarrow{\text{grad}}\,\Phi \cdot \overrightarrow{\text{grad}}\,\psi\, dx$ a pour inconnues en colonne et ligne 'Phi'.
- $\Lambda \int_\Omega b\,\psi\, dx$ a pour inconnue en colonne 'Lambda' et en ligne 'Phi'.

3.2. Déclaration d'une inconnue

La déclaration et la définition des inconnues est effectuée sous la forme

```
⊕ INCONNUE ||C
   < [ VALEURS NODALES]
      [ TYPE < SCALAIRE ∧ VECTORIEL [ COMPOSANTES ||I_c ] > ]
      [ INTERPOLATION
          < ISOPARAMETRIQUE
          ∧ DE LAGRANGE
              < P0 ∧ Q0 ∧ P1 ∧ Q1 ∧ P2 ∧ Q2 ∧ P3 ∧ Q3 ∧ Q4 >
              [SERENDIPITY] [NON CONFORME]
          >
   ∧ SPECTRALE
       NOMBRE DE FONCTIONS PROPRES ||I_f
       NOMBRE DE TERMES PAR FONCTION ||I_t >
   ∧ SUPPLEMENTAIRE
      [ TYPE < SCALAIRE ∧ VECTORIEL [ COMPOSANTES ||I_c ] > ]
   >
```

où

- VALEURS NODALES définit une (fonction) inconnue interpolée sur une base éléments finis (valeur par défaut) ;
- SPECTRALE définit une inconnue spectrale dans la méthode des éléments finis localisés ;
- SUPPLEMENTAIRE définit une inconnue non valeurs nodales autre que SPECTRALE (valeur propre ou multiplicateur de Lagrange par exemple).

et

||C est le nom de l'inconnue ;

||I_c est le nombre de composantes d'une inconnue vectorielle (= NDIM dimension d'espace, par défaut) ;

||I_f est le nombre de fonctions spectrales définissant l'opérateur frontière dans la méthode des éléments finis localisés ;

||I_t est le nombre de termes dans le développement des fonctions spectrales en somme d'exponentielles.

Valeurs par défaut

Elles correspondent pour l'inconnue 'INCO' à la syntaxe :

```
INCONNUE 'INCO' VALEURS NODALES
                TYPE SCALAIRE
                INTERPOLATION ISOPARAMETRIQUE
```

c'est-à-dire que l'inconnue est interpolée selon les mêmes fonctions de base que celles qui définissent la géométrie.

⚠ Il n'existe actuellement aucune autre possibilité de définir les inconnues du problème qu'en utilisant cette directive.

Procédures concernées, module lecdire

`LCDIRE` Procédure principale de lecture des directives
`LCINCO` Lecture des noms et de caractéristiques des inconnues du problème
`LCINTE` Lecture du type d'interpolation d'inconnue de type valeurs nodales

Structures de données mises à jour

`#OMINC` tableaux des noms des inconnues
`#NCONU` tableaux des caractéristiques des inconnues

Exemple 1 Déclaration de l'inconnue scalaire φ dans le problème de Helmholtz (H)

```
INCONNUE 'PHI'
! valeurs nodales, type scalaire et interpolation isoparamétrique
```

Exemple 2 Déclaration de l'inconnue vectorielle $\vec{e}$ dans le problème de Maxwell (M)

```
INCONNUE 'E' DE TYPE VECTORIEL
! interpolation isoparamétrique (par exemple)
```

Exemple 3 Déclaration des inconnues vectorielle $\vec{u}$ et scalaire p dans le problème de Stokes (S)

```
INCONNUE 'Vitesse' DE TYPE VECTORIEL INTERPOLATION DE LAGRANGE P2
INCONNUE 'Pression'INTERPOLATION DE LAGRANGE P1
```

Exemple 4 Déclaration des inconnues φ et λ dans le problème (B) ou $(B^{(n)})$

```
INCONNUE 'Phi'    VALEURS NODALES INTERPOLATION DE LAGRANGE P2
INCONNUE 'Lambda' SUPPLEMENTAIRE DE TYPE SCALAIRE
```

4. Déclaration des symétries du problème

Directives SYMETRIE ou ANTISYMETRIE

Cette directive n'offre d'intérêt que dans le cas où l'on utilise dans l'application les **noyaux de Green**, c'est-à-dire dans le cas où l'on calcule des termes de couplage ou encore une représentation intégrale. Cette directive permet d'informer le module de calcul des noyaux de Green, que l'on n'utilise qu'une partie du domaine physique, et que la solution du problème peut se déduire par symétrie ou antisymétrie sur les parties non maillées du domaine physique. Seules les symétries ou antisymétries par rapport aux plans ou aux axes de coordonnées, par exemple $x = 0$, $y = 0$ ou $z = 0$ en coordonnées cartésiennes, peuvent être déclarées. La syntaxe de cette directive est très simple :

$\oplus$ < SYMETRIE $\wedge$ ANTISYMETRIE> EN ‖C

où ‖C est le nom d'une des variables d'espaces tel qu'il a été défini dans le fichier de maillage. Cette directive indique une (anti)symétrie par rapport au plan ou à l'axe ‖C=0.

Procédures concernées, module lecdire

LCSYME Lecture des (anti)symétries du problème.

Structure de données concernée : #ARIAB, et variable ISYMGR du COMMON/GLOBAL/

5. Choix du degré des formules de quadrature

Directive QUADRATURE

Elle permet de modifier la valeur par défaut du numéro de la formule de quadrature utilisée pour les calculs des intégrales constituant par exemple la formulation variationnelle du problème. La formule de degré choisi dans cette directive sera la formule utilisée pour les intégrales sur tous les domaines de calcul, sauf modification à l'aide du mot-clé QUADRATURE de la directive CALCUL pour une intégrale sur un domaine particulier.
La syntaxe de cette directive est très simple :

```
QUADRATURE
< [DE] GAUSS [DE] DEGRE ||I
/ NODALE < P1 / P2 / Q1 / Q2 >
>
```

où ||I est le degré de la formule de Gauss, et NODALE indique le choix d'une formule construite sur les nœuds d'un élément fini.

Procédure concernée, module lecdire

LCQUAD Lecture des (anti)symétries du problème.

Variable mise à jour : QUADEF du COMMON/GLOBAL/

6. Déclaration des calculs sur les domaines

Directive CALCUL SUR LE DOMAINE

6.1. Déclaration des termes de type Éléments Finis

Il s'agit essentiellement de définir les tableaux qui recevront les matrices et vecteurs correspondant aux intégrales de la formulation variationnelle, et la façon de les calculer. Ces tableaux, associés aux caractéristiques qui permettent de les calculer sont baptisés termes éléments finis.

△ On verra plus loin qu'il est possible de (re)définir ces termes par procédure : voir les procédures MKTERM et RKTERM.

La déclaration d'un terme éléments finis est effectuée par donnée sous la forme :

```
⊕ CALCUL [SUR] [LE] DOMAINE ||C_d
  [ IMPRESSION [DE] [ NIVEAU ||I_md ] ]
  [ QUADRATURE [DE] DEGRE ||I_q ]
  ⊕ TERME < ELEMENTS [FINIS] ASSEMBLE ∧ MATRICE ELEMENTAIRE >
    ||C*6 [ [DE] NIVEAU ||I ]
       [ IMPRESSION [DE] [ NIVEAU ||I_mt ] ]
       [ INCONNUE [DE NOM] ||C_c [ EN COLONNE] ]
       [ INCONNUE [DE NOM] ||C_l [ EN LIGNE] ]
         INTEGRAND||C_i
       [ DONNEE ||C_do
          [< CONSTANTE
              [< ENTIERE ∧ REELLE ∧ COMPLEXE >]
           ∧ TABLEAU [ NIVEAU ||I_do ]
              [< ENTIER ∧ REEL ∧ COMPLEXE >]
           ∧ FONCTION
              [< REELLE ∧ COMPLEXE > ]
              [ DONNEE ASSOCIEE ||C_da
                [< CONSTANTE ∧ TABLEAU [NIVEAU ||I_da] >]
                [< ENTIER[E] ∧ REEL[LE] ∧ COMPLEXE >
           >]
       ]
```

où

||C_d est le nom du domaine géométrique sur lequel les calculs sont demandés ; il a été défini lors de la lecture du fichier de maillage ; la variable réceptrice s'appelle en général NOMDOM[4].

||I_{md} est le niveau d'impression des calculs sur le domaine (par défaut pas d'impression : niveau 0) ; variable réceptrice NIVDOM[4]

||I_q est le numéro du schéma de quadrature pour le calcul des termes sur le domaine (par défaut de degré 5) ; variable réceptrice NUSCHQ[4].

||C*6 est le nom d'un terme correspondant à la matrice ou au vecteur représentant une intégrale discrétisée. **Cette chaîne ne contient que des caractères alphanumériques et son initiale est une lettre** ; variable réceptrice NOMTRM ou NMTERM ou une variante NMTRM1, etc.[4]

||I est son niveau. Le couple (nom, niveau) désigne aussi le tableau géré par l'allocation dynamique qui contiendra ce terme (valeur par défaut 1) ; variable réceptrice NVTERM ou NIVTRM ou une variante NVTERM, NVTRM1[4], etc.

||I_{mt} est le niveau d'impression lors du calcul du terme (par défaut niveau 0 : pas d'impression) ; variable réceptrice NIVIMP[4].

[4] Ces renseignements sont fournis pour faire le lien avec la procédure de déclaration de termes MKTERM.

$\|\mathtt{C}_c$ représente le nom de l'inconnue en colonne pour le terme (cette notion n'a d'intérêt que pour un terme matriciel). Cette inconnue doit avoir été préalablement définie à l'aide de la directive `INCONNUE` ; variable réceptrice `NMINCC`[4].

$\|\mathtt{C}_l$ représente le nom de l'inconnue en ligne pour le terme. Pour un terme matriciel la valeur par défaut pour cette inconnue est le nom de l'inconnue en colonne ; variable réceptrice `NMINCL`[4].

$\|\mathtt{C}_i$ est le nom de l'intégrand correspondant au terme, à choisir dans la liste que l'on trouvera dans l'annexe **Intégrands des termes éléments finis** du présent chapitre.

$\|\mathtt{C}_{do}$ est le nom d'une donnée affectée au terme[5]. Cette donnée peut être

- une constante, comme la constante $-k^2$ affectée au terme $\int_\Omega \varphi\psi\, dx$ ou la constante $-ik$ affectée à $\int_\Sigma \varphi\psi\, d\sigma$ dans l'exemple 1, ou encore la constante ν dans l'exemple 2. La constante interviendra lors de l'assemblage (combinaison linéaire) de ce terme à d'autres, par exemple pour constituer le système linéaire issu de la formulation variationnelle ;
- une fonction, dans le cas d'une intégrale à coefficient variable, comme la fonction g dans le terme de second membre $\int_\Gamma g\psi\, d\gamma$ de l'exemple 1. Une donné fonction intervient lors des calculs des matrices et vecteurs élémentaires, elle est évaluée au point image dans l'élément courant de chaque point du schéma d'intégration numérique. Ces fonctions utilisateurs, qui dépendent de l'application doivent être programmées dans la procédure `FCTRM` à l'intérieur de laquelle les différentes données 'fonctions' peuvent être distinguées dans le corps de la procédure par leur nom de donnée (voir plus bas) ;
- un tableau de valeurs nodales, auquel on peut adjoindre un niveau (valeur par défaut 1). Dans ce cas l'évaluation de l'intégrale est identique à celle que l'on obtient si on se place dans le cas d'une fonction que l'on interpole (selon l'interpolation de l'inconnue en colonne). Le terme de second membre $\int_\Gamma g\psi\, d\gamma$ dans l'exemple 1 peut être calculé sous la forme $\sum_{i\in\Gamma} g_i \int_\Gamma w_i\psi\, d\gamma$ où $\{w_i\}_i$ sont les fonctions de base de l'interpolation en colonne, ψ une fonction de base de l'interpolation en ligne, et les valeurs g_i sont contenues dans le tableau de valeurs nodales associé.

- Le type *de déclaration* d'une telle donnée est soit entier, soit réel (valeur par défaut), soit complexe ; c'est-à-dire que, dans le cas d'une constante ou d'un tableau, la constante ou le tableau est de type entier, réel ou complexe ; dans le cas d'une fonction, que le résultat retourné par la procédure `FCTRM` est réel ou complexe.

⚠ La valeur par défaut, *i.e. si aucun nom de donnée* $\|\mathtt{C}_{do}$ *n'est fourni*, de la donnée affectée à un terme éléments finis est la constante 1.

$\|\mathtt{I}_{do}$ est le niveau de la donnée tableau, lorsque la donnée affectée au terme est de type tableau (valeur par défaut : 1).

$\|\mathtt{C}_{da}$ Dans le cas où la donnée affectée à un terme est une donnée de type fonction, il est possible d'associer à celle-ci, une donnée de type constante ou tableau[6]. Par exemple dans le cas d'un problème de diffraction du type de l'exemple 1, la donnée de Neuman de second membre g peut dépendre de la constante k : pour une onde incidente plane se propageant dans la direction x, on aurait $g = \mathrm{e}^{ikx}$ (ici on triche un peu pour les besoins de l'exposé, on aurait plutôt dans ce cas $g = \partial \mathrm{e}^{ikx}/\partial n_\Gamma$, voir plus bas les termes de type valeurs nodales). Dans le cas d'une donnée de type tableau associée à une fonction, le contenu du tableau est entièrement à la discrétion du développeur de l'application, le code se charge seulement de transmettre son adresse et son type (cf. la procédure utilisateur `FCTRM`).

$\|\mathtt{I}_{da}$ est le niveau de la donnée tableau associée à une donnée de type fonction (valeur par défaut : 1).

(5) On peut aussi affecter une donnée à un terme par procédure, cf. `DOASTR` de la librairie sdexplo.

(6) Il est possible d'associer une donnée à une fonction par procédure, cf. `DOASFC` de la librairie sdexplo.

⚠ ⚠ Si une même donnée est affectée à plusieurs termes, il est interdit de déclarer ses caractéristiques (types de représentation et de déclaration) plus d'une fois. Dans le cas contraire, un message d'erreur est émis. En particulier si une telle donnée est un tableau on ne devra pas redéfinir son niveau.

Exemple 1 Déclaration des termes pour le problème de Helmholtz (H)

`RIGID` $\overset{def}{\approx} \int_\Omega \overrightarrow{\text{grad}}\,\varphi \cdot \overrightarrow{\text{grad}}\,\psi\, dx$, inconnues `PHI`, `PHI`, constante 1 (défaut)

`MASSE` $\overset{def}{\approx} \int_\Omega \varphi\psi\, dx$, inconnues `PHI`, `PHI`, constante affectée $-k^2$

`GDELTA` $\overset{def}{\approx} \int_\Sigma \varphi\psi\, d\sigma$, inconnues `PHI`, `PHI`, constante affectée $-ik$,

`NEUGAM` $\overset{def}{\approx} \int_\Gamma g\psi\, d\gamma$, inconnues `PHI`, donnée affectée fonction g, la constante k étant associée à la donnée fonction g :

```
CALCUL SUR LE DOMAINE 'OMEGA'
       TERME ELEMENTS FINIS 'RIGID'
        INCONNUE 'PHI' (declaree dans la directive INCONNUE)
        INTEGRAND 'GRADGRAD'
       TERME ELEMENTS FINIS 'MASSE' INCONNUE 'PHI' INTEGRAND 'UV'
        DONNEE '-k2' CONSTANTE REELLE
CALCUL SUR LE DOMAINE 'SIGMA'
       TERME ELEMENTS FINIS 'GDELTA' INCONNUE PHI
        INTEGRAND 'UV' DONNEE '-ik' CONSTANTE COMPLEXE
CALCUL SUR LE DOMAINE 'GAMMA'
       TERME ELEMENTS FINIS 'NEUGAM' INCONNUE 'PHI'
        INTEGRAND 'V' DONNEE 'G' FONCTION COMPLEXE
                      DONNEE ASSOCIEE 'k'
```

Exemple 2 Déclaration des termes pour le problème de Stokes (S)

`RIGID` $\overset{def}{\approx} \int_\Omega \overrightarrow{\text{grad}}\,\vec{u} \cdot \overrightarrow{\text{grad}}\,\vec{v}\, dx$, inconnues `'Vitesse'`, `'Vitesse'`, constante affectée ν

`PDIVV` $\overset{def}{\approx} \int_\Omega p\ \text{div}\,\vec{v}\, dx$, inconnues `'Pression'` en colonne, `'Vitesse'` en ligne, constante affectée -1

`DIVUQ` $\overset{def}{\approx} \int_\Omega \text{div}\,\vec{u}\, q\, dx$, inconnues `'Vitesse'` en colonne, `'Pression'` en ligne.

```
CALCUL SUR LE DOMAINE 'OMEGA'
       TERME ELEMENTS FINIS 'RIGID'
        INCONNUE 'Vitesse' INTEGRAND 'GRADGRAD'
        DONNEE 'nu'
       TERME ELEMENTS FINIS 'PDIVV'
        INCONNUE 'Pression' EN COLONNE INCONNUE 'Vitesse' EN LIGNE
        INTEGRAND 'UDIVV' DONNEE '-1'
       TERME ELEMENTS FINIS 'DIVUQ'
        INCONNUE 'Vitesse' EN COLONNE INCONNUE 'Pression' EN LIGNE
        INTEGRAND 'DIVUV'
```

Exemple 3 Déclaration des termes pour le problème de bifurcation linéarisé ($B^{(n+1)}$)

`RIGID` $\overset{def}{\approx} \int_\Omega \overrightarrow{\text{grad}}\,\Phi \cdot \overrightarrow{\text{grad}}\,\psi\, dx$, inconnues `Phi`, `Phi`

`MASSE` $\overset{def}{\approx} \int_\Omega a\,\Phi\psi, dx$, inconnues `Phi`, `Phi`, donnée affectée a tableau

`BPSI` $\overset{def}{\approx} \int_\Omega b\,\psi\, dx$, inconnues `Lambda` en colonne, `Phi` en ligne, donnée associée b tableau

`APSI` $\overset{def}{\approx} \int_\Omega a\,\psi\, dx$, inconnues `Phi`, donnée affectée a

`CPSI` $\overset{def}{\approx} \int_\Omega c\,\psi\, dx$, inconnues `Phi`. Ce terme relève d'un traitement particulier, il sera calculé comme produit matrice×vecteur selon la formule

$$-\int_\Omega \Delta\varphi^{(n)}\,\psi\,dx = \int_\Omega \overrightarrow{\text{grad}}\,\varphi^{(n)}.\,\overrightarrow{\text{grad}}\,\psi\,dx$$

```
CALCUL SUR LE DOMAINE 'OMEGA'
       TERME ELEMENTS FINIS 'RIGID'
        INCONNUE 'Phi' (colonne et ligne) INTEGRAND 'GRADGRAD'
       TERME ELEMENTS FINIS 'MASSE'
        INCONNUE 'Phi' INTEGRAND 'UV'
        DONNEE 'a' TABLEAU DE NIVEAU 1 REEL
       TERME ELEMENTS FINIS 'BPSI'
        INCONNUE 'Lambda' EN COLONNE INCONNUE 'Phi' EN LIGNE
        INTEGRAND 'V' DONNEE 'b' TABLEAU DE NIVEAU 1 REEL
       TERME ELEMENTS FINIS 'APSI'
        INCONNUE 'Phi' INTEGRAND 'V' DONNEE 'a'
```

Les tableaux `'a'` et `'b'` seront définis ici comme termes valeurs nodales (voir § suivant).

Procédures concernées, module lecdire

`LCKDOM` sous-procédure principale de lecture des calculs sur les domaines.

`LCQUAD` lecture du type de la formule de quadrature pour le calcul des termes sur un domaine. Deux types de formule de quadrature différents pour le calcul de termes sur un même domaine géométrique donnent naissance à deux *domaines de calcul* distincts.

`LCTERM` lecture du nom d'un terme à calculer sur un domaine et procédure *principale* de lecture des caractéristiques du terme.

`LCINCT` lecture des noms des inconnues attachées à un terme.

`LCINTG` lecture du nom de l'intégrand correspondant à un terme éléments finis.

`LCDONT` lecture de l'éventuelle donnée affectée à un terme.

`LCCADO` lecture des caractéristiques de la donnée affectée à un terme.

`LCFLOC` lecture des caractéristiques d'un terme éléments finis localisés.

Structures mises à jour

`#TERDO` tableau de description des calculs dur les domaines

`#OMTRM` tableau des noms des termes

`$SDTRM` tableau des attributs des termes

`#OMDON` tableau des noms des données

`$DONNE` tableau des attributs des termes

6.2. Déclaration des termes de type 'Valeurs Nodales'

Il s'agit de définir des vecteurs qui recevront les tableaux contenant des valeurs aux nœuds, par exemple pour une donnée de condition de bord de type Neuman ou Fourier ou bien encore les valeurs nodales d'une solution exacte. La syntaxe de la déclaration est en tous points analogue à celle d'un terme éléments finis :

```
⊕ CALCUL [SUR] [LE] DOMAINE ||Cd
   [ IMPRESSION [DE] [ NIVEAU ||Imd ] ]
   ⊕ TERME VALEURS NODALES ||C*6 [NIVEAU ||It]
        [ [DE] [TYPE] ||Cv ]
        [ < SCALAIRE ∧ VECTORIEL ∧ <TENSORIEL∧ TENSORIEL> > ]
        [IMPRESSION [DE] [NIVEAU ||Imt] ]
         INCONNUE ||Cc
        [ DONNEE ||Cdo
           [< CONSTANTE
               [< ENTIERE ∧ REELLE ∧ COMPLEXE >]
            ∧ TABLEAU [ NIVEAU ||Ido ]
               [< ENTIER ∧ REEL ∧ COMPLEXE >]
            ∧ FONCTION
               [< REELLE ∧ COMPLEXE > ]
               [ DONNEE ASSOCIEE ||Cda
                 [< CONSTANTE ∧ TABLEAU [NIVEAU ||Ida] >]
                 [< ENTIER[E] ∧ REEL[LE] ∧ COMPLEXE >
            >]
        ]
```

où

- $\|C_d$ est le nom du domaine géométrique ;
- $\|I_{md}$ est le niveau d'impression des calculs sur le domaine ;
- $\|C*6$ est le nom d'un terme correspondant au vecteur représentant le terme valeurs nodales ;
- $\|I_t$ est son niveau ;
- $\|C_v$ est le type de calcul de valeurs nodales du terme, à choisir dans la liste des types de termes valeurs nodales que l'on trouvera dans l'annexe **Type des termes 'Valeurs Nodales'** du présent chapitre. Dans le cas des termes de type 'F' ou 'TABF', on doit préciser, lorsque l'inconnue associée $\|C_c$ a plusieurs composantes par nœud, si le terme est SCALAIRE, VECTORIEL ou MATRICIEL aux nœuds.
- $\|I_{mt}$ est le niveau d'impression lors du calcul du terme ;
- $\|C_c$ représente le nom de l'inconnue. Les calculs sont effectués selon l'interpolation de cette inconnue aux nœuds de cette inconnue appartenant au domaine. Cette inconnue doit avoir été préalablement définie à l'aide de la directive INCONNUE ;
- $\|C_{do}$ est le nom d'une donnée affectée au terme, etc. (voir TERME ELEMENTS FINIS, plus haut).

△ Dans la déclaration d'un terme valeurs nodales, l'absence de la directive DONNEE $\|C_{do}$ provoque le calcul d'un **terme nul** ; ceci permet d'initialiser un tableau de valeurs à zéro sur un domaine.

Exemple 1 Calcul d'une solution exacte dépendant d'une constante k

```
CALCUL SUR LE DOMAINE 'OMEGA'
       TERME VALEUR NODALE 'SOLEXA' ('F')
        INCONNUE 'PHI' (declaree dans la directive INCONNUE)
        DONNEE 'SOLEX' FONCTION COMPLEXE
                       DONNEE ASSOCIEE (a SOLEX) 'k'
```

Exemple 2 Calcul de la dérivée normale du potentiel d'une onde incidente

Le terme NEUGAM$\overset{def}{\approx} \int_\Gamma g\psi\, d\gamma$ où $g = -\partial\varphi_w/\partial n_\Gamma$ et $\varphi_w(x,y,z) = \mathrm{e}^{ik(x\cos\theta - y\sin\theta)}$ peut être calculé par la formule $\sum_{i\in\Gamma} g_j \int_\Gamma w_j\psi\, d\gamma$ et nécessite dans ce cas l'évaluation de $-\partial\varphi_w/\partial n_\Gamma$ aux nœuds du domaine Γ :

```
CALCUL SUR LE DOMAINE 'GAMMA'
       TERME VALEUR NODALE 'DPHIN' 'N.F' (ou 'DF/DN')
        INCONNUE 'PHI' (declaree dans la directive INCONNUE)
        DONNEE 'DPHIn' FONCTION COMPLEXE
          DONNEE ASSOCIEE (a DPHIn) 'k&teta' TABLEAU DE NIVEAU 1
```

Le tableau `k&teta` de niveau 1 contiendra les valeurs en cours des données k et θ. L'adresse et le type de ce tableau sont transmises à la procédure FCTRM chargée de calculer le gradient de φ_w.

Exemple 3 Déclarations des termes valeurs nodales pour le problème $B^{(n+1)}$

```
CALCUL SUR LE DOMAINE 'OMEGA'
       TERME VALEUR NODALE 'a'
        INCONNUE 'Phi'
        DONNEE 'ta' TABLEAU REEL
       TERME VALEUR NODALE 'b'
        INCONNUE 'Phi'
        DONNEE 'tb' TABLEAU REEL
```

Le tableau `tb` de niveau 1 contiendra les valeurs de $\mathrm{e}^{\varphi^{(n)}}$ aux nœuds x de Ω qui sont calculées par un sous-programme à la charge du développeur. Le tableau `ta` s'en déduit par multiplication par un scalaire (`Lambda`).

Procédures concernées, module lecdire

LCKDOM sous-procédure principale de lecture des calculs sur les domaines.

LCTERM lecture du nom d'un terme à calculer sur un domaine et procédure *principale* de lecture des caractéristiques du terme.

LCINCT lecture des noms des inconnues attachées à un terme.

LCVANO lecture du type de condition essentielle pour un terme valeurs nodales.

LCDONT lecture de l'éventuelle donnée affectée à un terme.

LCCADO lecture des caractéristiques de la donnée affectée à un terme.

Structures mises à jour

#TERDO tableau de description des calculs dur les domaines

#OMTRM tableau des noms des termes

$SDTRM tableau des attributs des termes

#OMDON tableau des noms des données

$DONNE tableau des attributs des données

6.3. Déclaration des termes 'Condition Essentielle de Dirichlet'

Il s'agit d'imposer certaines conditions aux limites essentielles (conditions de Dirichlet) qui entrent dans la définition de l'espace où est recherchée la solution et/ou auquel appartiennent les *fonctions-test*.

```
⊕ CALCUL [SUR] [LE] DOMAINE ||C_d
  [ IMPRESSION [DE] [ NIVEAU ||I_md ] ]
  CONDITION ESSENTIELLE ||C*6 [NIVEAU ||I_t] [DE] [TYPE] ||C_e
       [IMPRESSION [DE] [NIVEAU ||I_mt ]]
       [ DONNEE ||C_do
          [< CONSTANTE
              [< ENTIERE ∧ REELLE ∧ COMPLEXE >]
           ∧ TABLEAU [ NIVEAU ||I_do ]
              [< ENTIER ∧ REEL ∧ COMPLEXE ∧ CARACTERE*||I >]
           ∧ FONCTION
              [< REELLE ∧ COMPLEXE > ]
              [ DONNEE ASSOCIEE ||C_da
                [< CONSTANTE ∧ TABLEAU [NIVEAU ||I_da] >]
                [< ENTIER[E] ∧ REEL[LE] ∧ COMPLEXE >
          >]
       ]
```

où

- $\|C_d$ est le nom du domaine géométrique où l'on impose la condition ;
- $\|I_{md}$ est le niveau d'impression des calculs sur le domaine ;
- $\|C*6$ est le nom de la condition essentielle ; dans le cas d'une condition non homogène, c'est aussi le nom du terme qui contient les valeurs de *blocage* des d.l. ;
- $\|I_t$ est son niveau ;
- $\|C_e$ permet de définir le nom de l'inconnue à laquelle la condition se rapporte et le type de condition, type à choisir dans la liste des type de condition essentielle que l'on trouvera dans l'annexe **Type des termes 'Conditions Essentielles'** du présent chapitre.
  La chaîne $\|C_e$ est la concaténation du nom d'une inconnue préalablement définie à l'aide de la directive INCONNUE et d'une chaîne qui définit le type de la condition : par exemple '=G'[7] ou '.n=G' pour représenter respectivement une condition de Dirichlet pour une inconnue scalaire (*e.g.* $\varphi_{|\Sigma} = g$) et une de conditions essentielles courantes en pratique pour une inconnue vectorielle : (*e.g.* $\vec{u}.\vec{\nu} = g$ sur Σ, où $\vec{\nu}$ est le vecteur normal unitaire extérieur sur Σ).
- $\|I_{mt}$ est le niveau d'impression lors des calculs liés à la condition essentielle ;
- $\|C_{do}$ est le nom d'une donnée affectée au terme, etc.

⚠ L'absence de donnée affectée à un terme de condition essentielle implique une condition essentielle **homogène**. Dans le cas où une donnée de type fonction est fournie, cette fonction sera programmée dans la procédure FCTRM ; cette fonction permettra le calcul d'un terme du même type qu'un terme valeurs nodales, dit terme de condition essentielle. Dans le cas d'une donnée de type tableau, ce tableau contient les valeurs de *blocage* des d.l. pour cette condition essentielle.

6.3.1. Remarques sur les conditions esentielles vectorielles non homogènes

Les conditions essentielles de type $\vec{u} \wedge \vec{n} = \vec{g}$ et $\vec{u}.\vec{n} = g$ sont appliquées après passage au repère local orthonormé direct $(\vec{n}, \vec{t}_1, \vec{t}_2)$, ($\vec{n}$ normale unitaire extérieure, t_i vecteurs tangents). Dans cette base les inconnues deviennent $\vec{u}.\vec{n}$, $\vec{u}.\vec{t}_1$ et $\vec{u}.\vec{t}_2$.

(7) La notation 'G' n'a ici aucune signification particulière, et peut être remplacée par n'importe caractère, par exemple 0, les valeurs de blocage sont dans le cas non homogène définies par l'intermédiaire de la donnée affectée.

Remarquant que l'on a

$$\vec{u} = (\vec{u}.\vec{n})\vec{n} + (\vec{u} \wedge \vec{n}) \wedge \vec{n}\,,$$

la condition essentielle $\vec{u} \wedge \vec{n} = \vec{g}$, se ramène dans la base locale, aux conditions de Dirichlet

$$\vec{u}.\vec{t_i} = (\vec{n} \wedge (\vec{n} \wedge \vec{u})).\vec{t_i}\,.$$

de sorte que, si la donnée de Dirichlet $\vec{g}$ de $\vec{u} \wedge \vec{n} = \vec{g}$ est calculée à partir d'une fonction f (e.g. une solution exacte sur la frontière portant la condition) elle sera donnée par la formule

$$\vec{g} = \vec{n} \wedge (\vec{n} \wedge \vec{f})\,.$$

Cette donnée doit

– soit être programmée dans FCTRM (si la géométrie est telle que l'on connait $\vec{n}$) ;
– soit calculée comme tableau de valeurs nodales de type 'N∧(N∧F)' qui sera la donnée de la condition de Dirichlet.

Dans le cas d'une inconnue vectorielle à 2 composantes, on a

$$\vec{u} = (\vec{u}.\vec{n})\vec{n} + (\vec{u}.\vec{t})\vec{t} = (\vec{u}.\vec{n})\vec{n} + (\vec{n} \wedge \vec{u})\vec{t},$$

la condition essentielle $\vec{u} \wedge \vec{n} = g$, se ramène dans la base locale, à la condition de Dirichlet

$$\vec{u}.\vec{t} = (\vec{n} \wedge \vec{u}).\vec{t}\,.$$

de sorte que, si la donnée de Dirichlet g de $\vec{u} \wedge \vec{n} = g$ est calculée à partir d'une fonction $\vec{f}$ elle sera donnée par la formule

$$g = \vec{n} \wedge \vec{f}\,.$$

6.3.2. Exemples

Exemple 1 Condition de Dirichlet homogène sur le domaine Σ pour l'inconnue φ scalaire ou vectorielle

```
CALCUL SUR LE DOMAINE 'SIGMA'
       CONDITION ESSENTIELLE 'Cesse' 'PHI=0'
```

Le nom de la condition essentielle est PHISig ; cette condition étant homogène, il ne lui correspond aucun tableau et elle porte sur l'inconnue PHI, i.e. $\varphi_{|\Sigma} = 0$.

Exemple 2 Condition de Dirichlet non homogène sur le domaine Σ pour l'inconnue φ scalaire ou vectorielle, donnée fournie par fonction

```
CALCUL SUR LE DOMAINE 'SIGMA'
       CONDITION ESSENTIELLE 'Cesse' DE NIVEAU 2 'PHI=G'
         DONNEE 'DDiri' FONCTION
```

Exemple 3 Condition de Dirichlet non homogène sur le domaine Σ pour l'inconnue vectorielle $\vec{u}$, la donnée est fournie par FONCTION et porte sur la composante normale : $\vec{u}.\vec{n} = g$.

```
CALCUL SUR LE DOMAINE 'SIGMA'
       CONDITION ESSENTIELLE 'PHISig' 'U.N=G'
         DONNEE 'U.N' FONCTION REELLE
```

La donnée FONCTION à valeurs vectorielles de cette condition est programmée dans la procédure FCTRM.

Exemple 4 Condition de Dirichlet non homogène sur le domaine Σ pour l'inconnue vectorielle $\vec{u}$, la donnée est fournie par FONCTION et porte sur les composantes tangentielles : $\vec{u} \wedge \vec{n} = \vec{g}$ (cas 3D).

```
CALCUL SUR LE DOMAINE 'SIGMA'
       TERME VALEURS NODALES 'U∧N' DE NIVEAU 4 DE TYPE 'N∧(N∧F)'
         INCONNUE 'U'
         DONNEE 'U∧Vdat' FONCTION
       CONDITION ESSENTIELLE 'PHISig' 'U∧N=G'
         DONNEE 'UvN' TABLEAU DE NIVEAU 4
```

La donnée `FONCTION 'U∧Vdat'` à valeurs vectorielles de cette condition est programmée dans la procédure FCTRM, qui est utilisée pour la calcul de la donnée de Dirichlet pour le calcul des coefficients du tableau de valeurs nodales.
Dans le cas 2D, on remplace le type `'N∧(N∧F)'` du tableau de valeurs nodales par `'N∧F'`.

Procédures concernées, module lecdire

LCKDOM sous-procédure principale de lecture des calculs sur les domaines.
LCTERM lecture du nom d'un terme à calculer sur un domaine et procédure *principale* de lecture des caractéristiques du terme.
LCINCT lecture des noms des inconnues attachées à un terme.
LCESSE lecture du type de condition essentielle pour un terme condition essentielle.
LCDONT lecture de l'éventuelle donnée affectée à un terme.
LCCADO lecture des caractéristiques de la donnée affectée à un terme.

Structures mises à jour

#TERDO tableau de description des calculs dur les domaines
#OMTRM tableau des noms des termes
$SDTRM tableau des attributs des termes
#OMDON tableau des noms des données
$DONNE tableau des attributs des termes

6.4. Déclaration des termes 'Condition de Transmission'

Il s'agit d'imposer d'autres conditions aux limites liant les valeurs d'inconnues différentes de part et d'autre d'un domaine de bord. La syntaxe est analogue à celle d'une condition essentielle de Dirichlet :

```
⊕ CALCUL [SUR] [LE] DOMAINE ||C_d
  [ IMPRESSION [DE] [ NIVEAU ||I_md ] ]
  CONDITION DE TRANSMISSION ||C*6 [ NIVEAU ||I_t ] [DE] [TYPE] ||C_e
       [IMPRESSION [DE] [ NIVEAU ||I_mt ] ]
       [ DONNEE ||C_do
          [< CONSTANTE
              [< ENTIERE ∧ REELLE ∧ COMPLEXE >]
           ∧ TABLEAU [ NIVEAU ||I_do ]
              [< ENTIER ∧ REEL ∧ COMPLEXE ∧ CARACTERE*||I >]
           ∧ FONCTION
              [< REELLE ∧ COMPLEXE > ]
              [ DONNEE ASSOCIEE ||C_da
                [< CONSTANTE ∧ TABLEAU [NIVEAU ||I_da] >]
                [< ENTIER[E] ∧ REEL[LE] ∧ COMPLEXE >
           >]
       ]
```

où cette fois

||C_t permet de définir le nom des inconnues auxquelles la condition se rapporte et le type de condition, type à choisir dans la liste des type de condition de transmission que l'on trouvera dans l'annexe **Type des termes 'Conditions de Transmission'** du présent chapitre.

La chaîne $\|C_t$ est la concaténation de deux chaînes de caractères, séparées par le signe '=', chacune étant composée du nom d'une inconnue et, le cas échéant, d'une chaîne qui définit le type de la condition : par exemple '.N' ou '^N'pour représenter une condition de de transmission pour des inconnues vectorielles (*e.g.* $\vec{\varphi}_1 \cdot \vec{\nu} = \vec{\varphi}_2 \cdot \vec{\nu}$ sur Σ ou $\vec{\varphi}_1 \wedge \vec{\nu} = \vec{\varphi}_2 \wedge \vec{\nu}$ sur Σ où $\vec{\nu}$ est le vecteur normal unitaire sur Σ).

⚠ ⚠ Les inconnues figurant dans la chaîne $\|C_t$ doivent apparaître dans l'ordre où les inconnues ont été déclarées par l'utilisateur à l'aide de la directive INCONNUE.

Exemple 1 Condition de transmission homogène sur le domaine Σ pour φ_1 et φ_2

```
CALCUL SUR LE DOMAINE 'SIGMA'
        CONDITION DE TRANSMISSION 'PHISig' 'PHI1=PHI2'
```

Le nom de la condition de transmission est PHISig ; cette condition étant homogène, il ne lui correspond aucun tableau et elle porte sur les inconnues PHI1 et PHI2, i.e. $\varphi_{1_{|\Sigma}} = \varphi_{2_{|\Sigma}}$.

Exemple 2 Condition de transmission non homogène sur le domaine Σ pour les inconnues vectorielles $\vec{u}_1$ et $\vec{u}_2$:

$$\vec{u}_1 \wedge \vec{n}_{|\Sigma} = \vec{u}_2 \wedge \vec{n}_{|\Sigma} + \vec{g}$$

```
CALCUL SUR LE DOMAINE 'SIGMA'
        CONDITION DE TRANSMISSION 'USig' 'u1^N=u2^N+g'
             DONNEE 'Ftran' FONCTION REELLE
```

Exemple 3 Condition de transmission sur le domaine Σ pour les inconnues vectorielles $\vec{u}_1$ et $\vec{u}_2$, avec coefficients variables K_1 et K_2 à valeurs matricielles

$$K_1\vec{u}_1 \cdot \vec{n}_{|\Sigma} = K_2\vec{u}_2 \cdot \vec{n}_{|\Sigma}$$

```
CALCUL SUR LE DOMAINE 'SIGMA'
        CONDITION DE TRANSMISSION 'USig' 'u1.N=u2.N'
          DONNEE 'TuSig' TABLEAU CARACTERE*2
```

Le tableau de 'TuSig' de type chaîne de caractères contiendra le nom des données représentant K_1 et K_2. Les données K_1 et K_2, qui sont soit des constantes, soit des tableaux, devront être soit donnés á l'aide de la directive DONNEE, soit calculés comme termes VALEURS NODALES, par exemple pour des constantes

```
DONNEE DE NOM 'TuSig' de LONGUEUR 2 : 'K1' 'K2'
              'K1' CONSTANTE REELLE 1.
              'K2' CONSTANTE REELLE 10.
```

Procédures concernées, module lecdire

LCKDOM sous-procédure principale de lecture des calculs sur les domaines.
LCTERM lecture du nom d'un terme à calculer sur un domaine et procédure *principale* de lecture des caractéristiques du terme.
LCINCT lecture des noms des inconnues attachées à un terme.
LCTRAN lecture du type de condition pour un terme condition de transmission.
LCDONT lecture de l'éventuelle donnée affectée à un terme.
LCCADO lecture des caractéristiques de la donnée affectée à un terme.

Structures mises à jour

#TERDO tableau de description des calculs dur les domaines
#OMTRM tableau des noms des termes
$SDTRM tableau des attributs des termes
#OMDON tableau des noms des données
$DONNE tableau des attributs des termes

6.5. Déclaration des calculs des normales (à revoir pour interpolation)

Il s'agit de demander le calcul des composantes de la normale unitaire extérieure sur un domaine de bord. La normale est définie comme étant extérieure aux éléments dont les faces définissent le domaine de bord, il n'y a jamais d'ambiguïté en ce qui concerne son orientation. Ces tableaux sont nécessaires dans les cas suivants :

i) On souhaite calculer les valeurs d'un terme valeurs nodales de dérivée normale ;
ii) On souhaite calculer les valeurs de blocage sur un bord portant une condition essentielle faisant intervenir les composantes de la normales (voir Annexe 3) ;
iii) On utilise la méthode de couplage éléments finis–représentation intégrale : on a besoin des normales sur le domaine portant la représentation intégrale ou sur le domaine de couplage.

⚠ Dans les cas (i) et (ii), la directive est inutile, le programme se charge automatiquement de renseigner la structure #TERDO pour le domaine. Par contre dans le cas (iii), le calcul des composantes de la normale doit être demandé explicitement, puisqu'aucune directive ne renseigne sur l'utilisation de la méthode Éléments Finis–Représentation Intégrale.

```
⊕ CALCUL [SUR] [LE] DOMAINE ||C_d
         [ IMPRESSION [DE] [ NIVEAU ||I_md ] ]
         [ NORMALES [ASSEMBLEES] ]
              INCONNUE ||C_c
```

où

$||C_d$ est le nom du domaine géométrique où l'on demande les calculs des normales ;
$||C_c$ est l'inconnue valeurs nodales aux nœuds de laquelle les normales seront calculées.

△ Nous avons présenté séparément les différentes directives de calcul de termes sur les domaines. On peut évidement les regrouper pour un même domaine.

Exemple

```
CALCUL SUR LE DOMAINE 'GAMMA'
       TERME ELEMENTS FINIS 'PDELTA'
        INCONNUE 'PHI'
        INTEGRAND 'UV'
       TERME VALEUR NODALE 'NEUGAM' DE TYPE 'N.F'
        INCONNUE 'PHI'
        DONNEE 'DPHIn' FONCTION COMPLEXE
         DONNEE ASSOCIEE 'k&teta' TABLEAU DE NIVEAU 1
       NORMALES ASSEMBLEES
        INCONNUE 'PHI'
```

7. Déclaration des autres calculs

Directive AUTRES CALCULS

7.1. Déclaration des termes

Il s'agit essentiellement de définir les termes qui recevront les matrices et vecteurs qui ne peuvent être calculées sur les domaines de calcul comme termes éléments finis, **et dont le calcul effectif est sous l'entière responsabilité du développeur.** La syntaxe de déclaration de tels termes est la suivante :

```
⊕ AUTRES CALCULS
   ⊕ TERME ||C*6 [[DE] NIVEAU ||I_t]
   [ IMPRESSION [DE] [ NIVEAU ||I_mt ] ]
   [ INCONNUE [DE NOM] ||C_c EN COLONNE ]
   [ INCONNUE [DE NOM] ||C_l EN LIGNE ]
```

où

- ||C*6 est le nom d'un terme matriciel ou du vecteur.
- $||I_t$ est son niveau (valeur par défaut 1). Le couple (||C*6, $||I_t$) désigne aussi le tableau géré par l'allocation dynamique qui contiendra ce terme ;
- $||I_{mt}$ est le niveau d'impression lors du calcul du terme (par défaut niveau 0 : pas d'impression) ;
- $||C_c$ représente le nom de l'inconnue en colonne pour le terme (cette notion n'a d'intérêt que pour un terme matriciel). Cette inconnue doit avoir été préalablement définie à l'aide de la directive INCONNUE ;
- $||C_l$ représente le nom de l'inconnue en ligne pour le terme.

⚠ Dans cette directive, un terme matriciel est défini par un couple d'inconnues $||C_c$ et $||C_l$ qui doivent toutes les deux être données.

Procédure concernée, module lecdire

LCAUTR lecture du nom d'un terme (autre qu'éléments finis) et des inconnues qui s'y rattachent.

Structures de données remplies

#OMTRM tableau des noms de termes.
$SDTRM tableau des caractéristiques des termes.

8. Définition de l'assemblage des termes

Directive ASSEMBLAGE

Cette directive permet de construire les structures d'assemblage $ASMBL pour définir de nouveaux termes résultats de l'assemblage d'autres termes (construction de la matrice d'un système linéaire, par exemple). Dans certains cas où l'on doit assembler des termes définis sur un même domaine de calcul et se rapportant à la même inconnue (en ligne, et le cas échéant en colonne) on aura intérêt, pour minimiser les calculs, à utiliser les procédures de combinaisons linéaires CLTERM, T2TERM (voir la documentation du module assembl). Dans les autres cas, l'assemblage standard est utilisé.

L'assemblage proprement dit est déclenché par appel à l'une des procédures ASMTRM, ASVTRM ou DSMTRM, DSVTRM selon que le résultat est un terme matriciel ou vectoriel et que l'assemblage est uni-inconnue ou multi-inconnue.

La déclaration des constituants de l'assemblage d'un terme est effectuée sous la forme :

```
⊕ ASSEMBLAGE [ DU TERME ] ||C*6_a [ [DE NIVEAU] ||I_a ]
  [ IMPRESSION [ [DE NIVEAU] ||I_m ] ]
  [ < SYMETRIQUE ∧ ANTISYMETRIQUE ∧ AUTOADJOINT ∧ ANTIADJOINT > ]
  [ DONNEE ||DO [ CONSTANTE < ENTIER[E] ∧ REEL[LE] ∧ COMPLEXE > ] ]

  < [ COMPOSE ] [DES] [TERMES] ⊕ ||C*6_t [ [NIVEAU] ||I_t ]
  ∧ MULTI-INCONNUE [DES] [TERMES] ⊕ ||C*6_t [ [NIVEAU] ||I_t ] >
```

où

- ||C*6$_a$ est le nom du terme résultat de l'assemblage ;
- ||I$_a$ est son niveau (égal à 1 par défaut). Le couple (nom, niveau) désigne aussi le tableau géré par l'allocation dynamique qui contiendra ce terme ;
- ||I$_m$ est le niveau d'impression du terme ;
- ||C*6$_t$ est le nom d'un des termes opérandes de l'assemblage ;
- ||I$_t$ est son niveau (1 par défaut).

Dans le cas d'assemblage simple (par défaut) ou multi-inconnue les constantes de la combinaison sont celles qui sont associées aux termes constituants (1. par défaut). Il est possible d'effectuer le produit d'un terme par sa donnée associée, il suffit de 'combiner linéairement' le terme seul.

⚠ Dans tous les cas le terme résultat (||C*6$_a$,||I$_a$) est nécessairement distinct de tous les termes constituants (||C*6$_t$,||I$_t$).

△ La construction des structures d'assemblage à l'aide de cette directive peut être remplacée par des appels consécutifs des procédures ASMAKE ou DSMAKE ; on se reportera à la section 'Assemblage des termes' du chapitre **Rédaction du programme principal**.

Exemple 1 Déclaration de l'assemblage du terme matriciel uni-inconnue

$$\int_\Omega \overrightarrow{\text{grad}}\,\varphi\ \overrightarrow{\text{grad}}\,\psi\,dx - k^2 \int_\Omega \varphi\,\psi\,dx + ik \int_\Sigma \varphi\,\psi\,d\sigma$$

```
ASSEMBLAGE DU TERME 'MATRIS' DE NIVEAU 1
           TERMES 'RIGID' DE NIVEAU 1 'MASSE' DE NIVEAU 1
               ET 'GDELTA' DE NIVEAU 1
```

Exemple 2 Déclaration de l'assemblage du terme matriciel multi-inconnue

$$\begin{cases} \nu \displaystyle\int_\Omega \overrightarrow{\text{grad}}\,\vec{u} \cdot \overrightarrow{\text{grad}}\,\vec{v}\,dx - \int_\Omega p\ \text{div}\,\vec{v}\,dx \\ \displaystyle\int_\Omega \text{div}\,\vec{u}\,q\,dx \end{cases}$$

```
ASSEMBLAGE DU TERME 'MATRIS' DE NIVEAU 1 MULTI-INCONNUE
           TERMES 'RIGID' DE NIVEAU 1 'PDIVV' DE NIVEAU 1
               ET 'DIVUQ' DE NIVEAU 1
```

Exemple 3 Déclaration de l'assemblage d'un terme matriciel uni-inconnue suivi d'un assemblage matriciel multi-inconnue

$$\int_\Omega \overrightarrow{\text{grad}}\,\varphi \cdot \overrightarrow{\text{grad}}\,\psi\,dx + \int_\Omega a\,\varphi\,\psi\,dx + \Lambda \int_\Omega b\,\psi\,dx$$

```
ASSEMBLAGE DU TERME 'MATPHI' DE NIVEAU 1
           COMPOSE DES TERMES 'RIGID' ET 'MASSE'
ASSEMBLAGE DU TERME 'MATRIC' DE NIVEAU 1 MULTI-INCONNUE
           TERMES 'MATPHI' DE NIVEAU 1 ET 'BPSI' DE NIVEAU 1
```

Procédure concernée, module lecdire

LCASSE lecture des données d'assemblage d'un terme.
LCDONT lecture d'une donnée affectée à un terme.

Structures de données mises à jour

#OMTRM tableau des noms des termes
$SDTRM structure des attributs des termes
$ASMBL structure pilotant l'assemblage d'un terme uni-inconnue.

9. Déclaration des données et affectation de valeurs

Directive DONNEE

Dans les directives précédentes, on a vu apparaître les données affectées à des termes. Il s'agit maintenant de leur affecter une **valeur**, ce qui signifie que l'on va définir la valeur des données de type constante ou des coefficients des données de type tableau. On peut aussi définir de nouvelles données de type constante, fonction ou tableau, non nécessairement affectées à des termes.
La directive DONNEE permet à la fois de définir de nouvelles données et d'affecter une valeur aux données dites *constantes* [8]. Ces données seront dans l'exemple du problème de Helmholtz, la valeur de k et celles qui s'en déduisent $-k^2$ ou $-ik$; dans l'exemple du problème de Stokes la valeur de ν.

9.1. Affectation d'une valeur à une donnée constante

Lorsqu'il s'agit seulement d'affecter une valeur à une donnée constante dont les caractéristiques ont été préalablement définies, la directive DONNEE s'écrit simplement :[9]

⊕ DONNEE ⊕ [DE] [NOM] ||C_d = < ||I ∧ ||R ∧ ||R_e ||R_i ∧ ||C >

où

||C_d est le nom de la donnée de type constante
||I est une valeur entière
||R est une valeur réelle
||R_e est la valeur réelle de partie réelle d'un complexe
||R_i est la valeur de la partie imaginaire de ce même complexe
||C est une valeur chaîne de caractères

△ Le type de la donnée étant défini préalablement, la valeur donnée lui est affectée selon les règles d'affectation de Fortran77.

⚠ Une donnée réelle ||R de partie décimale nulle peut évidement être entrée comme une valeur entière ; l'inverse provoque une erreur, l'affectation d'un réel dans un entier produisant une troncature (le programme ne peut autoriser la modification des données de l'utilisateur !).

9.2. Affectation des valeurs d'une donnée tableau

Il est possible d'affecter des valeurs à tous les coefficients d'une donnée tableau de type numérique (i.e. non (chaîne de) caractères) dont les caractéristiques ont été préalablement définies, selon une syntaxe similaire à celle rencontrée ci-dessus, mais cette fois il est nécessaire de définir en outre la longueur (comptée en articles) du tableau :

⊕ DONNEE ⊕ [DE] [NOM] ||C_d [DE] LONGUEUR ||I_t ⊕ < ||I ∧ ||R ∧ ||R_e ||R_i >

où

||C_d est le nom de la donnée tableau ;
||I_t est la longueur du tableau comptée en articles ;
etc.

△ Les valeurs ||I, ||R, (||R_e, ||R_i) sont affectées aux coefficients du tableau successivement et dans l'ordre des adresses croissantes, et le nombre de valeurs doit nécessairement être inférieur ou égal à la longueur déclarée du tableau.

(8) Le mot est peut être mal choisi, ces *constantes* pouvant évidement changer de valeur en cours d'exécution !
(9) Le signe = est un séparateur des articles de données, il n'est ici que pour faire joli !

9.3. Définition d'une nouvelle donnée

La directive DONNEE permet aussi de définir de nouvelles données et, dans le cas d'une donnée constante de lui affecter directement une valeur, mais dans ce cas il est nécessaire de fournir les caractéristiques de la donnée. (10)

```
⊕ DONNEE ⊕ [DE] [NOM] ||C_d
           [ < CONSTANTE
               [ < ENTIERE ∧ REEL ∧ COMPLEXE ∧ CARACTERE > ]
               = < ||I ∧ ||R ∧ ||R_e ||R_i ∧ ||C >
             ∧ TABLEAU
               [ [DE] [ NIVEAU ||I_d ]
               [ [DE] LONGUEUR ||I_t ]
               [ < ENTIER ∧ REEL ∧ COMPLEXE ∧ CARACTERE||I > ]
               [ ⊕ = < ||I ∧ ||R ∧ ||R_e ||R_i ∧ ||C > ]
             ∧ FONCTION
               [ < ENTIERE ∧ REELLE ∧ COMPLEXE > ]
               [ DONNEE ASSOCIEE ||C_da
                 [< CONSTANTE
                    [< ENTIERE ∧ REELLE ∧ COMPLEXE >]
                    [ = < ||I_a ∧ ||R_a ∧ ||R_ae ||R_ai > ]
                  ∧ TABLEAU
                    [DE] [ NIVEAU ||I_da ]
                    [< ENTIER ∧ REEL ∧ COMPLEXE >]
                    [ [DE] LONGUEUR ||I_ta ]
                    [ ⊕ = < ||I_a ∧ ||R_a ∧ ||R_ae ||R_ai > ]
                 >]
           > ]
```

où

$||C_d$ est le nom de la donnée nouvelle.

– Son **type de représentation** peut être

- ▷ CONSTANTE (affectée à un terme, nécessaire au calcul de fonction de Green, le nombre de boucles d'une exécution, l'âge du nombre de dents de la poule du capitaine, etc.)
- ▷ FONTION qui correspond à la programmation effective d'instructions définissant cette fonction dans la procédure utilisateur FCTRM ou DFCTRM. Son nom de donnée pourra être utilisé pour la différencier des autres données de type fonction de l'application. La notion de donnée fonction n'a évidement d'intérêt que si cette donnée est affectée à un terme. Il est possible de transmettre des informations particulières à la procédure FCTRM (ou DFCTRM), soit sous la forme d'une donnée constante, soit sous la forme d'un tableau ; il suffit pour ce faire d'associer cette autre donnée à la donnée fonction (voir aussi les sous-directives TERME ELEMENTS FINIS, TERME VALEURS NODALES, ou CONDITION ESSENTIELLE, plus haut ou la procédure DOASFC, librairie sdexplo).
- ▷ TABLEAU dont le contenu est laissé à l'entière responsabilité du développeur de l'application. Ce tableau peut être rempli, soit par exemple en temps que tableau de valeurs nodales par le module de calcul 'éléments finis' (auquel cas cette donnée est à la fois **terme** et **donnée**), soit *à la main*.

– Le **type de déclaration** d'une donnée est soit REEL (valeur par défaut), soit COMPLEXE, soit encore ENTIER ou CARACTERE pour les données constantes ; c'est-à-dire que, dans le cas d'une donnée CONSTANTE, elle est de type entier, réel, complexe ou (chaîne de) caractères ; dans le cas d'une donnée TABLEAU, il est de type entier, réel ou complexe ;

(10) On verra plus loin qu'il est possible de réaliser ces définition et affectation par procédure dans le programme principal, voir les procédures PUTCST et GETCST.

dans le cas d'une FONCTION, que le résultat retourné par la procédure FCTRM est réel ou complexe.

||I$_d$ est le niveau (donnée optionnelle) d'une donnée de type tableau ;

||I$_t$ est la longueur d'une donnée de type tableau (nécessaire si on doit affecter des valeurs aux coefficients de ce tableau à l'intérieur de cette directive) ;

||I est une valeur entière d'affectation ;

||R est une valeur réelle d'affectation ;

||R$_e$ est la valeur de la partie réelle d'une donnée complexe ;

||R$_i$ est la valeur de la partie imaginaire ;

||C est la valeur d'une donnée chaîne de caractères.

||C$_{da}$ Dans le cas où la donnée affectée à un terme est de type fonction, il est possible *d'associer à la donnée de type fonction*, une donnée de type constante ou tableau. Par exemple dans le cas d'un problème de diffraction du type de l'exemple 1, la donnée de Neumann de second membre g peut dépendre de la constante k : pour une onde incidente plane se propageant dans la direction x, on aurait $g = \mathrm{e}^{ikx}$.
Dans le cas d'une donnée de type tableau associée à une fonction, le contenu du tableau est entièrement à la discrétion du développeur de l'application, le code se charge seulement de transmettre son adresse et son type (cf. la procédure *'utilisateur'* FCTRM).

||I$_{da}$ est le niveau de la donnée tableau associée à une donnée de type fonction (valeur par défaut 1).

||I$_a$ est la valeur entière de la donnée constante associée (à une donnée ||C$_d$ de type fonction) de type entier ou réel ;

||R$_a$ est la valeur réelle d'une donnée constante associée de type réel ;

||R$_{ae}$ est la valeur de la partie réelle d'une donnée constante associée de type complexe ;

||R$_{ai}$ est la valeur de la partie imaginaire de cette donnée complexe.

Valeur par défaut

Elle correspond, pour la donnée de nom k par exemple, à la syntaxe :

```
DONNEE 'k' CONSTANTE REELLE
```

Exemple 1 Définition de la constante complexe $\lambda = i$

```
DONNEE 'LAMBDA' CONSTANTE COMPLEXE = 0. 1.
```

Exemple 2 Définition du tableau réel k&teta à 2 coefficients $1/2$ et $\pi/2$

```
DONNEE 'K&TETA' TABLEAU REEL DE LONGUEUR 2 = 0.5 $PI/2$
```

Exemple 3 Définition de la fonction complexe $\phi_w = \mathrm{e}^{ikx}, \quad k \in \mathbb{R}$

```
DONNEE 'PHIw' FONCTION COMPLEXE DONNEE ASSOCIEE 'k' = $PI$
```

La procédure FCTRM contiendra les instructions suivantes définissant ϕ_w se trouve dans le l'encadré 1, dans lequel

POINT est le tableau contenant les coordonnées du point où la fonction ϕ_w est évaluée ;

NOMFCT est la variable chaîne réceptrice du nom de la donnée fonction, ici PHIw ;

TYPFCT est le type de déclaration de la donnée fonction REEL(LE) ou COMPLEXE ;

MCASSO est l'adresse de la constante affectée, ici k dans le super-tableau scalaire dans la version du type de la donnée constante affectée (IST pour le type entier, RST pour le type réel, CST pour le type complexe).

△ On trouvera la liste détaillée de tous les arguments de la procédure *'utilisateur'* FCTRM dans le chapitre **Les Procédures 'Utilisateurs'**.

```
      SUBROUTINE FCTRM (NDIM,POINT,NOMFCT,TYPFCT,NODONA,TYDONA
                       ,TYPRAS,MCASSO,IST,RST,CST,RESULT,CESULT)
*
      ...
      CHARACTER*(*) NOMFCT,TYPFCT,NODONA,TYDONA,TYPRAS
      INTEGER       NDIM,MCASSO,IST(*)
      REAL          POINT(NDIM),RST(*)
      COMPLEX       CST(*),CESULT
      ...
*     Donnees fonctions complexes
      IF (TYPFCT(1:1).EQ.'C') THEN
         IF (NOMFCT(1:4).EQ.'PHIw') THEN
             CESULT=EXP(CMPLX(0.,RST(MCASSO)*POINT(1))
         ELSE
             ...
         ENDIF
*     Donnees fonctions reelles
      ELSE
         ...
      ENDIF
      END
```

Encadré 1

Exemple 4 Définition de la fonction complexe $\phi_w = \mathrm{e}^{ik(x\cos\theta - y\sin\theta)}$, k, θ réels

```
DONNEE'PHIw' FONCTION COMPLEXE
      DONNEE ASSOCIEE 'k&teta' TABLEAU REEL DE NIVEAU 1
```

La fonction FCTRM contiendra les instructions suivantes définissant ϕ_w (voir encadré 2) dans lequel
MCASSO est l'adresse du tableau affecté, ici k&teta, dans le super-tableau numérique dans la version du bon type. Ce tableau de longueur 2 est supposé ici contenir les valeurs de k et θ dans cet ordre.

Procédures concernées, module lecdire et librairie utiliter

LCDONN Lecture des noms de données et gestion des structures de description des données.
LCCADO Lecture des caractéristiques de la donnée affectée à un terme.
LCCSTE Lecture des valeurs des données de type constante.
INCST Affectation des valeurs des constantes dans les tableaux de valeurs des constantes $ACSTE, $ECSTE, $RCSTE et $CCSTE, voir ci-dessous.

Structures de données remplies ou consultées

#OMDON Tableau des noms de données.
$DONNE Tableau des caractéristiques des données.
$ACSTE Tableau de caractères contenant les valeurs des données constantes (chaînes de) caractères.
$LGCSA Tableau entier contenant les longueurs des données constantes (chaînes de) caractères. Le premier élément du tableau contient le nombre de telles constantes.
$ECSTE Tableau entier contenant les valeurs des données constantes entières. Le premier élément du tableau contient le nombre de constantes entières.
$RCSTE Tableau réel contenant les valeurs des données constantes réelles. Le premier élément du tableau contient le nombre de constantes entières, il doit donc être systématiquement manipulé comme un entier, c'est-à-dire comme élément du super-tableau version entière IST.

```
      SUBROUTINE FCTRM (NDIM,POINT,NOMFCT,TYPFCT,NODONA,TYDONA
                       ,TYPRAS,MCASSO,IST,RST,CST,RESULT,CESULT)
*
      ...
      CHARACTER*(*) NOMFCT,TYPFCT,NODONA,TYDONA,TYPRAS
      INTEGER       NDIM,MCASSO,IST(*)
      REAL          POINT(NDIM),RST(*),RESULT
      COMPLEX       CST(*),CESULT
*
      REAL          K,TETA
      ...
*     Donnees fonctions complexes
      IF (TYPFCT(1:1).EQ.'C') THEN
         IF (NOMFCT(:4).EQ.'PHIw'.AND.NODONA(:6).EQ.'k&teta') THEN
            K=RST(MCASSO)
            TETA=RST(MCASSO+1)
            CESULT=
     &      EXP(CMPLX(0.,K*(POINT(1)*COS(TETA)-POINT(2)*SIN(TETA)))
         ELSE
            ...
         ENDIF
      ...
      ENDIF
      END
```

Encadré 2

$CCSTE Tableau complexe contenant les valeurs des données constantes complexes. La partie réelle du premier élément du tableau contient le nombre de constantes complexes, il doit donc être systématiquement manipulé comme un entier, c'est-à-dire comme élément du super-tableau version entière IST. La partie imaginaire de ce premier élément de $CCSTE n'est pas utilisée.

– Mai 2000 –

Notions de programmation

Nous présentons dans ce chapitre, les procédures dites de **haut niveau** qui permettent de construire une application simple du code MÉLINA, i.e. une application qui ne nécessite pas le développement de nouvelles structures de données ou de nouveaux modules de calculs.
Un programme principal d'une application développée avec MÉLINA, comporte essentiellement 5 phases, dont les 4 premières sont prédéfinies et peuvent être pilotées par directives :

Initialisation
Lecture du maillage
Lecture des données
Calculs éléments finis

La dernière, généralement aisément programmable, est entièrement à la charge du **développeur** de l'**application**. Il s'agit de la définition de l'algorithme de résolution qui est mise en œuvre à l'aide des procédures de **haut niveau**.

1. Initialisation et lecture des données

1.1. Initialisation

`INITIE`

Cette première phase du programme est l'initialisation de l'allocation dynamique et la création des structures de données indispensables pour le démarrage de l'exécution, elle consiste en un simple appel de la procédure `INITIE`, module initial, dont les arguments permettent essentiellement de transmettre à la librairie d'allocation dynamique la taille des 2 super-tableaux du code.

1.1.1. Déclaration des super-tableaux

Il n'y a pas de critère vraiment établi pour déterminer a priori la taille de ces tableaux[(1)](ça commence bien !). Il existe en effet un seuil minimal, qui dépend de la taille du maillage, du nombre de termes de la formulation variationnelle, de l'algorithme de résolution et de la programation de l'utilisateur, en deçà duquel le code ne peut fonctionner et . . . s'en plaint. Tant que ce minimum n'est pas atteint le code se charge automatiquement des transferts entre la mémoire centrale et la mémoire secondaire (entrées/sorties) de tous les tableaux pour lesquels il a reçu l'autorisation de sauvegarde (cf. procédure `TBSAVE` de la librairie alodyn). Il est cependant certain que le code fonctionnera plus rapidement si l'on a la possibilité de déclarer de gros super-tableaux. À titre d'exemple, un super-tableau de caractères de 80 000 caractères semble couvrir la majorité des cas d'exécutions moyennes à grosses. En ce qui concerne le super-tableau numérique la question de la détermination de critères fiables pour définir sa taille reste largement ouverte . . .

Les 2 super-tableaux se trouvent dans 2 zones communes :

- le `COMMON` *blanc* pour le super-tableau numérique, désigné sous le nom `IST`, `RST` ou `CST`, selon le type (entier, réel ou complexe) des données auxquelles on s'intéresse ;
- le `COMMON/STCHAR/` pour le super-tableau de caractères, désigné sous le nom `AST`.

La déclaration de la taille de ces super-tableaux se fait dans le programme principal, par exemple sous la forme :

```
      PARAMETER ( MAXAST = 80000 , MAXIST= 16000000 )
      COMMON/STCHAR/AST(MAXAST)  /      /IST(MAXIST)
```

(1) Une estimation a posteriori est possible : il est effet possible à tout moment dans le programme principal d'afficher (dans l'un des fichiers d'impression) l'état des super-tableaux et d'en déduire approximativement leurs tailles.

⚠ Il est nécessaire de rappeler que selon la norme Fortran77 la taille d'un common nommé ou étiqueté (c'est-à-dire autre que le COMMON *blanc*) ne peut différer d'une procédure à un autre d'un même programme. La longueur du COMMON/STCHAR/ doit donc être la même dans toutes les procédures du code. Si l'on souhaite modifier sa longueur, il est nécessaire de modifier le fichier d'**inclusion** 'ALLOC', et par suite de recompiler toutes les procédures incluant ce fichier à l'aide de la métacommande du compilateur INCLUDE. Pour sa part le COMMON *blanc* est déclaré de longueur 1 dans ce même fichier, dont voici la partie concernant les super-tableaux (on notera que la modification de la taille du super-tableau numérique ne nécessite que la compilation du programme principal, et une édition de liens) :

Fichier ALLOC

```
      COMMON/       /   RST(1)
      INTEGER           IST(1)
      DOUBLE PRECISION  DST(1)
      COMPLEX           CST(1)
      CHARACTER         AST
      COMMON/STCHAR/    AST(80000)
      EQUIVALENCE      (RST(1),IST(1),DST(1),CST(1))
```

Le fichier d'inclusion ALLOC est incorporé dans toutes les procédures utilisant explicitement les super-tableaux (en règle générale toutes les procédures de haut niveau) à l'aide de la déclaration (ou métacommande de compilation) INCLUDE autorisée par les compilateurs comme une instruction par exemple sous la forme :

```
      INCLUDE 'ALLOC'
```

(*fortran f77 sous Unix, avec un lien symbolique idoine ou l'option de compilation adéquate.*)

Ce fichier contient en outre la déclaration d'un certain nombres de constantes 'PARAMETER' propres aux procédures de l'allocation dynamique qu'il n'est pas nécessaire de détailler ici.

La première instruction exécutable de la procédure principale consiste alors à communiquer ces valeurs aux tables de gestion de la librairie d'allocation dynamique alodyn et à initialiser ces tables. Elle s'écrit simplement :

```
      CALL INITIE (MAXAST,MAXIST)
```

Cette procédure, et celles qu'elle appelle, INITIA, INISYS, CREESD, est chargée de la création des structures de données du code et de la lecture des données d'**Initialisation de l'allocation dynamique** (voir cette section dans le chapitre **Lecture des données par directives**).

1.1.2. Modification de valeurs par défaut lors de l'initialisation

Cette procédure (INITIE) permet en outre l'affectation de valeurs par défaut des variables entières suivantes (s'agissant de valeurs par défaut il n'est nullement nécessaire de modifier ces valeurs pour que le code fonctionne sur tout système, on peut donc sauter ce paragraphe en première lecture[(2)]) :

USYSIN Numéro logique du fichier d'entrée standard (sdtin : standard input file). La valeur programmée est USYSIN=5. Elle ne peut pas être modifiée par donnée.

USYSPR Idem mais pour l'unité logique de sortie standard sdtout : standard output file). La valeur par défaut programmée est USYSPR=6. Le fichier d'impression principal peut être modifié par donnée (directive utilitaire ECRITURE).

LASNUF Plus grand nombre de fichiers ouverts simultanément lors d'une exécution (par exemple 63 pour $f77$ sur SUN, 999 sur DEC_ALPHA). Cette variable transite par le COMMON/BORDEL/ (cf. la documentation concernant les **Structures de données**, librairie sdexplo), elle ne peut pas être modifiée par donnée.

(2) Ce paragraphe sort un peu du cadre de la rédaction du programme principal, il s'agit de modifier les procédures d'initialisation du code.

LGBUF Longueur des tableaux tampons utilisés par la librairie alodyn pour les accès à la mémoire secondaire ; ces transferts de la mémoire centrale à des fichiers à accès direct (mémoire secondaire) sont entièrement gérés par les procédures de l'allocation dynamique. La valeur par défaut est fixée à 5120 mots sur SUN. Cette valeur peut être modifiée par donnée à l'aide de la directive ALLOCATON DYNAMIQUE (voir la documentation concernant la **Lecture des Données**).

NBENRX Nombre maximum d'enregistrements sur un des fichiers à accès direct qui définissent la mémoire secondaire. Le produit NBENRX×LGBUF définit la taille maximum d'un fichier à accès direct sur le système sur lequel on exécute le code. A titre d'exemple on a choisi NBENRX=1000 sur SUN.
Cette valeur par défaut peut être modifiée par donnée à l'aide de la directive ALLOCATION DYNAMIQUE.

Ces valeurs par défaut programmées peuvent être modifiées en intervenant directement dans le source de la procédure INISYS qui ne contient que les instructions d'affectation de ces valeurs. Cette procédure fait partie, ainsi que les quelques autres procédures dépendant de la machine, de la librairie util-xxx où xxx définit le nom de la machine et/ou du système d'exploitation (Linux, sun, alpha, sgi, etc.). C'est la première procédure appelée par la procédure INITIE, module initial.

L'affectation des autres valeurs par défaut (qu'il ne semble pas nécessaire de modifier en fonction du système) est effectuée directement dans la procédure INITIE. Il s'agit des variables suivantes :

IMPSDR Numéro d'unité logique du fichier de sortie secondaire. Ce fichier contiendra tous les renseignements demandés par donnée (voir les mots-clés IMPRESSION dans les données par directives) ou par procédure (voir l'argument NIVIMP des procédures dites de *haut niveau)*. Si on ne souhaite aucune impression sur ce fichier on peut court-circuiter les valeurs des paramètres mentionnés ci-dessus en posant IMPSDR=0 dans le source de INITIE.
Valeur actuelle affectée dans le source de INITIE *:* 8.

IMPMES Numéro d'unité logique du fichier de sortie des messages. Il contiendra les échos des directives des données, ainsi que leur interprétation mot-à-mot par le code (ceci peut être très utile en cas d'introduction de *bugs* dans les données).
Valeur actuelle affectée dans le source de INITIE *:* 9.

IMPALO Numéro d'unité logique du fichier *'mouchard'* de l'allocation dynamique (voir la sous-directive MOUCHARD BAVARD ou la procédure MOUCHA('ON'/'OFF')).
Valeur actuelle affectée dans le source de INITIE *:* 10.

Procédures concernées pour la phase d'initialisation

INITIE module initial– initialisation de l'exécution (en particulier des tables de gestion de l'allocation dynamique).

LCALDY module initial– lecture des données de la directive ALLOCATION DYNAMIQUE.

LCSTRU module initial– lecture des données relatives aux tailles initiales des structures (peut accélérer les grosses exécutions).

INIFIC module initial– initialisation du tableau #unite de gestion des fichiers ouverts.

CREESD module initial, création des structures indispensables du code (leur taille initiale peut être fixée ci-dessus.).

STINIT librairie alodyn– initiaisation des tables de description de l'allocation dynamique.

CHRONO librairie utiliter– utilitaire d'initialisation et d'affichage des temps de calcul intermédiaires (utilise la fonction C ctemps.c).

HORODA librairie utiliter– impression date/heure (utilise la fonction C cdate.c).

INISYS librairie util-xxx– initiaisation de différents paramètres dépendant du système d'exploitation (nombre de fichiers ouvrets simultanément, entier, réel maximum, etc.).

1.2. Maillage

1.2.1. Lecture du fichier de maillage

LCGEOM

Cette phase est du point de vue programmation réduite à sa plus simple expression sous la forme d'un appel de LCGEOM, procédure principale, sans aucun argument, du module de lecture de la géométrie :

```
      CALL LCGEOM
```

Cette procédure est chargée de la lecture du fichier contenant le maillage au format MÉLINA. On trouvera dans le chapitre **Lecture des données** la description détaillée d'un fichier de maillage.

Procédures concernées, module lecgeom

LCGMAI Lecture du maillage proprement (procédure principale) ;

LCFORM Lecture des formats de lecture des coordonnées des points et de la numérotation globale des points ;

LCDESG Lecture de la description globale du maillage : noms des variables d'espace (définition de la dimension d'espace NDIM du COMMON/GLOBAL/ et création du tableau #ARIAB), nombre d'éléments (variable NBTEL du COMMON/GLOBAL/) ;

LCDESL Lecture de la description des blocs d'éléments du maillage (cf. mot-clé BLOC) et création de la structure #LOKEL ou de son ersatz du COMMON/GLOBAL/ (variables NBBBLK et NU1BLK) ;

RDELEM Lecture des données définissant un élément ;

RDCOOR Lecture des coordonnées des points d'un élément selon le format de lecture par défaut ou le format déclaré par directive, calcul de l'orientation de l'élément et construction de la structure #ORPTL, tableau de niveau 0, contenant les coordonnées des points du maillage distribuée élément par élément ;

RDNUGL Lecture de la numérotation globale des points d'un élément et construction de la structure #GNEEL contenant la numérotation globale des points du maillage, distribuées élément par élément.

LCNOMD Lecture des noms des domaines géométriques (construction du tableau des noms de domaine #OMDOM) ;

LCDOMA Lecture des constituants d'un domaine géométrique (construction d'une structure #ISTEL, dont le niveau est le numéro du domaine géométrique, i.e. le rang de son nom dans le tableau des noms de domaine #OMDOM) ;

LCELDO Cas d'un constituant *volumique*, i.e. d'un élément dont la dimension est égale à la dimension d'espace ;

LCFADO Cas d'un constituant de dimension inférieure à la dimension d'espace c'est-à-dire un domaine de bord : face, arête ou point.

1.2.2. Renumérotation des points du maillage

RENUME

Il s'agit d'un ensemble de procédures de renumérotation des numéros globaux des points du maillage afin d'optimiser la largeur de bande des matrices créées dans le module des calculs éléments finis. Cette renumérotation est effectuée par l'appel, par exemple dans le programme principal

```
      CALL RENUME (NIVNG1,NIVNG2,NIVIMP)
```

qui transforme la structure de numérotation #GNEEL de niveau NIVNG1 en la structure optimisée #GNEEL de niveau NIVNG2.

⚠ Il n'est actuellement possible que de renuméroter les **points du maillage**, et de ce fait on doit nécessairement avoir NIVNG1=NIVNG2 à l'appel. De plus, ce tableau de nom #GNEEL a pour niveau l'ordre d'interpolation géométrique (variable NIVENG du COMMON/GLOBAL/ du fichier CONTEX) égale à 1 pout un maillage P_1 ou Q_1, 2 pour un maillage P_2 ou Q_2, etc. La variable NIVNG1=NIVNG2 doit donc obligatoirement être égale à NIVENG.

Procédures concernées, module glonum

RINFOR recherche du degré maximum dans l'arbre de connexion des noeuds et initialisation ;
RALGOR algorithme de renumérotation proprement dit ;
RDLVOI relations de voisinage des noeuds ;
RLBAND calcul de la largeur de bande ;
RDLTRI classement des noeuds par numéro de degré croissant ;
RMOTOR moteur de renumérotation ;, marquage des nœuds par composante connexe ;
RMAJNU correspondance ancienne/nouvelle numérotation ;
RRANGE renumérotation des éléments.

1.2.3. Sauvegarde du maillage sur fichier

ECGEOM

Cette procédure permet de sauvegarder les données constitutives d'un fichier de maillage au format MÉLINA après qu'il ait été modifié géométriquement ou que la numérotation de ses points ait été optimisée. Ce transfert sur fichier est effectué par l'appel, par exemple dans le programme principal

```
      CALL ECGEOM (NOMFIC,TITRE)
```

où NOMFIC est une chaîne de caractères définissant le nom du fichier de sauvegarde et TITRE est le titre du fichier de maillage sauvegardé.

Procédures concernées, module lecgeom

LCFORM Ecriture des formats de lecture des coordonnées des points et de la numérotation globale des points ;
ECDESG Ecriture de la description globale du maillage : noms des variables d'espace (définition de la dimension d'espace NDIM du COMMON/GLOBAL/ et création du tableau #ARIAB), nombre d'éléments (variable NBTEL du COMMON/GLOBAL/) ;
ECDESL Ecriture de la description des blocs d'éléments du maillage (cf. mot-clé BLOC) et création de la structure #LOKEL ou de son ersatz du COMMON/GLOBAL/ (variables NBBBLK et NU1BLK) ;
ECELEM Ecriture des données définissant un élément (coordonnées et numérotation des points) ;
ECDOMA Ecriture des constituants d'un domaine géométrique (construction d'une structure #ISTEL, dont le niveau est le numéro du domaine géométrique, i.e. le rang de son nom dans le tableau des noms de domaine #OMDOM).

1.3. Définition des calculs & assemblage des termes

Elle consiste à définir par directives les calculs à effectuer sur les domaines créés dans la phase de lecture du fichier de maillage et de définir les structures conduisant à leur assemblage. Elle est réduite dans le cas les plus simples à l'appel de la procédure sans argument :

```
      CALL LCDIRE
```

Il s'agit de la procédure principale de lecture des directives ; voir la section **Déclaration des calculs sur les domaines**, du chapitre **Lecture des données par directives**.

Procédures concernées, module lecdire

LCINCO Lecture des noms et de caractéristiques des inconnues du problème ;
LCINTE Lecture du type d'interpolation d'inconnue de type valeurs nodales ;
LCSYME Lecture des (anti)symétries du problème ;
LCKDOM sous-procédure principale de lecture des calculs sur les domaines ;
LCQUAD lecture du type de la formule de quadrature pour le calcul des termes sur un domaine ;
LCTERM lecture du nom d'un terme à calculer sur un domaine et procédure *principale* de lecture des caractéristiques du terme ;
LCINCT lecture des noms des inconnues attachées à un terme ;
LCVANO lecture du type de condition essentielle pour un terme valeurs nodales ;
LCESSE lecture du type de condition essentielle pour un terme condition essentielle ;
LCTRAN lecture du type de condition pour un terme condition de transmission ;
LCINTG lecture du nom de l'intégrand correspondant à un terme éléments finis ;
LCDONT lecture de l'éventuelle donnée associée à un terme ;
LCCADO lecture des caractéristiques de la donnée associée à un terme ;
LCFLOC lecture des caractéristiques d'un terme éléments finis localisés ;
LCAUTR lecture du nom d'un terme (autre qu'éléments finis) et des inconnues qui s'y rattachent ;
LCASSE lecture des données d'assemblage d'un terme.

1.4. Lecture des (valeurs des) données

Il s'agit de la lecture des directives de définition de nouvelles données et/ou d'affectation de valeurs aux données de type constante ou tableau. Elle est réduite à l'appel de la procédure sans argument

```
CALL LCDONN
```

qui exploite les mots-clés de la directive DONNEE de création de nouvelles données.
Il s'agit de définir de nouvelles données non déjà associées aux termes définis par la directive CALCUL, ou d'affecter une valeur aux données de type constante éventuellement déjà définies (cf. syntaxe de la directive DONNEE du chapitre **Lecture des données**).

△ On verra plus loin qu'il est possible, par appel de procédures dans le programme principal, de créer de nouvelles données associées à des termes (cf. procédure de *haut niveau* DOASTR) et d'affecter une valeur à une donnée de type constante (cf. procédures PUTCST et GETCST).

Procédures concernées, module lecdire

LCCSTE Lecture des valeurs des données de type constante ;
LCCADO Lecture des caractéristiques de la donnée associée à une donnée de type fonction ;
INCST Affectation des valeurs des constantes dans les tableaux de valeurs des constantes $ACSTE, $ECSTE, $RCSTE et $CCSTE.

2. Calcul des termes Éléments Finis

Pré-assemblage

Les phases de l'exécution décrites précédemment ne sont que la partie préparatoire de la résolution proprement dite d'un problème (remplissage des structures de données pilotant les calculs). On verra qu'il s'agit d'un principe général de MÉLINA : on commence par créer les structures de données contenant toutes les informations nécessaires à une tâche donnée, puis dans un deuxième temps on effectue les calculs liés à cette tâche. Cette section concerne les calculs éléments finis eux-mêmes. Elle se rédige (ça se complique !) par l'appel de 3 procédures sans arguments dont l'action représente l'ensemble des calculs éléments finis :

```
CALL NUMNEU
CALL CPTDOM
CALL CALKEF
```

2.1. Numérotations globales et coordonnées des nœuds

NUMNEU module glonum

Le fichier d'entrée du maillage ne contient que des renseignements *géométriques*, en particulier aucune notion d'interpolation des (fonctions) inconnues n'est contenue dans ce fichier, et ainsi la numérotation globale des nœuds n'est pas définie à ce stade. De plus il est possible dans MÉLINA d'interpoler une inconnue selon une interpolation différente de celle de la géométrie (interpolation non isoparamétrique) ou encore de choisir des interpolations distinctes pour deux inconnues différentes (cas des méthodes d'éléments finis mixtes par exemple). Il s'en suit qu'il est nécessaire de créer les tableaux de numérotation globale et des coordonnées des nœuds pour toutes les interpolations éléments finis définies par directives (cf. Directive générale INCONNUE). C'est l'objet de NUMNEU procédure *principale* du module de numérotation globale glonum.

Procédures du module glonum

NUMNEU Procédure *de haut niveau* du module. Il est chargé de la recherche des adresses des structures utilisées et de la création des structures de sorties.

GLNUSO Numérotation globale des sommets.

GLNUFA Numérotation globale des arêtes des éléments du maillage, préalable à la numérotation des nœuds. Il est basé sur un algorithme de chaînage des arêtes du maillage que l'on trouvera dans la procédure CARET2 du module de maillage de MODULEF dû à P.L. GEORGE.

GLNUNE Création (pour chaque type d'interpolation) de la numérotation globale des nœuds à partir de la numérotation des faces ou arêtes.

GLNUNS Renumérotation globale des nœuds sommets d'un élément.

GLNUNA Renumérotation globale des nœuds des arêtes non-sommets d'un élément.

GLNUNF Renumérotation globale des nœuds des faces d'un élément non situés sur les arêtes.

GLCISO Création du tableau global des coordonnées pour l'interpolation isoparamétrique.

GLCNIS Idem pour les interpolations non isoparamétriques.

Structures de données d'entrée du module glonum : #LOKEL, #NCONU, #GNEEL, #ORPTL

Structures de données de sortie : #GNEEL, #ORNOE

Le module construit les tableaux #GNEEL et #ORNOE dont les niveaux sont donnés par les numéros de type de toutes les interpolations utilisées pour les inconnues du problème.

2.2. Distribution des coordonnées des points par domaine

`CPTDOM module cordom`

Les calculs éléments finis étant effectués successivement pour chacun des domaines de calcul, ce module est chargé de créer les structures de coordonnées des points `#ORPTL` pour chaque domaine (ces tableaux sont différenciés par leur niveau qui est égal au numéro du domaine de calcul correspondant). Ces tableaux sont construits à partir du tableau `#ORPTL` de niveau 0 contenant les coordonnées des points de tous les éléments du maillage. Ils contiennent les coordonnées des points des éléments du domaine pour un domaine de dimension égale à la dimension d'espace (domaine *volumique*), des points des faces ou des arêtes des éléments pour un domaine de dimension inférieure à la dimension d'espace.

Procédures du module cordom

`CPTDOM` Procédure principale, i.e. *de haut niveau*.
`CODOVO` Constitution d'une structure `#ORPTL` pour un domaine volumique.
`CODOFA` Idem mais pour les domaines de bords.
`SACHA` `DIST`ribution des coordonnées des points pour un `EL`ément ou pour une face ou une arête.

Structures de données d'entrée du module cordom : `#LOKEL`, `#ORPTL`

Structures de données définies : `#ORPTL`

Le niveau d'une telle structure est le numéro du domaine de calcul sur lequel elle construite.

2.3. Calculs Éléments Finis sur les domaines

`CALKEF module caldom`

Il s'agit du plus gros ensemble de procédures appelé par une procédure de haut niveau. Il utilise pratiquement toutes les librairies de MÉLINA. Les calculs de tous les termes simples (intégrales de la formulation variationnelle, termes valeurs nodales et termes condition essentielle) y sont effectués. L'arborescence du module de calculs éléments finis est donné sur la figure 3.

2.3.1. Numérotations globales par domaine

`NUMDOM module caldom`

Création[3] des numérotations globales des nœuds de l'interpolation en colonne (et le cas échéant en ligne) de tous les termes d'un domaine de calcul.

Procédures de numérotations du module caldom

`NUMDOM` Procédure principale.
`NUNEDO` Constitution d'une structure `#GNEDO` pour les nœuds d'un domaine de calcul.
`RGNEDO` Constitution d'une structure `#RGNDO` pour un domaine. Contient pour tous les éléments d'un domaine les rangs des nœuds de lélément dans la numérotation globale du domaine. Plus précisement, en assimilant les structures `#GNEDO` et `#RGNDO` à des tableaux fortran, `'#GNEDO'('#RGNDO'(I,K))` désigne le numéro global du nœud (pour l'interpolation de l'inconnue idoine) du nœud `I` (numérotation locale) de l'élément `K`.

Structures de données définies : `#GNEDO`, `#RGNDO`

Lors de sa création le niveau d'une structure `#GNEDO` est fourni par un appel à la procédure `KLNIVE`, librairie sdexplo qui retourne le prochain numéro de structure disponible ; ce niveau est dans la suite un des attributs du domaine et des termes qui y sont calculés. Le niveau de la structure `#RGNDO` est le numéro du domaine de calcul.

(3) Les calculs ne sont évidement effectués qu'une fois pour chaque domaine de calcul. Si ces procédures sont appelées dans une boucle, le programme garde la mémoire des calculs déjà effectués.

```
NUMDOM NUNEDO                                          numérotation des nœuds
       RGNEDO
CALNOR REPLOC                                          calcul des normales aux nœuds
       ASNORM
DSKNIV INTATR
DSKBMO BMPACT CMPLGN                                   calcul des tables de pointeurs
DSKMLI CMPACT CMPLGN
DSKMCO CMPACT CMPLGN
DSKMLM
TRMDOM INTATR                                          création des termes éléments finis
VNQLTR VNATRI
       VNCALC FCTRM
ELQLTR EFATRI INTTAY INTATR
              EFRPFB
       EFTBAD EFADTB
       EFADRS
       EFVNLO
       CALELM ELEMQD HEXAQD
                    /PRISQD/TETRQD/QUADQD/TRIAQD/SEGMQD
              ECOBAS ELEMFB HEXAFB
                           /PRISFB/TETRFB/QUADFB/TRIAFB/SEGMFB
              DELAFO
              OFALED
              TRANSF
              CALINT FCTRM                    intégrales élémentaires de 'volume'
                     W      /WW     /GRAGRA/ROTROT/DIVDIV etc.
       CALELB ELEMQD etc.
              ECOBAS etc.
              DELAFO
              ELDIFF PROSCA /PROEXT
              TRANSF
              CALINB FCTRM                     intégrales élémentaires de bord
                     W      /WWISCA/WWISCA/GRTGRT/WNWN    etc.
       CALELD ELEMQD etc.
              ELEMQF HEXAQF /HEXAQA
                     PRISQF /PRISQA/TETRQF/TETRQA/QUADQA/TRIAQA
              ECOBAS etc.
              DELAFO
              ELEMTF HEXATF
                    /PRISTF/TETRTF/QUADTF/TRIATF
              OFALED
              ELDIFF PROSCA /PROEXT
              TRANSF
              CALIND FCTRM       intégrales élémentaires de bord avec dérivées
                     DW     /WDW    /DNW    /GRVGRV/NDIVWW etc.
       PTRBML/PTRBMU/PTRMLI/PTRMCO                     tables d'assemblage
              DIKLPL
       PASTDO INTATR                    assemblage des termes éléments finis
              PASLBM /PASUBM/PAVLBM/PAVUBM
              PASMLI /PAVMLI
              PASMCO /PAVMCO
       NASTDO INTATR                      termes éléments finis non-assemblé
SAVTRM                                   sauvegarde des termes éléments finis
```

Figure 3. Schéma simplifié de l'arborescence des calculs éléments finis

2.3.2. Calculs géométriques par domaine

`CALNOR` module `caldom`

Création[(4)] des tableaux des normales assemblées aux nœuds sur un domaine de calcul de bord

(4) Les calculs ne sont évidement effectués qu'une fois pour chaque domaine de calcul. Si ces procédures sont appelées dans une boucle, le programme garde la mémoire des calculs déjà effectués.

(voir la sous-directive NORMALES).

Procédures de calcul des normales assemblées, librairie calelem

ASNORM Procédure principale.
ASNORM Assemblage des normales aux nœuds du domaine.
REPLOC Calcul des normales élélementaires aux nœuds d'un élément.

Structure de donnée définie : &NORMA
Le niveau de ces tableaux (cf. ci-dessus) est un des attributs du domaine et des termes qui y sont calculés.

2.3.3. Compactage des matrices par domaine
DSKNIV, DSKBMO, DSKMLI, DSKMCO module caltrm

Création(4) des pointeurs de compactage des termes matriciels définis sur un domaine de calcul.

Procédures : DSKNIV, DSKBMO, DSKMLI, DSKMCO et BMPACT, CMPACT et CMPLGN de la librairie morse

Structures de données définies : &BMORS, &PLAGE, &EGALP
Le niveau de ces tableaux (cf. ci-dessus) est un des attributs du domaine et des termes qui y sont calculés.

2.3.4. Calculs élémentaires et pré-assemblage par domaine
TRMDOM, EFQLTR, PASTDO, NASTDO module caltrm

Calcul des matrices élémentaires éléments finis, pré-assemblage séparé des termes éléments finis (procédures CALELM, CALELB, CALELD et CALINT, CALINB et CALIND, librairie calelem) et assemblage séparé des termes (PASTDO) ou création des termes matrice elementaire (NASTDO).

Librairies utilisées : calelem, intégran, ef3d (ou ef2d)

Structures : Les termes calculés sur le domaine de calcul

2.3.5. Calculs des termes valeurs nodales
VNQLTR, VNCALC module caltrm

Calcul des termes valeurs nodales et des termes de condition essentielle ou de transmission (procédures VNQLTR et VNCALC).

Structures : Les termes calculés sur le domaine de calcul

2.3.6. Sauvegarde des termes par domaine
SAVTRM module caldom

Il s'agit de la mise en état de sauvegarde différée (TBSAVE) de tous les termes calculés sur les domaines.

Structures : Les termes calculés sur le domaine de calcul

À ce stade de l'exécution, tous les termes dont le calcul a été demandé par directive ou par procédure, les tableaux de coordonnées et des normales, ont été calculés et sauvegardés si nécessaire. Il en est de même pour toutes les structures (numérotations globales, pointeurs de compactage, etc.) associées aux domaines de calcul et aux termes.
Il reste donc pour résoudre le problème à assembler ces termes, calculer d'autres termes propres à la méthode utilisée (par exemple termes de couplage éléments finis–représentation intégrale, termes éléments finis localisés), et à définir l'algorithme qui conduit à sa solution.

En résumé, dans les cas les plus simples, le programme principal peut se réduire à ce stade à [•] :

```
      PROGRAM XXXXXX
*
      PARAMETER ( MAXAST = 80000 , MAXIST= 8000000 )
      COMMON/STCHAR/AST(MAXAST)   /      /IST(MAXIST)
*
*     Initialisation des tables de l'allocation dynamique
*     et creation des structures de base
      CALL INITIE (MAXAST,MAXIST)
*     Lecture du fichier de maillage
      CALL LCGEOM
*     Lecture des donnees par directives
      CALL LCDIRE
*     Lecture des valeurs des donnees 'constantes'
      CALL LCDONN
*     Numerotation globales pour toutes les interpolations
      CALL NUMNEU
*     Distribution des coordonnees des points du maillage par domaine
      CALL CPTDOM
*     Calcul 'elements Finis' (Pre-assemblage)
      CALL CALKEF
*
      .... La suite releve du developpement de l'application
      .... et de la programmation de l'algorithme de resolution
*
      CALL BYE ('C''est fini.')
      END
```

[•] Les valeurs de `MAXAST` et de `MAXIST` ne sont données qu'à titre d'exemple.

⚠ Rappelons cependant, que selon la norme Fortran77, la longueur `MAXAST` est définie dans la procédure `CONTEX` et doit être la même ici du `COMMON/STCHAR/AST(MAXAST)`.

– Mai 2000 –

3. Description des sous-programmes de haut niveau

On trouvera ci-dessous une liste non-exhaustive des **procédures de haut niveau** permettant de définir la phase suivante (algorithmique) du programme principal. Leur utilisation et ordonnancement relève du problème à résoudre et de l'algorithme de résolution à mettre en œuvre. Une description plus complète figure dans les documentations des librairie sdexplo, module assembl et module couplag.

△ *Dans la liste des paramètres formels des procédures les arguments d'entrée non modifiés, qui peuvent donc être des constantes, apparaissent en caractères* TYPEWRITER TYPE *standards, les arguments modifiés ou les arguments de sortie, qui doivent donc être des variables, en caractères* *TYPEWRITER TYPE penchés*.

3.1. Valeurs des données de type constante

GETCST, PUTCST librairie sdexplo

Ces procédures permettent de chercher la valeur d'une donnée de type constante ou d'affecter une valeur à une donnée de type constante. La définition précise de leurs arguments est donnée dans la documentation de la librairie sdexplo.

• SUBROUTINE GETCST (NOMCST,*TYPCST*,*ENTIER*,*REEL*,*COMPLE*,*CARAC*)

Cette procédure permet de rechercher la valeur d'une donnée constante définie par son nom NOMCST ; la valeur est retournée, selon le type (TYPCST) de la constante, dans la variable ENTIER, REEL, COMPLE ou CARACT.

⚠ La constante doit évidement avoir été définie, soit par directive, soit à l'aide de la procédure qui suit.

⚠ Remarquer ici que *TYPCST* est un argument de sortie fourni à l'utilisateur pour un éventuel test ; **on prendra soin d'y placer une variable de type** chaîne de caractères**.**

⚠ Dans le cas d'une constante de type chaîne de caractères, on prendra soin de s'assurer d'une longueur suffisante pour l'argument–chaîne récepteur CARAC ; dans le cas d'une longueur insuffisante la valeur de la constante chaîne sera tronquée.

• SUBROUTINE PUTCST (NOMCST,TYPCST,ENTIER,REEL,COMPLE,CARAC)

Cette procédure permet l'introduction ou la modification de la valeur de la donnée de type constante définie par son nom NOMCST et déclarée de type TYPCST. Si la donnée constante n'existe pas, elle est créée dans le tableau des noms de données #OMDON et la valeur affectée est introduite dans le tableau de constantes du type adéquat ($ECSTE, $RCSTE, $CCSTE, $ACSTE). La valeur de la constante est introduite à travers l'argument d'entrée (ENTIER,REEL,COMPLE,CARAC) correspondant au type déclaré de la donnée.

⚠ Si la donnée constante existe déjà, son type de déclaration ne peut être redéfini dans cette procédure (c'est l'ancien type qui prévaut ici).

Exemple 1 Affectation, et le cas échéant création, de la constante réelle $1/4\pi$

```
      CALL PUTCST ('1/4Pi','REEL',ENTIER,1./(16.*ATAN(1.)),COMPLE,CARACT)
```

Exemple 2 Affectation de données constantes à partir d'une autre, définition de la constante réelle $-k^2$ et complexe $-ik$ à partir de la constante réelle k

```
      CALL GETCST ('k',TYPE,ENTIER,REEL,COMPLE,CARACT)
      CALL PUTCST ('-k**2','REEL',ENTIER,-REEL*REEL,COMPLE,CARACT)
      CALL PUTCST ('-ik','COMPLEXE',ENTIER,REEL,CMPLX(0.,-REEL),CARACT)
```

Les variables ENTIER, REEL, COMPLE, CARACT et TYPE sont supposées déclarées de manière idoine. L'appel de GETCST fournit la valeur de k, constante réelle de nom 'k' dans l'argument REEL. Les appels de PUTCST affecte une valeur aux constantes réelle $-k^2$ de nom '-k**2' et complexe $-ik$ de nom '-ik'.

Exemple 3 Calcul et utilisation d'un incrément de la valeur d'une constante

On suppose que l'on s'est donné deux valeurs réelles 'kmin' et 'kmax' et un nombre entier 'nbk' et que l'on souhaite boucler sur les valeurs de 'k' équiréparties sur l'intervalle $[k_{min}, k_{max}]$:

```
*     Calcul de l'increment Deltak
      CALL GETCST ('kmin',TYPE,ENTIER,RKMIN,COMPLE,CARACT)
      CALL GETCST ('kmax',TYPE,ENTIER,RKMAX,COMPLE,CARACT)
      CALL GETCST ('nbk',TYPE,NBK,REEL,COMPLE,CARACT)
      DELTAK=(RKMAX-RKMIN)/NBK
*     Valeur initiale de 'k' : celle de 'kmin'
      RK=RKMIN
      CALL PUTCST ('k','REEL',ENTIER,RK,COMPLE,CARACT)
      ...
*     Boucle sur les valeurs de 'k'
      DO 1000 K=1,NBK
         ...
         ... Traitement pour la valeur courante de k
         ...
*        Valeur suivante de k
         CALL PUTCST ('k','REEL',ENTIER,RK+DELTAK,COMPLE,CARACT)
         ...
1000  CONTINUE
```

3.2. Valeurs des coefficients d'un tableau

PUTTAB, GETTAB librairie sdexplo

Ces procédures permettent d'affecter une valeur à un des coefficients d'un tableau défini ou non comme un terme ou de rechercher la valeur d'un coefficient d'un tableau. La définition précise de leurs arguments est donnée dans la la documentation de la librairie sdexplo.

△ Voir aussi les procédures PUTERM/INTERM et GETERM du module assembl lorsque que le tableau est un terme.

• SUBROUTINE PUTTAB (NOMTRM,NIVTRM,KLADRS,TYPTAB,ENTIER,REEL,COMPLE,CARACT)

Affectation de la valeur entière ENTIER, réelle REEL, complexe COMPLE ou chaîne de caractères CARACT selon le type TYPTAB du terme au coefficient d'adresse relative KLADRS du tableau de nom NOMTRM et de niveau NIVTRM.

• SUBROUTINE GETTAB (NOMTRM,NIVTRM,KLADRS,*TYPTAB,ENTIER,REEL,COMPLE,CARACT*)

Recherche de la valeur entière, réelle, complexe ou chaîne de caractères du coefficient d'adresse relative KLADRS du tableau de nom NOMTRM et de niveau NIVTRM. Selon le type de déclaration du terme la valeur est retournée dans l'argument ENTIER (cas entier), REEL (cas réel), COMPLE (cas complexe) ou CARACT (cas chaîne de caractères).

3.3. Affectation d'une donnée à un terme

DOASTR librairie sdexplo

Cette procédure permet d'affecter une donnée à un terme. Les arguments parlent d'eux-mêmes mais leurs définitions précises sont données dans la documentation de la librairie sdexplo.

• SUBROUTINE DOASTR (NOMTRM,NIVTRM,NOMDON,TYPDON,NIVDON,TYPEDO)

Cette procédure permet d'affecter une donnée à un terme ou de modifier (le nom de) la donnée affectée à un terme : c'est la version 'procédure' de la sous-directive DONNEE des directives de calculs des termes sur les domaines. Le type de représentation de la donnée est défini par TYPDON (constante, fonction ou tableau), le type de déclaration par TYPEDO (caractère, entier, reel, complexe et NIVDON est le niveau du tableau dans le cas d'une donnée de type tableau (il est ignoré dans autres cas).

⚠⚠ Lors de l'appel de cette procédure, le terme (NOMTRM, NIVTRM) doit nécessairement exister. La constante peut ou non avoir déjà été définie par ailleurs (par directive générale DONNEE ou procédure PUTCST, par exemple).

Exemple 1 Affectation de la constante '-k**2' au terme 'UV'$= \int_\Omega u\,v\,d\omega$ (de niveau 1) pour obtenir le terme $-k^2 \int_\Omega u\,v\,d\omega$ lors de l'assemblage des termes.

```
CALL DOASTR ('UV',1,'-k**2','CONSTANTE',0,'REEL')
```

Exemple 2 Affectation de la donnée fonction 'G' au terme 'NEUGAM' de niveau 1 pour obtenir le terme $\int_\Gamma g\,v\,d\gamma$ lors des calculs éléments finis.

```
CALL DOASTR ('NEUGAM',1,'G','FONCTION',0,'REEL')
```

3.4. Association d'une donnée à une donnée fonction

DOASFC librairie sdexplo

Cette procédure permet d'associer une donnée de type constante ou tableau à une donnée de type fonction. Les arguments parlent d'eux-mêmes mais leurs définitions précises sont données dans la documentation de la librairie sdexplo.

• SUBROUTINE DOASFC (NOMFON,NOMDON,TYPDON,NIVDON,TYPEDO)

Cette procédure permet d'associer une donnée de nom NOMDON à une donnée de type fonction de nom NOMFON. Le type de représentation de la donnée est défini par TYPDON (constante ou tableau), le type de déclaration par TYPEDO (caractère, entier, reel, complexe et NIVDON est le niveau du tableau dans le cas d'une donnée de type tableau ; c'est la version 'procédure' de la sous-directive DONNEE ASSOCIEE de la commande DONNEE des sous-directives TERME de la directive générale CALCUL SUR LE DOMAINE.
Elle permet de 'transmettre' à une donnée de type fonction un ou plusieurs paramètres nécessaire à son évaluation : par exemple le calcul de la fonction
— $\phi_{w_1} = \mathrm{e}^{ikx}$ nécessite une donnée de type constante k ;
— $\phi_{w_2} = \mathrm{e}^{ik(x\cos\theta - y\sin\theta)}$ nécessite une donnée de type tableau pour transmettre les valeurs de k et θ.

△ Dans le cas d'une donnée de type tableau associée à une fonction, le contenu du tableau est entièrement à la discrétion du développeur de l'application, le code se charge seulement de transmettre son adresse et son type (cf. la procédure *'utilisateur'* SUBROUTINE FCTRM). Un tel tableau peut être "rempli" par directives (DONNEE) et on verra plus loin qu'il est aussi possible de "remplir" un tableau à l'aide de procédures de haut niveau (PUTTAB, PUTERM, CSTERM).

⚠ Si la donnée existe déjà, ses types de représentation et de déclaration ne peuvent être redéfinis dans cette procédure (les anciens types prévalent ici).

Exemple 1 Association de la constante 'k' à la fonction 'PHIw1'

```
CALL DOASFC ('PHIw1','k','CONSTANTE',0,'REEL')
```

Exemple 2 Association de la donnée tableau 'kteta' de niveau 5 à la fonction 'PHIw2'

```
      CALL DOASFC ('PHIw2','kteta','TABLEAU',5,'REEL')
```

3.5. Demande du calcul ou du recalcul de termes

MKTERM, RKTERM librairie sdexplo

Il s'agit de procédures de réactualisation des structures #TERDO, #OMTRM et $SDTRM en vue de leur (ré-)exploitation par le module de calcul des termes éléments finis et valeurs nodales (procédure CALKEF module caldom). Il s'agit simplement de (re)demander le calcul des termes ; **ces procédures ne déclenchent pas le calcul effectif des termes**. Le calcul de termes définis par ces procédures (ou par directives) est effectué par appel à la procédure CALKEF qui peut être appelée autant de fois que voulu. Il est conseillé néanmoins de regrouper le calcul effectif de ces termes en utilisant successivement les appels à MKTERM ou RKTERM avant de lancer la procédure CALKEF.

- SUBROUTINE MKTERM (NOMTRM,NIVTRM,TYPTRM,NOMDOM,TYPCAL,NMKALE,NUSCHQ
  ,NOMINC,NOMINL,NIVIMP)

Cette procédure peut être utilisée comme substitut à la directive de définition de termes à calculer sur les domaines. Ses arguments correspondent tous à l'un des mots-clés de la sous-directive TERME de la directive CALCUL SUR LE DOMAINE (voir le chapitre **Lecture des données**).

Description des arguments, voir documentation de la librairie sdexplo

⚠ Le terme dont on demande le calcul ici peut déjà avoir été défini soit par directive, soit par cette procédure même, et peut avoir été déjà calculé. Cependant dans ce cas un nombre important de ses caractéristiques ne peuvent être redéfinies ; on ne peut ainsi modifier les paramètres de calcul suivants : type de déclaration (TYPTRM = réel, complexe), nom du domaine géométrique NOMDOM, type de calcul du terme (TYPCAL = elfini, matelm, valnod, c.esse, c.tran), noms des inconnues associées (NOMINC, NOMINL), numéro du schéma de quadrature (NUSCHQ si TYPCAL = 'elfini' ou 'matelm'). Il est donc recommandé pour redemander le calcul d'un terme d'utiliser la procédure RKTERM qui suit.

Exemple 1 Demande du calcul des termes éléments finis

RIGID $\overset{déf}{\approx} \int_\Omega \overrightarrow{\text{grad}}\,\phi\ \overrightarrow{\text{grad}}\,\psi\, d\omega$,

MASSE $\overset{déf}{\approx} \int_\Omega \phi\psi\, d\omega$, constante affectée $-k^2$

GDELTA $\overset{déf}{\approx} \int_\Sigma \phi\psi\, d\sigma$, constante affectée $-ik$,

NEUGAM $\overset{déf}{\approx} \int_\Gamma g\psi\, d\gamma$, donnée fonction affectée g, la constante k étant associée à la donnée g :

```
      CALL MKTERM
     &('RIGID',1,'REEL','Omega','ELFINI','GRADGRAD',2,'PHI','PHI',0)
      CALL MKTERM
     &('MASSE',1,'REEL','Omega','ELFINI','UV',2,'PHI','PHI',0)
      CALL MKTERM
     &('GDELTA',1,'REEL','Sigma','ELFINI','UV',2,'PHI','PHI',0)
      CALL MKTERM
     &('NEUGAM',1,'REEL','Gamma','ELFINI','V',2,'PHI','PHI',0)
*     Association des donnees aux termes
      CALL DOASTR ('MASSE',1,'-k**2','CONSTANTE',0,'REELLE')
      CALL DOASTR ('GDELTA',1,'-ik','CONSTANTE',0,'COMPLEXE')
      CALL DOASTR ('NEUGAM',1,'G','FONCTION',0,'COMPLEXE')
      CALL DOASFC ('G','k','CONSTANTE',0,'REELLE')
```

• SUBROUTINE RKTERM (NOMTRM,NIVTRM,NIVIMP)

Le lecteur quelque peu attentif aura remarqué que pratiquement aucun des attributs d'un terme fournis comme arguments de la procédure MKTERM n'est modifiable ; cette seconde procédure permet de redemander le calcul de ce terme avec les même attributs. Ce recalcul n'est évidement nécessaire que lorsqu'il conduit à un résultat différent d'un calcul antérieur ; cette circonstance se produit en particulier lorsque une donnée fonction ou tableau associée à ce terme ne fournit pas les mêmes résultats qu'antérieurement (cas courant), ou encore (cas extrême) lorsque la géométrie a changée (problème de surface libre par exemple).

Exemple Re-demande du calcul du terme NEUGAM si la valeur de la constante k utilisée dans le calcul de la fonction g a changé.

```
      CALL RKTERM ('NEUGAM',1,0)
```

⚠ Le terme dont on redemande le calcul doit évidement avoir été défini précédemment !

3.6. Demande de recalcul de tous les termes

RKGEOM librairie sdexplo

Dans le cas où l'on modifie la géométrie du problème (estimations à posteriori d'erreur, raffinement de maillage) il est nécessaire de remettre en cause tous les calculs effectués. L'appel (sans arguments)

```
      CALL RKGEOM ( )
```

permet un *reset* de toutes les structures de calcul des termes déclarés sur les domaines de calculs.

3.7. Valeurs des coefficients d'un terme

CSTERM, INTERM, OUTERM, PUTERM, GETERM, MXTERM module assembl

Ces procédures permettent d'affecter une valeur à un ou à tous les éléments d'un tableau défini comme un terme ou de rechercher la valeur d'un élément d'un tableau. La définition précise de leurs arguments est donnée dans la la documentation du module assembl.

• SUBROUTINE CSTERM (NOMTRM,NIVTRM,NOMCST)

Affectation de la valeur de la donnée constante de nom NOMCST à tous les éléments du terme de nom NOMTRM et de niveau NIVTRM.

⚠ La constante doit être de type reel ou complexe.

• SUBROUTINE INTERM (NOMTRM,NIVTRM,KLADRS,NOMCST)

Affectation de la valeur de la donnée constante NOMCST au coefficient d'adresse relative KLADRS du tableau défini comme le terme de nom NOMTRM et de niveau NIVTRM.

⚠ La constante doit être de type reel ou complexe.

• SUBROUTINE OUTERM (NOMTRM,NIVTRM,KLADRS,NOMCST)

Recherche de la valeur du coefficient d'adresse relative KLADRS du tableau représenté par le terme de nom NOMTRM et de niveau NIVTRM, et affectation de cette valeur à la donnée constante NOMCST.

• SUBROUTINE PUTERM (NOMTRM,NIVTRM,KLADRS,REEL,COMPLE)

Affectation de la valeur réelle REEL ou complexe COMPLE selon le type du terme au coefficient d'adresse relative KLADRS du tableau défini comme le terme de nom NOMTRM et de niveau NIVTRM.

• SUBROUTINE GETERM (NOMTRM,NIVTRM,KLADRS,*REEL*,*COMPLE*)

Recherche de la valeur réelle ou complexe du coefficient d'adresse relative KLADRS du tableau représenté par le terme de nom NOMTRM et de niveau NIVTRM. Selon le type de déclaration du terme la valeur est retournée dans l'argument REEL (cas réel) ou COMPLE (cas complexe).

• SUBROUTINE MXTERM (NOMTRM,NIVTRM,*KLADRS*,*REEL*,*COMPLE*)

Retourne l'adresse *KLADRS* et la valeur *REEL* (resp.*COMPLE*) du premier coefficient ayant la valeur absolue (resp. le module) maximum parmi les coefficients du terme de nom NOMTRM et de niveau NIVTRM.

3.8. Assemblage des termes

Le module de calcul éléments finis fournit les termes matriciels ou vectoriels (par exemple de la formulation variationnelle) assemblés indépendamment les uns des autres. On obtient ainsi après exécution de ce module des matrices et vecteurs ayant chacun leurs structures de numérotation et de pointeurs de stockage propres. L'assemblage (par exemple d'un système linéaire correspondant à la formulation variationnelle discrétisée) consiste essentiellement à combiner linéairement des termes ou à construire des termes vectoriels par produit matrice×vecteur.

3.8.1. Combinaison linéaire ou non de termes
CLTERM, AXTERM, T2TERM module assembl

• SUBROUTINE CLTERM (NOMTR1,NIVTR1,NOMCS1,NOMTR2,NIVTR2,NOMCS2
,NOMTR3,NIVTR3,NIVIMP)

Reporte la combinaison linéaire :

"(NOMTR1,NIVTR1)*NOMCS1 + (NOMTR2,NIVTR2)*NOMCS2"

dans le terme NOMTR3,NIVTR3. Si le nom d'une constante est vide 'NOMCSi=' ', la multiplication est effectuée avec la constante affectée au terme 'i' si une telle affectation existe, avec la constante 1 sinon.

△ Des informations complémentaires, concernant les caractéristiques du terme résultat, peuvent être trouvées dans la documentation du module assembl.

• SUBROUTINE AXTERM (NOMTR1,NIVTR1,NOMCST,NOMTR2,NIVTR2,NIVIMP)

Reporte le terme '1' multiplié par la constante NOMCST dans le terme '2' avec les mêmes caractéristiques (structure $SDTRM), excepté le type, le numéro de donnée affectée (le nouveau terme n'a pas de donnée affectée). Si le nom de la constante est vide 'NOMCST=' ', la multiplication est effectuée avec la constante affectée au terme '1' si une telle affectation existe.

• SUBROUTINE T2TERM (NOMTR1,NIVTR1,TRANS,NOMTR2,NIVTR2,NOMTR3,NIVTR3,NIVIMP)

Combinaison de 2 termes ('1' et '2') reportée dans le terme '3' ; l'argument chaîne de caractères TRANS permet de choisir l'opération définissant la combinaison :

- ▷ "+" somme des 2 termes : '3'='1'+'2'.
- ▷ "−" différence des 2 termes : '3'='1'−'2'.
- ▷ "CL" combinaison des 2 termes : '3'= c_1.'1'+c_2.'2', où c_i est la constante affectée au terme i (1. par défaut).
- ▷ "DIFFERENCE RELATIVE" différence relative : '3'$_\ell$ = 2|'1'$_\ell$ - '2'$_\ell$| / |'1'$_\ell$ + '2'$_\ell$|
- ▷ "PRODUIT TENSORIEL" produit tensoriel des 2 termes : '3'$_{\ell,k}$ = '1'$_\ell$. '2'$_k$.
- ▷ "PRODUIT COMPOSANTE A COMPOSANTE" produit coefficient par coefficient des 2 termes : '3'$_\ell$ = '1'$_\ell$. '2'$_\ell$.

3.8.2. Assemblage de termes matriciels
ASMAKE, DSMAKE librairie sdexplo, ASMTRM, DSMTRM module assembl

La combinaison linéaire de termes matriciels (c'est-à-dire auxquels sont affectées une inconnue en colonne et une inconnue en ligne) est divisé en deux étapes : création d'une structure dite

d'assemblage (répondant au doux nom '$ASMBL') qui permet de déclarer les différents termes à combiner, puis assemblage proprement dit.

Il existe deux types d'assemblage selon que les termes à combiner possèdent ou non le même couple d'inconnues affectées.
– dans le premier cas, baptisé 'assemblage uni-inconnue', **tous** les termes doivent avoir le même couple d'inconnues en colonne et ligne (l'inconnue en colonne et l'inconnue en ligne peuvent bien sûr différer).
– dans le second cas, baptisé 'assemblage multi-inconnues', deux termes à assembler doivent nécessairement avoir des couples d'inconnues distincts.

⚠ On prendra donc soin d'assembler dans un premier temps les termes résultats d'assemblage uni-inconnue qui entre dans la constitution d'une structure d'assemblage multi-inconnues.

△ Lorsque les termes à assembler sont stockés selon une même structure, par exemple s'ils sont calculés sur le même domaine de calcul, les opérations d'assemblage peuvent être avantageusement remplacées par des combinaisons linéaires : cf. procédures CLTERM, T2TERM.

Construction de la structure d'assemblage d'un terme

Il s'agit seulement de définir le nom (et niveau) du terme résultat de la combinaison linéaire des différents termes et de donner la liste des termes entrant dans la combinaison linéaire. Cette déclaration de la liste des termes peut se faire soit par directive (voir la directive ASSEMBLAGE dans le chapitre **Lecture des Données**), soit par appel successif à l'une des procédures de création ou d'enrichissement d'une structure d'assemblage ; dans ce dernier cas, on utilise autant de fois qu'il est nécessaire l'une des procédures suivantes selon que l'assemblage est uni- (préfixe AS) ou multi-inconnues (préfixe DS) :

- `SUBROUTINE ASMAKE (NOMTRM,NIVTRM,NOMTRA,NIVTRA)`
- `SUBROUTINE DSMAKE (NOMTRM,NIVTRM,NOMTRA,NIVTRA)`

qui crée la structure $ASMBL, contenue dans le tableau de même nom et dont le niveau est une des caractéristiques du terme assemblé, pour le terme assemblé de nom NOMTRA et de niveau NIVTRA si elle n'existe pas déjà, et enrichi cette structure du numéro du terme NOMTRM de niveau NIVTRM élément de la combinaison linéaire.

Exemple 1 Construction de la structure d'assemblage de la forme bilinéaire

$$a(u,v) = \int_\Omega \nabla u \, \nabla v \, dx - k^2 \int_\Omega u \, v \, dx - ik \int_\Sigma u \, v \, d\sigma$$

```
CALL ASMAKE ('RIGID',1,'ADEUV',1)
CALL ASMAKE ('MASSE',1,'ADEUV',1)
CALL ASMAKE ('GDELTA',1,'ADEUV',1)
```

Exemple 2 Construction de la structure d'assemblage de la forme bilinéaire

$$a(\vec{u},\vec{v};p,q) = \begin{pmatrix} \nu \int_\Omega \overrightarrow{\text{grad}}\,\vec{u} \cdot \overrightarrow{\text{grad}}\,\vec{v}\, dx & -\int_\Omega p \ \text{div}\,\vec{v}\, dx \\ \int_\Omega \text{div}\,\vec{u}\, q\, dx & 0 \end{pmatrix}$$

```
CALL DSMAKE ('RIGID',1,'ADUVPQ',1)
CALL DSMAKE ('PDIVV',1,'ADUVPQ',1)
CALL DSMAKE ('DIVUQ',1,'ADUVPQ',1)
```

△ Il n'est pas nécessaire que les termes constituants de l'asemblage soit définis lors de la création de la structure d'assemblage.

⚠ Compte-tenu de la complexité des calculs d'assemblage, le terme assemblé ne peut être l'un des termes composant la combinaison linéaire.

⚠ ⚠ Lorsque dans un processus itératif, si un terme doit être assemblé toujours selon la même structure d'assemblage (i.e. résulte de l'assemblage du même ensemble de termes), il est inutile de redéfinir sa structure d'assemblage. La redéfinition de la structure d'assemblage implique dans le code 'l'oubli' de sa structure d'assemblage précédente et la création d'une nouvelle, et par suite le recalcul des différents pointeurs d'assemblage, qui peu, lors d'un nombre elevé d'itérations, provoquer la saturation des super-tableaux. Il est toujours préférable, lorsque cela est possible, de déclarer les structures d'assemblage à l'aide de la directive ASSEMBLAGE.

Assemblage proprement dit d'un terme matriciel

Il s'agit seulement de déclencher cette opération d'assemblage du terme au moment opportun, c'est-à-dire quand tous les termes de la combinaison ont été calculés ; l'assemblage est réalisé par l'apel à l'une des procédures :

- `SUBROUTINE ASMTRM (NOMTRA,NIVTRA,NIVIMP)`
- `SUBROUTINE DSMTRM (NOMTRA,NIVTRA,NIVIMP)`

selon le type d'assemblage (uni- ou multi-inconnues).

⚠ C'est seulement lors de ces calculs de combinaison linéaire de termes que sont prises en compte les données de type constante affectées aux termes ; ces constantes sont les coefficients affectés aux termes de la combinaison linéaire.

Exemple 1 Calcul de la forme bilinéaire $a(u, v)$

```
      CALL ASMTRM ('ADEUV',1,0)
```

On peut difficilement imaginer plus simple.

3.8.3. Assemblage d'un terme vectoriel

`ASMAKE, DSMAKE librairie sdexplo, ASVTRM, DSVTRM module assembl`

La création d'une structure d'assemblage pour un terme vectoriel (c'est à dire d'un terme auquel est affectée une seule inconnue) assemblé est rigoureusement identique au cas d'un terme matriciel, par directive ou par appels successifs aux procédures ASMAKE et DSMAKE avec la même contrainte que celle indiquée plus haut. L'assemblage proprement dit d'un terme est déclenché par l'une des procédures

- `SUBROUTINE ASVTRM (NOMTRA,NIVTRA,NIVIMP)`
- `SUBROUTINE DSVTRM (NOMTRA,NIVTRA,NIVIMP)`

△ Lorsque les termes à assembler sont stockés selon une même structure, par exemple s'ils sont calculés sur le même domaine de calcul, les opérations d'assemblage peuvent être avantageusement remplacées par des combinaisons linéaires : cf. procédures CLTERM, T2TERM.

3.8.4. Dé-assemblage d'un terme vecteur multi-inconnue

`SDEKAM module assembl`

Il s'agit de l'opération inverse de DSVTRM. Il est possible, après résolution d'un système linéaire par exemple, de reconstruire un terme 'uni-inconnue' par extraction des valeurs d'un terme 'multi-inconnue' liées à une inconnue à l'aide de la procédure (DSMAKE lu de droite à gauche) :

• SUBROUTINE SDEKAM (NOMTRA,NIVTRA,NOMINC,NOMTRM,NIVTRM,NIVIMP)

où NOMTRA,NIVTRA désigne un terme liée à une structure 'multi-inconnue', NOMINC est le nom d'une inconnue de cette structure. NOMTRM,NIVTRM désigne le terme résultat de ce 'dé-assemblage'.

3.9. Calcul de produits scalaires ou hermitiens

XSTERM module assembl

Le calcul du produit scalaire ou hermitien de 2 termes vectoriels (uni-colonnes) s'obtient par appel à la procédure

• SUBROUTINE XSTERM (SOUH,NMVEC1,NVVEC1,NMVEC2,NVVEC2,*TYRESU*,*RESULT*,*CESULT*)

où la valeur de SOUH est

- ▷ "S" pour un produit scalaire.
- ▷ "H" pour un produit hermitien.

et le résultat est retourné dans la variable réelle *RESULT* ou complexe *CESULT* selon le type du résultat (retourné dans *TYRESU*).

3.10. Calcul de termes par multiplication matrice×vecteur

MATVEC module assembl

Il s'agit en particulier du calcul des termes de second membre d'un problème issus d'une donnée *de second membre* (de condition de Neuman ou Fourier, par exemple), cette donnée étant calculée préalablement comme un terme de type '*valeurs nodales*'.

Reprenons le cas de l'intégrale NEUGAM $\approx \int_\Gamma g\psi\, d\gamma \quad où \quad g = \dfrac{\partial \phi_w}{\partial n_\Gamma}$ dans lequel on suppose que la fonction g a été calculée comme terme VALEUR NODALE 'DPHIn' DE TYPE 'DF/DN' (ou 'N.F'). On aura alors NEUGAM $= \sum_{i\in\Gamma} g_i \int_\Gamma w_i\, w_j\, d\gamma$ qui sera calculé par le produit matrice×vecteur :

'NEUGAM=PDELTA×DPHIn'

où PDELTA représente la matrice de l'intégrale $\int_\Gamma \phi\, \psi\, d\gamma$.

Cette opération est déclenchée par appel à la procédure

• SUBROUTINE MATVEC (NOMMAT,NIVMAT,NOMVEC,NIVVEC,NOMRES,NIVRES,NIVIMP)

dont les arguments parlent d'eux-mêmes.

Exemple 1 Calcul du terme de second membre correspondant à $\int_\Gamma \partial\phi_w/\partial n_\Gamma\, \psi\, d\gamma$

```
CALL MATVEC ('PDELTA',1,'DPHIn',1,'NEUGAM',1,0)
```

où DPHIn, de niveau 1 est le nom du terme VALEUR NODALE défini sur Γ par $\partial\phi_w/\partial n_\Gamma$.

Exemple 2 Calcul de la norme L_2 approchée d'un vecteur à coefficients réels (après résolution, par exemple) sous la forme $v^T A v$ où A est une matrice de masse :

```
CALL MATVEC ('MASSE',1,'VECT',1,'AUX',1,0)
CALL XSTERM ('S','VECT',1,'AUX',1,TYPE,L2NORM,COMPLE)
L2NORM = SRQT (L2NORM)
```

Pour calculer la norme L_2 de la divergence d'un vecteur, on remplacera ci-dessus la matrice de masse par la matrice issue de l'intégrand DIVDIV.

3.11. Calcul de termes à partir d'autres termes

`COTERM, EXTERM, T1TERM module assembl`

Il s'agit de procédures de type 'post-traitement'. Elles permettent de construire un terme constitué des valeurs absolues ou modules des coefficients d'un autre terme, d'extraire les parties réelles ou imaginaires, les modules et arguments des coefficients d'un terme complexe, d'extraire les coefficients d'un terme multi-inconnues correspondant à une inconnueetc.

• `SUBROUTINE COTERM (NOUT,NOMTR1,NIVTR1,NOMDOM,NOMTR2,NIVTR2,NIVIMP)`

Transformation d'un terme associé à une inconnue vectorielle défini sur un domaine de bord de nom `NOMDOM` en un terme dont les coefficients sont les composantes normale ou tangentielle (2D seulement) du premier terme selon la valeur de l'argument caractère `NOUT` :

▷ `"N"` composante normale du terme.
▷ `"T"` composante tangentielle du terme (2D seulement).

⚠ Le calcul des normales doit avoir été effectué sur le domaine (le cas échéant en le demandant explicitement par directives.

⚠ Le terme (`NOMTR1,NIVTR1`) est nécessairement défini sur le domaine de nom `NOMDOM`. Dans le cas contraire on peut préalablement utilisé la procédure suivante.

• `SUBROUTINE EXTERM (NOMTR1,NIVTR1,NOMDOM,INDEXC,INDEXL,NOMTR2,NIVTR2 ,NIVIMP)`

Extraction du terme '1' affectée d'une seule inconnue (un vecteur) sur un domaine de nom `NOMDOM` en le terme '2'. Les indices `INDEXC` et `INDEXL` permettent de choisir un domaine de calcul particulier associé au domaine géométrique de nom `NOMDOM` ; on conseille de les choisir tous deux nuls.

• `SUBROUTINE T1TERM (NOMTR1,NIVTR1,TRANS,NOMTR2,NIVTR2,NIVIMP)`

Transformation des coefficients du terme (`'1'` reportée dans le terme `'2'`) ; l'argument chaîne de caractères `TRANS` permet de choisir l'opération définissant la transformation :

▷ `"=", "CP"` duplication du terme (les attributs sont ceux du terme `'1'`).
▷ `"-"` opposé du terme (les attributs sont ceux du terme `'1'`).
▷ `"ABS"` terme dont les coefficients sont les modules des coefficients du terme `'1'` (les attributs sont ceux du terme `'1'`).
▷ `"EXP"` terme dont les coefficients sont les exponentielles des coefficients du terme `'1'` (les attributs sont ceux du terme `'1'`).
▷ `"LOG"` idem avec les logarithmes.
▷ `"SQRT"` idem pour les racines carrées.
▷ `"RE", "REAL"` idem pour les parties réelles.
▷ `"IM", "AIMAG"` idem pour les parties imaginaires.
▷ `"CMPLX"` idem en transformant un terme à coefficients réels en un terme à coefficients complexes.
▷ `"CONJG"` idem en transformant un terme à coefficients complexes en un terme à coefficients complexes conjugués.
▷ `"ATAN2", "PHASE"` idem en transformant un terme à coefficients complexes en un terme à coefficients réels dont les valeurs sont les arguments (`ATAN2(X,Y)`).
▷ `"MODULE ET PHASE"` idem en transformant un terme à coefficients complexes en un terme à coefficients complexes dont les parties réelles contiennent les modules et les parties imaginaires l'argument (`ATAN2(X,Y)`).
▷ `"COL[ONNE] i"` extraction de la i^{e}me colonne du terme multi-colonnes `'1'`.
▷ `"COMPOSANTE i"` extraction des coefficients correspondant à la i^{e}me composante de l'inconnue vectorielle du terme `'1'`.
▷ `"NUL"` retourne le terme `'2'` avec tous ses coefficients nuls et les attributs du terme `'1'`.
▷ `"IDENTITE"` retourne le terme matriciel Identité `'2'` avec les attributs du terme matriciel `'1'`.

▷ `"=VALEURS"` retourne les valeurs du terme '1' avec les attributs du terme '2' (qui doit nécessairement exister).

3.12. Prise en compte des conditions essentielles de Dirichlet

`CDESSE` module `cdesse`

Il s'agit de prendre en compte ces valeurs de *blocage* de degrés de liberté du type

$$\varphi_j = g_j \quad \text{ou} \quad \vec{\varphi}_j = \vec{g}_j \quad \text{pour } j \in D$$

(D est l'ensemble des 'degrés de liberté' bloqués), pour une inconnue φ scalaire ou vectorielle, ou du type

$$\vec{\varphi}_j \cdot \vec{\nu}_j = g_j \quad \text{ou} \quad \vec{\varphi}_j \wedge \vec{\nu}_j = \vec{g}_j$$

pour une inconnue vectorielle, dans un système linéaire

$$A\varphi = b\,.$$

Dans le cas d'une inconnue scalaire le procédé utilisé actuellement est
— de reporter au second membre les valeurs issues de la donnée de Dirichlet :

$$b_i \longrightarrow b_i - \sum_{j\in D} a_{ij} g_j \text{ , pour tous les nøeuds non Dirichlet } i \notin D,$$

— de remplacer les lignes i correspondant aux *d.l. Dirichlet* ($i \in D$) par la ligne

$$\sum_j a_{ij}\varphi_j = b_i \longrightarrow a_{ii}\varphi_i = a_{ii} g_i,$$

— d'annuler les colonnes j de la matrice correspondant aux *d.l. Dirichlet* ($j \in D$).

Dans le cas d'une inconnue de type vectorielle et pour les conditions de type $\vec{\varphi}_j \cdot \vec{\nu}_j = g_j$ ou $\vec{\varphi}_j \wedge \vec{\nu}_j = \vec{g}_j$, le procédé est le même après passage au repère local ($\vec{\nu}, \vec{\tau}$ ou $\vec{\nu}, \vec{\tau}_1, \vec{\tau}_2$) en chaque nœud du domaine supportant la condition aux limites.

Les conditions essentielles sont déclarées soit par directives (à l'aide de la sous-directive `CONDITION ESSENTIELLE` de la directive générale `CALCUL SUR LE DOMAINE`), soit à l'aide de la procédure de haut niveau `MKTERM`. Les calculs d'élimination des d.l. bloqués sont déclenchés par appel à la procédure

• `SUBROUTINE CDESSE (NOMESS,NIVESS,NOMMAT,NIVMAT,NOMSCM,NIVSCM)`

dont les arguments sont :

`NOMESS` nom de la condition essentielle ; il s'agit aussi du nom du terme contenant le vecteur des valeurs de blocage des d.l. pour une condition non homogène ;
`NIVESS` son niveau ;
`NOMMAT` nom du terme contenant la matrice du système linéaire sur lequel est effectuée l'élimination ;
`NIVMAT` son niveau ;
`NOMSCM` nom du terme contenant le second membre du système linéaire sur lequel est reportée l'élimination ; un tel terme peut ne pas exister avant l'élimination, auquel cas il est créé dans la procédure.
`NIVSCM` son niveau.

△ La condition essentielle peut ne porter que sur une matrice (e.g. pour un calcul de valeurs propres), dans ce cas la variable `NOMSCM` est remplacée par la chaîne *blanche* ' '.

△ L'inconnue sur laquelle porte l'élimination est l'inconnue affectée au 'terme' condition essentielle. Elle est nécessairement aussi affectée à la matrice et au second membre ou à un sous-ensemble de ceux-ci dans le cas d'un système multi-inconnues.

3.13. Prise en compte des conditions de transmission

CDTRAN module cdesse

Il s'agit de prendre en compte la transmission des valeurs de deux inconnues du problème sur le domaine frontière T sous la forme :

$$[K_{1,j}]\,\varphi_j^{(1)} = [K_{2,j}]\,\varphi_j^{(2)} + g_j \quad \text{ou} \quad [K_{1,j}]\,\vec{\varphi}_j^{(1)} = [2,j]\,\vec{\varphi}_j^{(2)} + \vec{g}_j \quad \text{pour } j \in T$$

pour inconnues scalaires ou vectorielles ou de la forme

$$[K_{1,j}]\,\vec{\varphi}_j^{(1)} \cdot \vec{\nu}_j = [2,j]\,\vec{\varphi}_j^{(2)} \cdot \vec{\nu}_j + g_j \quad \text{ou} \quad [K_{1,j}]\,\vec{\varphi}_j^{(1)} \wedge \vec{\nu}_j = [2,j]\,\vec{\varphi}_j^{(2)} \wedge \vec{\nu}_j + \vec{g}_j \text{ , pour } j \in T$$

par exemple dans un système linéaire multi-inconnues du type

$$\begin{pmatrix} A^{(11)} & A^{(12)} \\ A^{(21)} & A^{(22)} \end{pmatrix} \begin{pmatrix} \varphi^{(1)} \\ \varphi^{(2)} \end{pmatrix} = \begin{pmatrix} b^{(1)} \\ b^{(2)} \end{pmatrix}$$

Dans ces formules T est l'ensemble des 'degrés de liberté' portant la condition de transmission $[K_1]$ et $[K_2]$ représente des scalaires (par défaut 1) ou des matrices (à l'heure actuelle diagonales) de sorte que $K_1^2 + K_2^2$ soit définie positive.
Le processus d'élimination est le même que dans dans le cas d'une condition de Dirichlet après le changement d'inconnues (avec une notation de matrice par bloc si K_i est une matrice) :

$$\begin{pmatrix} \Phi^{(1)} \\ \Phi^{(2)} \end{pmatrix} = \begin{pmatrix} (K_1^2+K_2^2)^{-\frac{1}{2}} K_1 & -(K_1^2+K_2^2)^{-\frac{1}{2}} K_2 \\ (K_1^2+K_2^2)^{-\frac{1}{2}} K_2 & (K_1^2+K_2^2)^{-\frac{1}{2}} K_1 \end{pmatrix} \begin{pmatrix} \varphi^{(1)} \\ \varphi^{(2)} \end{pmatrix}$$

soit, pour $K_i = 1$,

$$\begin{pmatrix} \Phi^{(1)} \\ \Phi^{(2)} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \varphi^{(1)} \\ \varphi^{(2)} \end{pmatrix}$$

Les calculs d'élimination sont effectuées par appel à la procédure

• SUBROUTINE CDTRAN (NOMESS,NIVESS,NOMMAT,NIVMAT,NOMSCM,NIVSCM,NIVIMP)

dont les arguments sont :

NOMESS nom du terme définissant la condition de transmission déclarée par directives ou à l'aide de la procédure MKTERM ;
NIVESS son niveau
NOMMAT nom du terme contenant la matrice multi-inconnues du système linéaire sur lequel est effectuée l'élimination ;
NIVMAT son niveau ;
NOMSCM nom du terme contenant le second membre multi-inconnues du système linéaire sur lequel est reportée l'élimination ; un tel terme peut ne pas exister avant l'élimination, auquel cas il est créé dans la procédure.
NIVSCM son niveau ;
NIVIMP niveau d'impression en cours des calculs.

△ La condition essentielle peut ne porter que sur une matrice (e.g. pour un calcul de valeurs propres), dans ce cas la variable NOMSCM est remplacée par la chaîne *blanche* ' '.

3.14. Couplage 'Éléments Finis–Représentation Intégrale'

La résolution de problèmes aux limites extérieurs par la méthode de couplage 'éléments finis–représentation intégrale' est une fonctionnalité importante du code MÉLINA.
On trouvera dans la documentation du module couplag (si on le trouve) la description de la méthode implémentée et des procédures de calcul des termes de couplage RELFRI, RELDIR, COGDNO, COGNOP, COUMAT, COUVEC et de calcul de représentation intégrale. Nous renvoyons le lecteur avide d'aventures scabreuses à sa description détaillée dans la documentation du module couplag.

3.15. Termes de couplage Éléments Finis Localisés

CALEFL module efloc

La méthode de Éléments Finis Localisés est une autre technique pour la résolution de problèmes extérieurs. . . et la documentation n'est pas (toujours) disponible.

3.16. Méthodes itératives de résolution de systèmes linéaires

La majorité des méthodes et problèmes traités actuellement avec MÉLINA conduisent à des systèmes linéaires à matrice non-symétrique et sans propriétés particulières, et pour lesquels aucune des méthodes itératives de type Gauss-Seidel ou sur-relaxation, par exemple, ne semble converger. Les méthodes itératives utilisées dans MÉLINA sont la méthode de Gradient Bi-Conjugué (Préconditionné) et la méthode de Gradient Conjugué Carré (Préconditionné) (*Conjugate Gradient Squared*). On trouvera d'autres méthodes de résolution dans le module syslin, voir le chapitre de la documentation le concernant.

La mise en œuvre de ces méthodes de résolution comporte 3 étapes :
– Construction de la matrice de préconditionnement à l'aide des procédures d'assemblage de termes matriciels du module assembl (voir plus haut) ;
– Factorisation $L_1 \cdot U$ ou $L \cdot L^*$ incomplète de cette matrice, ou transformation du stockage sous forme bimorse avec remplissage éventuel et factorisation incomplète ou transformation du stockage sous forme profil et factorisation ;
– Résolution du système par itérations de gradient.

3.16.1. Factorisation (incomplète) d'une matrice de préconditionnement

FILU, FIMLU, FALU module syslin

Factorisation $L_1 \cdot U$ incomplète sans remplissage

Les matrices issues du module d'assemblage sont stockées sur forme bimorse ou morse-I. Cette factorisation de la matrice de préconditionnement P incomplète est une factorisation de Gauss $L_1 \cdot U$ (L_1 matrice triangulaire inférieure à diagonale unité, U matrice triangulaire supérieure) au cours de laquelle tous les coefficients nuls du profil de la matrice sont *oubliés* (il s'agit des coefficients *inter-plages* dans une structure (bi)morse(-I)). Il n'y a donc aucun *remplissage* de la matrice, et les matrices L_1 et U sont stockées selon les mêmes pointeurs de stockage (bi)morse(-I) que la matrice P et le cas échéant $L \cup U$ est stockée dans le même tableau que P.

• SUBROUTINE FILU (NOMMAT,NIVMAT,SYMSTR,NOMFAC,NIVFAC,NIVIMP)

où NOMMAT,NIVMAT est le (nom,niveau) du terme contenant la matrice P à factoriser $L \times U$ incomplètement et NOMFAC,NIVFAC le (nom,niveau) du terme contenant les matrices L et U selon la même structure bimorse ou morse-I que P ; SYMSTR est une (chaîne de) caractère(s) indiquant que la matrice P est symétrique en structure ('S') ou non ('N'). Cette donnée est ignorée en cas de stockage bimorse. Dans le cas ou la matrice, stockée morse-I, est symétrique en structure on peut faire l'économie d'un tableau de pointeurs de la partie triangulaire supérieure sous forme morse-c.

△ On peut, si le terme NOMMAT,NIVMAT n'est plus utile dans la suite, utiliser le même terme pour P et $L \cup U$, il suffit d'entrer NOMFAC=NOMMAT **et** NIVFAC=NIVMAT.

Factorisation $L_1 \cdot U$ incomplète après remplissage partiel

Les matrices issues du module d'assemblage sont stockées sur forme (bi)morse(-I). Il est possible afin d'accroître le poids des coefficients voisins de la diagonale de transformer la structure initiale de la matrice par concaténation de plages proches de la diagonale. Cette transformation résulte en une matrice stockée sous forme bimorse.

• SUBROUTINE FIMLU (NOMMAT,NIVMAT,SYMSTR,NPLCAT,NOMFLU,NIVFLU,NIVIMP)

Arguments identiques à ceux de FILU avec en outre NPLCAT qui représente un paramètre de remplissage de la matrice préalable à la factorisation incomplète. Plus précisément NPLCAT représente le nombre de plages sous-diagonales concaténées en une plage unique se terminant par le coefficient sous-diagonal et le nombre de plages sur-diagonales concaténées en une plage unique commençant par le coefficient sur-diagonal.

⚠ ⚠ On ne peut utiliser le même terme pour P et $L \cup U$, car les types de stockage de ces termes diffèrent.

Factorisation $L_1 \cdot U$ complète

Cette factorisation de la matrice (de préconditionnement P) est une factorisation de Gauss $L_1 \cdot U$ après transformation du type de stockage sous forme profil stable par factorisation de Gauss (L_1 matrice triangulaire inférieure à diagonale unité, U matrice triangulaire supérieure).

• SUBROUTINE FALU (NOMMAT,NIVMAT,SYMSTR,NOMFLU,NIVFLU,NIVIMP)

Arguments identiques à ceux de FILU. La matrice est transformée selon le stockage profil avant factorisation.

⚠ ⚠ On ne peut utiliser le même terme pour P et $L \cup U$, car les types de stockage de ces termes diffèrent.

△ Il existe aussi dans le code des procédures de factorisation pour des matrices symétriques ou auto-adjointes. Voir la prochaine section ou consulter la documentation du module syslin.

3.16.2. Méthodes de gradient (préconditionné)

GBCT, GCJ, GCS module syslin

Gradient biconjugué [préconditionné]

• SUBROUTINE GBCT (NOMMAT,NIVMAT,NOMSCM,NIVSCM,NOMPRE,NIVPRE,EPSILO
,MXITER,NOMSOL,NIVSOL,NIVIMP,*)

où NOMMAT,NIVMAT est le terme contenant la matrice A du système linéaire à résoudre, NOMSCM,NIVSCM le terme contenant le second membre, NOMPRE,NIVPRE le terme contenant les matrices L_1 et U de la factorisation $L_1 \times U$ de la matrice de préconditionnement ou la chaîne *blanche* ' ' en cas de non-préconditionnement et NOMSOL,NIVSOL le terme contenant le vecteur initial en entrée et recevant la solution en sortie. EPSILO est seuil d'arrêt des itérations de gradient et MXITER le nombre maximum d'itérations de gradient au delà duquel on suppose la non-convergence de la méthode.
* est un paramètre étiquette de retour exceptionnel en cas de non-convergence en MXITER itérations.

⚠ Les matrices des systèmes linéaires issues des méthodes utilisées dans MÉLINA pour la résolution des problèmes extérieurs n'ont pas de propriétés pour lesquelles on peut obtenir un quelconque résultat de convergence de la méthode du gradient conjugué. L'utilisation de cet algorithme de résolution se fait sans aucune garantie de convergence, ni même de qualité du résultat en cas de convergence.

△ De plus la *'qualité'* de la factorisation incomplète proposée ci-dessus est essentiellement liée à la numérotation des degrés de liberté du problème.
Cependant dans de nombreux cas traités de résolution de problèmes de propagation d'ondes par la méthode 'Éléments Finis–Représentation Intégrale', et en choisissant un préconditionnement convenable(?), on constate un bon comportement de l'algorithme.

Gradient conjugué 'squared' [préconditionné]

• SUBROUTINE GCS (NOMMAT,NIVMAT,NOMSCM,NIVSCM,NOMPRE,NIVPRE,EPSILO
,MXITER,NOMSOL,NIVSOL,NIVIMP,*))

Mêmes arguments et remarques.

△ Il existe aussi dans le code des procédures de gradient, comme GCJ pour des matrices symétriques ou auto-adjointes. Consulter la documentation du module syslin.

Exemple Cas du problème de Helmholtz extérieur

On choisit comme matrice de préconditionnement la matrice du problème dans on impose la condition de rayonnement à distance finie (il est connu que la solution de ce problème converge vers la solution du problème extérieur lorsque la frontière Σ s'éloigne à l'infini) :

$$\int_\Omega \overrightarrow{\text{grad}}\,\phi\ \overrightarrow{\text{grad}}\,\psi\,d\omega - k^2\int_\Omega \phi\psi\,d\omega - ik\int_\Sigma \phi\psi\,d\sigma = \int_\Gamma g\psi\,d\gamma$$

On suppose que la matrice de préconditionnement a déjà été calculée (cf. procédure ASMTRM et est stockée dans le terme PRECON de niveau 0, et que la matrice et le second membre du système linéaire sont définis par les termes MATRIS,1 et SCMEMB,1.

```
*     Factorisation incomplete de la matrice de preconditionnement
*     Le meme terme est utilise pour la factorisee incomplete
      CALL FILU    ('PRECON',0,'S','PRECON',0,0)
*     Resolution par Gradient conjugue preconditionne
      CALL GCS    ('MATRIS',1,'SCMEMB',1,'PRECON',0,EPSIL,MAXTER
     &            ,'SOLUTI',1,0,*100)
      GO TO 200
100   CALL BAISE  ( 'Non-convergence du gradient conjugue')
*     Suite du programme
200   ...
      END
```

3.17. Méthodes directes de résolution de systèmes linéaires

FALU, FALLT, FALDLT, FALLA, FALDLA, FASV, GAUSMC module syslin

Il s'agit de résolution par descente/remontée après factorisation de type Gauss ou Choleski. Pour la factorisation de Gauss, voir FALU ci-dessus ; pour les factorisations de Choleski $L\cdot{}^tL$, $L_1\cdot D\cdot{}^tL_1$, $L\cdot L^*$ et $L_1\cdot D\cdot L_1^*$, dont les arguments sont les mêmes, consulter la documentation du module syslin.

Systèmes linéaires à matrice factorisée : Descente/Remontée

Après factorisation de la matrice sous l'une des formes ci-dessus, la résolution d'un système linéaire se ramène à la résolution de deux systèmes triangulaires inférieur et supérieur effectuée par appel à la procédure

• SUBROUTINE FASV (NOMMAT,NIVMAT,NOMSCM,NIVSCM,NOMSOL,NIVSOL,NIVIMP)

dont les arguments parlent d'eux-mêmes.

3.18. Condensation statique dans un système linéaire

PGAUSS module syslin

Il s'agit d'une méthode d'élimination partielle des inconnues d'un système linéaire, pour n'en conserver que les inconnues liées aux nœuds d'un ou de plusieurs domaines.

• SUBROUTINE PGAUSS (NOMMAT,NIVMAT,NOMSCM,NIVSCM,DOMAIN
,NOMMAS,NIVMAS,NOMSCS,NIVSCS,NIVIMP)

où NOMMAT,NIVMAT est le terme contenant la matrice A du système linéaire, NOMSCM,NIVSCM le terme contenant le second membre, NOMMAT,NIVMAT et NOMSCS,NIVSCS sont la matrice et le second membre modifié après élimination. DOMAIN est le nom du domaine ou désigne un domaine union de plusieurs domaines sous la forme d'une chaîne de caractères de la forme 'domaine1[Union domaine2 [Union ...]].

3.19. Méthode de calcul de valeurs propres

VPINIT, VPMISE librairie valpro

Calcul de quelques valeurs/vecteurs propres pour le problème [généralisé]

$$Av = \lambda[B]v$$

par la méthode d'itérations de sous-espaces.
Le calcul est effectué par appels des deux procédures VPINIT, qui crée les tableaux nécessaires et calcule une base orthogonale de vecteurs aléatoires (point de départ des itérations), et VPMISE pour le calcul des valeurs/vecteurs propres proprement dit.

• SUBROUTINE VPINIT (NMMATA,NVMATA,NMVALP,NVVALP,NMVECP,NVVECP
,NMESVP,NVESVP,NBVPDS,NVECAC,DIMEST,NIVIMP)

dont les arguments sont

NMMATA nom de la matrice A ou B de $Av = \lambda[B]v$
NVMATA son niveau
NMVALP nom du tableau recepteur des valeurs propres
NVVALP son niveau
NMVECP nom du tableau recepteur des vecteurs propres, rangés par colonnes
NVVECP son niveau
NMESVP nom du tableau recepteur des bornes d'erreur résiduelles
NVESVP son niveau
NBVPDS nombre de valeurs propres souhaité
NVECAC nombre de valeurs précédement calculé (le cas échéant 0)
DIMEST dimension du sous-espace de travail ($>$ NBVPDS)
NIVIMP niveau d'impression

• SUBROUTINE VPMISE (CHOIX,NMMATA,NVMATA,NMMATB,NVMATB,NMVALP,NVVALP
,NMVECP,NVVECP,NMESVP,NVESVP,NBVPDS,NBVPCA,OPTION)

dont les arguments sont

CHOIX chaîne de caractères définissant les valeurs propres à calculer et le type de factorisation de la matrice A ou B :
- si NMMATB$\neq$' ' (problème généralisé)
  - ▷ 'L.LT(A)' calcul des plus petites valeurs propres de $Av = \lambda Bv$, la matrice A (définie positive) est préalablement factorisée $L.L^t$.
  - ▷ 'L.LT(B)' calcul des plus grandes valeurs propres de $Av = \lambda Bv$, la matrice B (définie positive) est préalablement factorisée $L.L^t$.
- sinon (NMMATB=' ')
  - ▷ ' ' calcul des plus grandes valeurs propres de $Av = \lambda v$
  - ▷ 'L.LT' calcul des plus petites valeurs propres de $Av = \lambda v$, la matrice A (définie positive) est préalablement factorisée $L.L^t$.

NMMATA nom contenant la matrice (factorisée) A
NVMATA son niveau
NMMATB nom contenant la matrice (factorisée) B (' ' pour $Av = \lambda v$)
NVMATB son niveau

NMVALP nom du tableau recepteur des valeurs propres
NVVALP son niveau
NMVECP nom du tableau recepteur des vecteurs propres, rangés par colonnes
NVVECP son niveau
NMESVP nom du tableau recepteur des bornes d'erreur résiduelles
NVESVP son niveau
NBVPDS nombre de valeurs propres souhaité
NBVPCA nombre de valeurs précédement calculé (le cas échéant 0)
OPTION chaîne de caractères (égale à ' ', si l'on retient les valeurs par défaut) définissant la liste des valeurs des paramètres optionnels, résultant de la concaténation de chaînes du type 'option=nom_de_donnée' (séparés par des virgules) où option est à choisir dans la liste ci-dessous (**attention les minuscules/majuscules sont significatives**) et nom_de_donnée est le nom d'une donnée (!) :

option	nom de donnée
'Epsilon'	constante réelle contenant le seuil d'arrêt des itérations (défaut 10^{-5})
'Saut'	constante réelle caractérisant le 'niveau d'isolement' des valeurs propres calculées par rapport aux autres (défaut 10.)
'Maximum_Iteration'	constante entière limitant le nombre d'itérations
'Impression'	constante entière définissant le niveau d'impression des calculs (défaut 0)

Exemple 1 Calcul des 5 plus petites valeurs propres et des vecteurs propres associés de $Av = \lambda Bv$, dans un sous-espace de dimension 8 avec valeurs par défaut des paramètres optionnels

```
      CALL FALLT  ('A',1,'Afact',1,0)
      CALL VPINIT ('A',1,'Valpro',1,'Vecpro',1,'Bornes',1,5,0,8,0)
      CALL VPMISE ('L.LT(A)','Afact',1,'B',1,'Valpro',1,'Vecpro',1
     &            ,'Bornes',1,5,0,' ')
```

Exemple 2 Idem avec modification des paramètres optionnels

```
      CALL FALLT  ('A',1,'Afact',1,0)
      CALL VPINIT ('A',1,'Valpro',1,'Vecpro',1,'Bornes',1,5,0,8,0)
      CALL VPMISE ('L.LT(A)','Afact',1,'B',1,'Valpro',1,'Vecpro',1
     &            ,'Bornes',1,5,0,'Epsilon=VALEPS,Impression=IMPPRO')
```

où VALEPS est le nom d'une donnée constante réelle, IMPPRO le nom d'une donnée constante entière.

3.20. Comparaison de 2 termes

DFTERM librairie sdexplo

• SUBROUTINE DFTERM (NOMTR1,NIVTR1,NOMTR2,NIVTR2,NIVIMP)

Il s'agit d'une procédure qui permet de calculer et d'afficher dans un fichier l'écart entre deux termes. Elle calcule les écarts quadratiques (norme ℓ_2) absolus et relatifs et maximaux (norme ℓ_∞) absolus et relatifs. Les deux termes peuvent correspondre à des numérotations différentes (par exemple, définis sur des domaines différents), la comparaison ne porte que sur les supports des degrés de liberté communs aux termes. Cette procédure est surtout utilisée pour comparer une solution calculée à une solution exacte, ou pour comparer deux résultats issus de deux calculs différents (par exemple la solution d'un système linéaire issue de deux méthodes de résolution différentes).

3.21. Fichiers de sortie pour exploitation graphiques de résultats

`M2UOUV, M2UENR, M2UESD, M2UGEO` module `me2uni`

3.22. Affichage de temps de calcul

`CHRONO` librairie `utiliter`

Cette procédure permet l'impression de temps de calcul des tâches de l'exécution. Elle utilise une fonction C (`ctemps.c`, librairie utiliter) qui fournit le temps écoulé en milli-secondes depuis le 1^{er} Janvier 1970. Le temps affiché est la durée écoulée depuis son précédent appel (le premier appel a lieu au début de la procédure d'initialisation `INITIE`).

3.23. Terminer une exécution

`BYE` librairie `utiliter`

- `SUBROUTINE BYE (CHAINE)`

Un appel à cette procédure permet une fin d'exécution 'propre' avec impression de la chaîne de caractères `CHAINE` et impression du nombre de 'warnings' émis et imprimés dans le fichier des messages.

```
      SUBROUTINE BYE ('C''est fini.')
```

Il et fortement conseillé de terminer un programme principal par l'appel de cette procédure qui indique, le cas échéant, qu'un certain nombre de *warnings* ont été émis en cours de l'exécution qui pourrait indiquer un dysfonctionnement. . . ET de lire ces éventuels messages.

Procédures d'allocation dynamique

Dans ce cours chapitre nous décrivons quelques procédures de l'allocation dynamique essentiellement pour permettre à l'utilisateur de créer des tableaux de rechercher leur adresse, de les détruire lorsqu'ils sont devenus inutiles ou de les sauvegarder[(1)] pour libérer, si nécessaire, la place qu'ils occupent dans les super-tableaux du code. On prendra soin en rédigeant le programme principal d'utiliser ces procédures. Le code ne peut évidement pas décider si tel tableau est provisoirement ou définitivement inutile. Ceci concerne principalement les tableaux créés explicitement par le développeur de l'application, ou les tableaux définis comme arguments des procédures de *haut niveau* et créés lors de leur utilisation.

1. Description des procédures principales

1.1. Création d'un tableau

TBCREE, TBCRSU

La création d'un tableau, c'est-à-dire la réservation dans les super-tableaux de la place nécessaire au tableau, est effectuée par appel à la procédure

- `SUBROUTINE TBCREE (NOMTB,NIVTB,TYPTB,LNGTB,Commentaire)`

Cette procédure provoque la création du tableau de nom `NOMTB` (chaîne de caractères d'au plus 6 caractères dont le premier est nécessairement une lettre) et de niveau `NIVTB` (entier). `TYPTB` est un code désignant le type du tableau (1 : entier, 2 : réel, 3 : double précision, 4 : complexe, 10+*l* : character**l*). `LNGTB` est la longueur, comptée en articles, du tableau. `Commentaire` est une chaîne de caractères.

△ Si le tableau existe, la création se réduit à la vérification de la cohérence de ses anciens attributs avec les nouveaux avec un éventuel réajustement de sa taille.

△ Il existe aussi une procédure de création (TBCRSU) pour les tableaux dont on ne connaît pas la taille exacte mais seulement un majorant plus ou moins bien estimé. Il est souvent nécessaire dans la suite d'ajuster sa taille lorsqu'elle peut être connue (TBAJST). Consulter votre revendeur habituel.

1.2. Destruction d'un tableau

TBTUER

À l'inverse il est possible de libérer la place qu'occupe un tableau devenu inutile (dans un super-tableau ou dans la mémoire secondaire), à l'aide de la procédure

- `SUBROUTINE TBTUER (NOMTB,NIVTB)`

Cette procédure provoque la destruction du tableau de nom `NOMTB` (chaîne de caractères d'au plus 6 caractères) et de niveau `NIVTB` (entier).

⚠ ⚠ Le tableau doit évidement exister ! dans le cas contraire votre beau programme se terminera en erreur.

(1) Il s'agit plutôt de donner l'autorisation de transfert en mémoire secondaire, les procédures de l'allocation dynamique se chargeant des transferts vers la mémoire secondaire, si la place dans les super-tableaux devient insuffisante, lors de la création d'un nouveau tableau, par exemple.

1.3. Sauvegarde d'un tableau

TBSAVE

Dans le même ordre d'idée si un tableau est provisoirement inutilisé, il est possible d'autoriser son transfert en mémoire secondaire dans l'un des tableaux buffers du code à l'aide de la procédure

• SUBROUTINE TBSAVE (NOMTB,NIVTB)

Cette procédure place le tableau (NOMTB,NIVTB) en état de sauvegarde différée. Son transfert effectif vers la mémoire secondaire n'a lieu qu'en cas de nécessité (manque de place dans les super-tableaux). Toutes les procédures des modules de calculs (en particulier les procédures de *haut niveau*) utilisent cette procédure pour les tableaux qu'elles manipulent, à l'exception de certains tableaux contenant les structures de données (tableaux de noms, tableaux de descriptions des termes, etc.) indispensables à tout moment de l'exécution.

1.4. Recherche de tableaux

TBRR1, TBRR*l*

Pour pouvoir utiliser à nouveau un tableau mis en état de sauvegarde ou effectivement sauvegardé en mémoire secondaire, il est nécessaire de supprimer son état de sauvegarde différée ou de le réintroduire dans les super-tableaux. On utilise à cet effet l'une des procédures

• SUBROUTINE TBRR1 (ERCODE,NOMTB,NIVTB,*LNGTB*)

pour le tableau (NOMTB,NIVTB) ou

• SUBROUTINE TBRR*l* (ERCODE,NOMTB1,NIVTB1,*LNGTB1*,NOMTB2,NIVTB2,*LNGTB2*
. . . . . ,NOMTB*l*,NIVTB*l*,*LNGTBl*)

pour les *l* tableaux (NOMTB1,NIVTB1) à (NOMTB*l*,NIVTB*l*) pour *l*=1, 2, . . . , 20.

Ces procédures retournent la longueur des tableaux recherchés comptée en articles. On ne trouvera pas ces procédures explicitement dans la librairie allodyn, il s'agit des points d'entrée secondaires (ENTRY POINTS) de la procédure TBRRFL (Recherche de TaBleaux en RaFale).

△ ERCODE est ici une variable chaîne de caractères de longueur *l* au moins qui contient en sortie, à l'adresse *i*, le caractère ' ' si le tableau numéro *i* existe, 'C' sinon. Ces renseignements sont utilisés de manière interne et sont en général inaccessibles au développeur standard. En effet

⚠ ⚠ Si un tableau n'existe pas (faute de frappe dans le nom), le programme se termine en erreur après impression d'un message.

1.5. Adresses de tableaux

TBAR1, TBAR*l*

Pour utiliser un ou plusieurs tableaux du code géré par l'allocation dynamique, il est nécessaire de rechercher son ou leurs adresses en mémoire. C'est l'objet des procédures

• SUBROUTINE TBAR1 (ERCODE,NOMTB,NIVTB,*ADRTB*)

qui retourne l'adresse en mémoire ADRTB du tableau (NOMTB,NIVTB) et

• SUBROUTINE TBAR*l* (ERCODE,NOMTB1,NIVTB1,*ADRTB1*,NOMTB2,NIVTB2,*ADRTB2*
. . . . . ,NOMTB*l*,NIVTB*l*,*ADRTB1*)

qui retourne les adresses des *l* tableaux (NOMTB1,NIVTB1) à (NOMTB*l*,NIVTB*l*) pour *l*=1, 2, . . . , 20.

Comme ci-dessus, ces procédures ne figurent pas explicitement dans la librairie allodyn, il s'agit des points d'entrée secondaires (ENTRY POINTS) de la procédure TBARFL (Adresses de TaBleaux en RaFale).

△ ERCODE est défini comme plus haut et contient en sortie, à l'adresse *i*, le caractère ' ' si le tableau numéro *i* existe et réside en mémoire centrale, 'R' si le tableau réside en mémoire secondaire ou est en état de sauvegarde différée, 'C' si le tableau n'existe pas. Ces renseignements sont utilisés de manière interne et sont en général inaccessibles au développeur standard. En effet

⚠ ⚠ Si un tableau n'existe pas (faute de frappe dans le nom), le programme se termine en erreur après impression d'un message.

⚠ ⚠ Si un tableau existe mais est en mémoire secondaire ou en état de sauvegarde différée, c'est-à-dire si on a autorisé son transfert en mémoire secondaire en cas de nécessité (TBSAVE), il est nécessaire de le rechercher et au besoin de le rapatrier vers la mémoire centrale a l'aide de la procédure TBRR*l* ; le programme se termine en erreur après impression d'un message qui en informe.

1.6. Type, longueur et ajustement de la taille d'un tableau

TBTYPE, TBARTI, TBAJST

On peut dans certains cas ignorer le type, par exemple (réel ou complexe), d'un tableau ; on utilisera alors la procédure

• SUBROUTINE TBTYPE (NOMTB,NIVTB,*TYPTB*)

qui retourne TYPTB le type du tableau (entier égal à 1 : entier, 2 : réel, 3 : double précision, 4 : complexe, 10+*l* : character***l*).
De même la longueur (comptée en nombre d'articles) d'un tableau en mémoire centrale ou en mémoire secondaire peut être obtenu à l'aide de la procédure

• SUBROUTINE TBARTI (NOMTB,NIVTB,*NBARTI*)

qui retourne NBARTI le nombre d'articles du tableau.
Lorsque la taille d'un tableau ne peut être prévue a priori, elle est en général surestimée et il est souvent nécessaire de l'ajuster après qu'il est été 'rempli', c'est la fonction de

• SUBROUTINE TBAJST (NOMTB,NIVTB,NWARTI)

qui permet de réduire ou d'agrandir la place mémoire allouée à un tableau en portant sa taille (comptée en articles) à NWARTI.

1.7. États des tableaux

STIMPR, PRRUME

• SUBROUTINE STIMPR (IMPFCH)

Cette procédure permet d'afficher à tout moment l'état des super-tableaux sur le fichier d'impression IMPFCH (par exemple IMPPAL=6). C'est un outil de mise au point de la taille des super-tableaux. Il permet de vérifier aussi l'état des tableaux créés par le développeur de l'application.

• SUBROUTINE PRRUME (IMPFCH)

Cette procédure du même type que la précédente permet d'afficher des statistiques concernant tous les tableaux qui ont été utilisés jusquà l'appel de cette procédure (nombre de tableaux vivants, en sauvegarde différée, en mémoire secondaire, morts, . . .).

2. Précautions d'emploi

Il s'agit surtout de quelques conseils d'utilisation de ces procédures d'allocation dynamique.

2.1. Remise en cause des adresses des tableaux

La création d'un nouveau tableau TBCREE, TBCRSU ou le retour dans les super-tableaux d'un tableau figurant dans la mémoire secondaire (TBRRl) est susceptible de remettre totalement en cause les adresses de tous les tableaux gérés par la librairie d'allocation dynamique (c'est aussi bien sûr vrai pour l'ajustement de la taille d'un tableau à l'aide de la procédure TBAJST ou de la procédure SDAJST (librairie sdexplo).
Aucune des adresses déjà connues pour des tableaux quelconques gérés par l'allocation dynamique n'est plus valable après un appel de ces procédures. Il est donc nécessaire de redemander les adresses de tous les tableaux nécessaires dans la suite (ou du moins jusqu'à la prochaine remise en cause).
Il est aussi clair que tout appel à une procédure de **haut niveau** est susceptible de créer des tableaux ou de réintroduire des tableaux en mémoire centrale qui peut à l'extrême provoquer le déplacement de tous les tableaux gérés.
De même un appel aux procédures utilitaires suivantes

CRINCO création d'une inconnue, librairie sdexplo ;
CRTERM création d'un terme, librairie sdexplo ;
KLCSTE recherche ou création d'une constante, librairie sdexplo ;

est susceptible de remettre en cause des adresses des tableaux gérés par la librairie d'allocation dynamique.

3. Exemples d'utilisation

Toutes les procédures de *haut niveau* du code peuvent être consultées comme exemples plus ou moins simples (voir en particulier les procédures d'assemblage de termes par combinaison linéaire dans le module assembl ou les procédures d'utilisation des constantes dans la librairie sdexplo).
On suppose dans les exemples suivants que l'utilisateur programme ces procédures qui pourront être appelées dans le programme principal.

3.1. Création et remplissage d'un tableau

```
      SUBROUTINE EXPTAB (NOMTBE,NIVTBE,NOMTBS,NIVTBS)
*
*     Auteur : Duchenoque Paulo (Mars 1996)
*     Derniere modification : Dugenou Lulu (Mars 1998)
*
*     Calcul d'un terme NOMTBS,NIVTBS dont les coefficients sont
*     les exponentielles des coefficients du terme NOMTBE,NIVTBE
*
      CHARACTER*(*) NOMTBE,NOMTBS
      INTEGER       NIVTBE,NIVTBS
*
      INCLUDE 'ALLOC'
      INCLUDE 'CONTEX'
*
      INTEGER       LGTBE,TYPTBE,MCTBE,MCTBS,INTTYP
      CHARACTER*120 ERCODE
      COMMON/FORMAH/ERCODE
*
*     Pour la signalisation de la procedure en cas de Bug
      NIVESP         = NIVESP+1
      PREFIX(NIVESP) = '<ExpTab>'
*
```

```
*     Recherche du tableau d'entree (restauration en M.C.)
      CALL TBRR1  (ERCODE,NOMTBE,NIVTBE,LGTBE)
*     Type du tableau d'entree
      CALL TBTYPE (NOMTBE,NIVTBE,TYPTBE)
      IF (NOMTBE.EQ.NOMTBS.AND.NIVTBE.EQ.NIVTBS) THEN
*        Les tableaux coincident: leur adresse ?
         CALL TBAR1  (ERCODE,NOMTBE,NIVTBE,MCTBE)
         MCTBS=MCTBE
      ELSE
*        Creation du tableau de sortie
*        (il s'agit d'un simple tableau et non d'un terme
*        sauf s'il existait en tant que terme auparavant)
*        Pour la creation d'un terme utiliser CRTERM (Sdexplo)
         CALL TBCREE (NOMTBS,NIVTBS,TYPTBE,LGTBE,'Tableau cree')
*        Les tableaux different: Leurs adresses ?
         CALL TBAR2  (ERCODE,NOMTBE,NIVTBE,MCTBE,NOMTBS,NIVTBS,MCTBS)
      ENDIF
*
*     Remplissage du tableau de sortie
*      (l'adresse ou les adresses des tableaux sont maintenant connues)
*
      IF (TYPTBE.EQ.INTTYP ('REEL')) THEN
*        Cas d'un tableau reel
         DO 2 I=0,LGTBE-1
            RST(MCTBS+I) = EXP (RST(MCTBE+I))
2        CONTINUE
      ELSEIF (TYPTBE.EQ.INTTYP ('COMPLEXE')) THEN
*        Cas d'un tableau complexe
         DO 4 I=0,LGTBE-1
            CST(MCTBS+I) = EXP (CST(MCTBE+I))
4        CONTINUE
      ELSE
*        Message d'erreur
         CALL ENCLER (TYPTBE,ERCODE(1:8))
         CALL BAISE  ('Type de tableau non autorise :'//ERCODE(1:8))
      ENDIF
*
*     (Autorisation de) Sauvegarde des tableaux
      CALL TBSAVE (NOMTBE,NIVTBE)
      CALL TBSAVE (NOMTBS,NIVTBS)
*     A ne pas oublier en sortie
      NIVESP = NIVESP-1
      END
```

– Mai 2000 –

Procédures 'Utilisateurs'

Nous présentons dans ce chapitre la procédure FCTRM indispensable au fonctionnement du code, mais dont la programmation est liée au problème à traiter et dont la programmation incombe au développeur de l'application.

1. Données de type 'fonction'

Une application de MÉLINA nécessite un certain nombre de données de type fonction. On trouvera les données associées aux termes éléments finis contenant un coefficient variable, par exemple la fonction **f** dans l'intégrale $\int_\Omega \mathbf{f}(x)\,u(x)\,v(x)\,dx$ déclarée par

```
CALCUL SUR LE DOMAINE 'OMEGA'
       TERME ELEMENTS FINIS 'MASSE'
         INCONNUE 'U' (declaree dans la directive INCONNUE)
         INTEGRAND 'UV'
         DONNEE 'F' FONCTION REELLE
```

ou la fonction **g** dans l'intégrale $\int_\Gamma \mathbf{g}(\sigma)\,\psi(\sigma)\,d\sigma$ déclarée par

```
CALCUL SUR LE DOMAINE 'GAMMA'
       TERME ELEMENTS FINIS 'NEUGAM'
         INCONNUE 'PHI'
         INTEGRAND 'V'
         DONNEE 'G' FONCTION COMPLEXE DONNEE ASSOCIEE 'k'
```

On définira aussi les données de type fonction associées aux conditions essentielles, ou aux termes valeurs nodales, par exemple déclarées par

```
CALCUL SUR LE DOMAINE 'OMEGA' (d'une solution exacte)
       TERME VALEURS NODALES 'PhiExa' DE TYPE 'F'
         INCONNUE 'PHI'
         DONNEE 'SOLEXA' FONCTION COMPLEXE DONNEE ASSOCIEE 'k'
```

qui indique le calcul de $\varphi(n_j)$ aux nœuds n_j du domaine Ω ;

ou encore la donnée de type gradient d'une fonction pour calculer par exemple un terme valeurs nodales dérivée normale, déclaré par

```
CALCULSUR LE DOMAINE 'GAMMA'
      TERME VALEURS NODALES 'dPhidn' DE TYPE 'N.F'
        INCONNUE 'PHI'
        DONNEE 'GradPhi' FONCTION COMPLEXE DONNEE ASSOCIEE 'k'
```

auquel cas la procédure devra retourner un vecteur (tableau à NDIM composantes contenant les valeurs de $\overrightarrow{\mathrm{grad}}\,\varphi$) qui indique le calcul de $\dfrac{\partial\varphi}{\partial\nu}(n_j) = \overrightarrow{\mathrm{grad}}\,\varphi(n_j)\cdot\vec{\nu}_{n_j}$ aux nœuds n_j du domaine de bord Γ où $\vec{\nu}_{n_j}$ désigne la normale unitaire.

Ces fonctions, qui peuvent être scalaires, vectorielles ou tensorielles, à valeurs réelles ou complexes doivent être fournies par l'utilisateur de l'application. Elles sont toutes regroupées dans une même procédure FCTRM dont la syntaxe est la suivante

• SUBROUTINE FCTRM (NDIM,POINT,NOMFCT,TYPFCT,NODONA,TYDONA,TYPDOA,MCASSO
,IST,RST,CST,*RESULT*,*CESULT*)

et c'est en particulier la valeur de la chaîne NOMFCT qui permet de retourner la ou les valeurs adéquates.

Déclarations

CHARACTER*(*)	NOMFCT,TYPFCT,NODONA,TYDONA,TYPDOA
INTEGER	NDIM,MCASSO,IST(*)
REAL	POINT(*),RST(*),RESULT(*)
COMPLEX	CST(*),CESULT(*)

Définition des arguments

NDIM dimension d'espace ;

POINT tableau de dimension
– NDIM lors du calcul des intégrales élémentaires en dimension d'espace (intégrales de volume en 3D, de surface en 2D) ou lors du calcul des termes valeurs nodales. Ce tableau contient les coordonnées du point où la fonction doit être évaluée (image d'un point de la formule de quadrature dans l'élément courant, par exemple) ;
– 2*NDIM+1 lors du calcul des intégrales élémentaires de bord (intégrales de surface en 3D, de ligne en 2D). Les NDIM premiers adresses du tableau contiennent les coordonnées du point où la fonction doit être évaluée, la NDIM+1$^{\text{ème}}$ la valeur de l'élément de surface ou de ligne, et les NDIM suivantes les composantes de la normale sur le bord, extérieure à l'élément. Le vecteur normal unitaire peut alors être obtenu par division de ces composantes par POINT(NDIM+1). Ces renseignements sont fournis pour effectuer certains calculs d'intégrales ou de termes de type valeurs nodales faisant intervenir (une composante de) la normale.

NOMFCT est le nom d'une des données de type fonction fourni par l'utilisateur et qui permet de différencier les *'formules'* programmées dans la procédure.

TYPFCT est le type de *déclaration* (réel ou complexe) tel qu'il a été fourni par l'utilisateur lors de la déclaration de la donnée fonction dont le nom est transmis par l'argument précédent ;

NODONA est le nom de l'éventuelle donnée associée à la donnée fonction dont le nom est transmis par l'argument NOMFCT ;

TYDONA est le type de *représentation* (constante ou tableau) de la donnée dont le nom est transmis par l'argument NODONA tel qu'il a été fourni par l'utilisateur.

TYPDOA est le type de *déclaration* (entier, réel ou complexe) de la donnée dont le nom est transmis par l'argument NODONA tel qu'il a été fourni par l'utilisateur.

MCASSO est l'adresse de la donnée (constante ou tableau) associée (NODONA) dans le super-tableau du type adéquat :
IST pour une donnée entière (si TYPDOA(1:6)='ENTIER'),
RST pour une donnée réelle (si TYPDOA(1:4)='REEL') et
CST pour une donnée complexe (si TYPDOA(1:8)='COMPLEXE').

IST super-tableau numérique version entière : IST(MCASSO) contient la valeur de la constante associée ou le premier élément du tableau associé si TYPDOA(1:1)="E".

RST super-tableau numérique version réelle : RST(MCASSO) contient la valeur de la constante associée ou le premier élément du tableau associé si TYPDOA(1:1)="R".

CST super-tableau numérique version complexe : CST(MCASSO) contient la valeur de la constante associée ou le premier élément du tableau associé si TYPDOA(1:1)="C".

RESULT valeur réelle ou vecteur réel retourné par la fonction dans le cas réel (TYPFCT(1:2)='RE').

CESULT valeur ou vecteur complexe retourné par la fonction dans le cas complexe (TYPFCT(1:2)='CO').

⚠ ⚠ Les paramètres formels de la liste : `NOMFCT`, `NODONA` et si nécessaire `TYPFCT`, `TYDONA`, `TYPDOA` permettent à l'utilisateur de distinguer les différentes données `FONCTION` de l'application.

Procédures utilisant `FCTRM` : librairie calelem et module caltrm

`CALINT` calcul des intégrales élémentaires de volume ;
`CALINB` calcul des intégrales élémentaires de bord ;
`CALIND` calcul des intégrales élémentaires de bord avec dérivées ;
`VNCALC` calcul des termes valeurs nodales ou des données des conditions aux limites essentielles (Dirichlet ou transmission).

EXEMPLE

```
      SUBROUTINE FCTRM  (NDIM,POINT,NOMFCT,TYPFCT,NODONA,TYDONA
     &                  ,TYPDOA,MCASSO,IST,RST,CST,RESULT,CESULT)
*************************************************************************
  Auteur : D.Martin (Fevrier/Mars 1999)
  Derniere modification : D.Martin (8 Mars 1999)
*
*-- Arguments --
*
*************************************************************************
      CHARACTER*(*) NOMFCT,TYPFCT,NODONA,TYDONA,TYPRAS
      INTEGER       NDIM,IST(*),MCASSO
      REAL          POINT(*),RST(*),RESULT(*)
      COMPLEX       CST(*),CESULT(*)
*
      REAL          NORMAL(2),CALCUL(3)
*-----------------------------------------------------------------------
      IF (NOMFCT(1:6).EQ.'SOLEXA') THEN
         Solution exacte
         CALL SOLEXA (POINT,RESULT)
*
      ELSEIF (NOMFCT(1:5).EQ.'SECMB') THEN
*
         Rot_{-Beta}Rot_{Beta} E - Grad_{Beta}Div_{Beta} E + E
         CALL OPERAE (RST(MCASSO),POINT,RESULT)
*
      ELSEIF (NOMFCT(1:7).EQ.'RotBeta') THEN
*
         Rot_{Beta} E
         CALL ROTBTE (RST(MCASSO),POINT,RESULT)
*
      ELSEIF (NOMFCT(1:7).EQ.'DivBeta') THEN
*
         Div_{Beta} E
         CALL DIVBTE (RST(MCASSO),POINT,RESULT(1))
*
      ELSEIF (NOMFCT(1:3).EQ.'EsN') THEN
*
         CALL SOLEXA (POINT,CALCUL)
         RESULT(1)=PROSCA(2,CALCUL,POIN(NDIM+2))/POIN(NDIM+1)
*
       ELSEIF (NOMFCT(1:3).EQ.'EvN') THEN
*
         CALL SOLEXA (POINT,RESULT)
*
      ELSE
         CALL BAISE  ('*Fctrm (Ce2d3comp)* '
     &               //'Nom de fonction inconnue : '//NOMFCT(1:7))
      ENDIF
      END
```

```
      SUBROUTINE SOLEXA (POINT,E)
      REAL POINT(*),E(*)
      E(1)=EXP(POINT(1))
      E(2)=EXP(POINT(2))
      E(3)=0.
      END
```

```
      SUBROUTINE OPERAE (BETA,POINT,F)
      REAL BETA,POINT(*),F(*)
      F(1)= BETA*BETA*EXP(POINT(1))
      F(2)= BETA*BETA*EXP(POINT(2))
      F(3)=0.
      END
```

```
      SUBROUTINE ROTBTE (BETA,POINT,ROT)
      REAL BETA,POINT(*),ROT(*)
      ROT(1)=-BETA*EXP(POINT(2))
      ROT(2)= BETA*EXP(POINT(1))
      ROT(3)=0.
      END
```

```
      SUBROUTINE DIVBTE (BETA,POINT,DIV)
      REAL BETA,POINT(*),DIV
      DIV=EXP(POINT(1))+EXP(POINT(2))
      END
```

2. Données de type 'gradient de fonction'

Le calcul des termes de type valeurs nodales dérivée normale, par exemple

$$\partial\phi_w/\partial n_\Gamma(n_j)$$

aux nœuds n_j du domaine (de bord) Γ, déclarés sous la forme :

```
CALCUL SUR LE DOMAINE 'GAMMA'
       TERME VALEURS NODALES 'dPhidn' 'DF/DN'
        INCONNUE 'PHI'
          DONNEE 'GradPHIw' FONCTION COMPLEXE DONNEE ASSOCIEE 'k'
```

nécessite le calcul des dérivées de la fonction dont on veut calculer la dérivée normale aux nœuds d'un domaine. Ces dérivées doivent être fournies par l'utilisateur de l'application, soit selon la forme décrite ci-dessus (FCTRM), soit regroupées dans une même procédure DFCTRM dont les arguments sont en tout point analogues à ceux de FCTRM :

- SUBROUTINE DFCTRM (NDIM,POINT,NOMFCT,TYPFCT,NODONA,TYDONA,TYPDOA,MCASSO
  ,IST,RST,CST, *RESULT,CESULT*)

△ Dans la version actuelle du code, qui permet de définir d'autres types de valeurs nodales (en particulier pour des inconnues vectorielles) la procédure FCTRM peut remplacer la procédure DFCTRM pour le calcul de valeurs nodales dérivée normale : il est préférable de programmer le gradient de la fonction dans FCTRM et d'invoquer le calcul d'un terme de valeurs nodales de type 'N.F' (produit scalaire avec le vecteur normal).

– Mai 2000 –

Application Helmholtz 3D

On trouvera dans cette annexe, un exemple complet d'application de MÉLINA : la résolution du problème de la diffraction d'une onde acoustique par un corps parfaitement réfléchissant.

1. Le problème

Il s'agit de trouver une fonction φ, définie dans le domaine non borné $\check{\Omega}$ complémentaire d'un compact de frontière Γ ($\varphi \in \mathrm{H}^1_{loc}(\check{\Omega})$), solution du problème

$$(\check{P}) \qquad \begin{cases} \Delta\varphi + k^2\,\varphi = 0 & \text{dans } \check{\Omega} \\ \dfrac{\partial\varphi}{\partial\nu} = g & \text{sur } \Gamma \\ \dfrac{\partial\varphi}{\partial r} - ik\varphi \in L^2(\check{\Omega}) \end{cases}$$

où k représente le nombre d'onde, $g = -\partial\phi_w/\partial n_\Gamma$ où ϕ_w est le potentiel d'une onde incidente, ν la normale unitaire extérieure à $\check{\Omega}$, et r la distance à l'origine.

1.1. Formulation variationnelle

On note G la fonction de Green du problème telle qu'elle est programmée (la multiplication par $-\frac{1}{4\pi}$ apparaît dans les termes de couplage) :

$$G(M,P) = \frac{\mathrm{e}^{ik\|\overrightarrow{MP}\|}}{\|\overrightarrow{MP}\|},$$

et λ un nombre complexe (avec $\mathrm{Im}(\lambda \neq 0)$. On considère la frontière de couplage $\Sigma \subset \overline{\check{\Omega}}$ et telle que $d(\Gamma,\Sigma) > 0$ (par exemple si Γ est une sphère de rayon R, on choisit pour Σ une sphère concentrique de rayon $R+h$). La formulation variationnelle standard ($F.V.S.$) en domaine borné de ce problème s'écrit

Trouver $\varphi \in \mathrm{H}^1(\Omega)$, tel que $\forall\psi \in \mathrm{H}^1(\Omega)$

$$\int_\Omega \nabla\varphi.\nabla\psi\,d\omega - k^2\int_\Omega \varphi\psi\,d\omega + \lambda\int_\Sigma \varphi\psi\,d\sigma$$
$$+\frac{1}{4\pi}\int_\Sigma \psi(M)\int_\Gamma \varphi(P)\left(\frac{\partial}{\partial\nu_M}+\lambda\right)\frac{\partial G}{\partial\nu_P}(M,P)d\gamma(P)\,d\sigma(M) =$$
$$= \int_\Gamma g\,\psi\,d\gamma \;+\frac{1}{4\pi}\int_\Sigma \psi(M)\int_\Gamma g(P)\left(\frac{\partial}{\partial\nu_M}+\lambda\right)G(M,P)\,d\gamma(P)\,d\sigma(M)$$

1.2. Problème annexe – Préconditionnement

On considère le problème annexe suivant (dit 'avec conditionnement de rayonnement à distance finie')

$$(P_p) \qquad \begin{cases} \Delta\varphi_p + k^2\,\varphi_p = 0 & \text{dans } \Omega \\ \dfrac{\partial\varphi_p}{\partial\nu} = g & \text{sur } \Gamma \\ \dfrac{\partial\varphi_p}{\partial\nu} - ik\varphi_p = 0 & \text{sur } \Sigma \end{cases}$$

dont la formulation variationnelle s'écrit :

$$\text{Trouver } \varphi_p \in \mathrm{H}^1(\Omega), \text{ tel que } \forall \psi \in \mathrm{H}^1(\Omega)$$
$$\int_\Omega \nabla\varphi_p.\nabla\psi\, d\omega - k^2 \int_\Omega \varphi_p\psi\, d\omega - ik \int_\Sigma \varphi_p\psi\, d\sigma = \int_\Gamma g\,\psi\, d\gamma$$

et pour lequel il est connu que la solution φ_p, qui dépend de Σ, converge (dans $\mathrm{H}^1_{loc}(\check{\Omega})$) vers celle de $(\check{P})$ lorsque "$\Sigma \mapsto \infty$".
La matrice de ce problème annexe est factorisée $L_1 \cdot U$ incomplètement (elle est complexe symétrique mais non hermitienne) et est utilisée comme matrice de préconditionnement lors de la résolution du problème à l'aide de la méthode du Gradient Biconjugué préconditionné avec comme point de départ des itérations φ_p.

1.3. Une solution exacte

Lorsque l'obstacle est une sphère de rayon R, centrée à l'origine et que l'onde incidente est une onde plane, c'est-à-dire $g = -\partial\phi_w/\partial n_\Gamma$ avec $\phi_w(x,y,z) = \mathrm{e}^{ikx}$, la solution exacte s'exprime comme le développement en série de fonctions de Bessel sphériques et de polynômes de Legendre suivant

$$\varphi_{ex}(r,\theta) = \sum_0^\infty a_n(kR) h_n^{(1)}(kr) P_n(\cos\theta) \quad \text{où} \quad a_n(t) = -i^n(2n+1)\frac{j_n'(t)}{h_n^{(1)'}(t)}$$

où P_n est le polynôme de Legendre de degré n
$h_n^{(1)} = j_n + iy_n$ est la fonction de Bessel sphérique de 3$^{\text{ème}}$ espèce et d'ordre n ;
j_n est la fonction de Bessel sphérique de 1$^{\text{ère}}$ espèce et d'ordre n ;
y_n est la fonction de Bessel sphérique de 2$^{\text{ème}}$ espèce et d'ordre n.

1.4. Les Notations

On notera dans le programme principal et les procédures 'utilisateur' :

`'k'`	le nombre d'onde k ;
`'-k**2'`	la constante de valeur $-k^2$ calculée à partir de la valeur de `'k'` ;
`'ik'`	la constante de valeur ik calculée à partir de la valeur de `'k'` ;
`'LAMBDA'`	la constante complexe de couplage λ ;
`'1/4PI'`	la constante $\frac{1}{4\pi}$;
`'EPSIL'`	la constante ε seuil de convergence des itérations de gradient ;
`'MAXTER'`	la constante entière nombre maximum d'itérations de gradient ;
`'RIGID',1`	le terme de rigidité sur le domaine Ω : $\int_\Omega \nabla\varphi.\nabla\psi\, d\omega$;
`'MASSE',1`	le terme de masse sur Ω : $\int_\Omega \varphi\,\psi\, d\omega$;
`'GDELTA',1`	le terme de masse sur la frontière de couplage Σ : $\int_\Sigma \varphi\,\psi\, d\sigma$;
`'PDELTA',1`	le terme de masse sur Γ : $\int_\Gamma \varphi\,\psi\, d\gamma$; ce terme n'apparaît pas explicitement dans la formulation variationnelle mais sera utilisé pour le calcul du terme éléments finis de second membre et les termes de couplage ;
`'DPHIN',1`	le terme contenant les valeurs de g sur Γ ;
`'NEUGAM',1`	le terme de second membre éléments finis sur Γ : $\int_\Gamma g\,\psi\, d\gamma$; il sera calculé comme produit de `'PDELTA',1` et `'DPHIN',1` .
`'MATGRN',1`	la matrice des noyaux de Green du terme de couplage de premier membre : $\big(\partial/\partial\nu_M + \lambda\big)\big(\partial/\partial\nu_P G(M,P)\big)_{M\in\Sigma, P\in\Gamma}$
`'MATGRE',1`	la matrice des noyaux de Green du terme de couplage de second membre : $\big(\partial/\partial\nu_M + \lambda\big)\big(G(M,P)\big)_{M\in\Sigma, P\in\Gamma}$;
`'COUPLA',1`	le terme de couplage de premier membre : $\int_\Sigma \psi(M) \int_\Gamma \varphi(P)\Big(\partial/\partial\nu_M + \lambda\Big)\partial/\partial\nu_P G(M,P) d\gamma(P)\, d\sigma(M)$ il sera calculé comme produit de `'GDELTA',1,'MATGRN',1` et `'PDELTA',1` ;

'COUPLS',1 le terme de couplage de second membre :
$\int_\Sigma \psi(M) \int_\Gamma g(P) \Big(\partial/\partial\nu_M + \lambda \Big) G(M,P)\, d\gamma(P)\, d\sigma(M)$
il sera calculé comme produit de 'GDELTA',1,'MATGRE',1 et 'NEUGAM',1.

Pour les termes assemblés, on note

'ADEUV',1 la forme bilinéaire $\int_\Omega \nabla\varphi.\nabla\psi - k^2\,\varphi\,\psi\,d\omega$;
'PRECON',0 la matrice de préconditionnement $\int_\Omega \nabla\varphi.\nabla\psi - k^2\,\varphi\,\psi\,d\omega - ik\int_\Sigma \varphi\,\psi\,d\sigma$;
'MATRIS',1 la matrice du système ;
'SECMBR',1 le second membre.

On note enfin

'PHICAL',1 la solution (φ) du problème dans Ω,
'PHIEXA',1 la solution exacte ;
'PHIREC',1 la solution recalculée à partir de PHICAL' par reconstitution par représentation intégrale sur la frontière Σ.

La fonction de Green est programmée au facteur $1/4\pi$ près, ce qui explique les constantes affectées aux termes de couplage et aux termes de calcul de la représentation intégrale de la solution.

1.5. La procédure 'Utilisateur' : FCTRM

Dans la procédure FCTRM sont programmés

- le calcul du gradient de la fonction $\phi_w = \mathrm{e}^{ikx}$, donnée fonction 'DPHIN' à laquelle est associée la constante k
- le calcul de la solution exacte, donnée fonction 'PHIEXA' programmée dans la procédure SLXHLM.

2. Le fichier de données des directives

```
GEOMETRIE LECTURE SUR UNITE 'quartsphere.mai'
! Le quart de sphere y>0, z>0
   IMPRESSION DE NIVEAU 2
!
! Description de l'inconnue
!
INCONNUE 'PHI' DE TYPE SCALAIRE (INTERPOLATION ISOPARAMETRIQUE)
!
! Description des termes de la formulation variationnelle
!
CALCUL SUR LE DOMAINE 'OMEGA'
   TERME ELEMENTS FINIS 'RIGID'
      INCONNUE 'PHI' INTEGRAND 'GRADGRAD'
   TERME ELEMENTS FINIS 'MASSE'
      INCONNUE 'PHI' INTEGRAND 'UV'
      DONNEE '-K**2' CONSTANTE REELLE
   TERME VALEUR NODALE 'PHIEXA' DE TYPE 'F' ! solution exacte
      INCONNUE 'PHI'
      DONNEE 'PHIEXA' FONCTION COMPLEXE DONNEE ASSOCIEE 'K'
CALCUL SUR LE DOMAINE 'GAMMA'
   NORMALES ASSEMBLEES ! pour le calcul des noyaux de Green
   COORDONNEES DES NOEUDS
   TERME ELEMENTS FINIS 'PDELTA'
      INCONNUE 'PHI' INTEGRAND 'UV'
   TERME VALEUR NODALE 'DPHIN' DE TYPE 'N.F' ! derivee normale de Phiw
      INCONNUE 'PHI'
      DONNEE 'DPHIN' FONCTION COMPLEXE DONNEE ASSOCIEE 'K'
CALCUL SUR LE DOMAINE 'SIGMA'
   NORMALES ASSEMBLEES ! pour le calcul des noyaux de Green
   COORDONNEES DES NOEUDS
   TERME ELEMENTS FINIS 'GDELTA'
      INCONNUE 'PHI' INTEGRAND 'UV'
      DONNEE 'LAMBDA' CONSTANTE COMPLEXE
!
! Description des symetries du probleme pour les termes de couplage
!
SYMETRIE EN 'Y' EN 'Z'
!
! Declaration des structure d'assemblage des termes
!
ASSEMBLAGE DU TERME 'PRECON' NIVEAU 0
TERMES 'ADEUV' ET 'GDELTA'
!
ASSEMBLAGE DU TERME 'MATRIS' NIVEAU 1
TERMES 'ADEUV' 'GDELTA' ET 'COUPLA'
!
ASSEMBLAGE DU TERME 'SECMBR' NIVEAU 1
TERMES 'NEUGAM' ET 'COUPLS'
!
FIN !des directives generales INCONNUE, CALCUL, SYMETRIE
!
! Donnee des constantes
!
DONNEE DE NOM 'K' 1.0 'LAMBDA' 0. 1.
              '1/4PI' CONSTANTE REELLE $0.25/PI$
              'EPSIL' CONSTANTE REELLE 1.E-06 !Seuil d'arret des iterations
              'MAXTER' CONSTANTE ENTIERE 20 !Nombre maximum d'iterations
              'IMPRES' CONSTANTE ENTIERE 0 !Impressions courantes
              'IMPSOL' CONSTANTE ENTIERE 0 !Impression de la solution
FIN (du jeu de donnee)
```

3. Le programme principal

```
*
                          PROGRAM HELMZ3
*
*     Declarations globales des super-tableaux
      CHARACTER     AST,TYPE,CARAC
      INTEGER       MOTMXR,NBCHMX,IMPPAL
      PARAMETER    (MOTMXR=2000000,NBCHMX=80000,IMPPAL=6)
      COMMON/      /RST(MOTMXR) /STCHAR/AST(NBCHMX)
      COMPLEX       KC
      REAL          KR
*
*     Initialisation de l'allocation dynamique et
*     creation des structures de donnees.
      CALL INITIE (NBCHMX,MOTMXR)                                !Initial
*     Lecture du fichier de geometrie (maillage)
      CALL LCGEOM                                                !Lecgeom
*     Lecture des directives de definition des inconnues variationnelles
*     et de construction des termes elements-finis
      CALL LCDIRE                                                !Lecdire
*     Lecture des donnees scalaires
      CALL LCDONN                                                !Lecdire
*     Numerotations globales en noeuds (pour toutes les interpolations)
      CALL NUMNEU                                                !Glonum
*     Distribution des coordonnees sur les domaines
      CALL CPTDOM                                                !Cordom
*     Calcul des termes elements-finis sur les domaines
      CALL CALKEF                                                !Caldom
*
*     Definition des constantes pour les impressions
      CALL GETCST ('IMPRES',TYPE,IMPRES,KR,KC,CARAC)             !Sdexplo
      CALL GETCST ('IMPSOL',TYPE,IMPSOL,KR,KC,CARAC)             !Sdexplo
*     Definition des constantes pour les iterations de gradient
      CALL GETCST ('EPSIL',TYPE,IBID,EPSIL,KC,CARAC)             !Sdexplo
      CALL GETCST ('MAXTER',TYPE,MAXTER,KR,KC,CARAC)             !Sdexplo
*     Definition des constantes -K**2 = - K * K et -iK = - i * K
      CALL GETCST ('K',TYPE,IBID,KR,KC,CARAC)                    !Sdexplo
      CALL PUTCST ('-K**2','REEL',IBID,-KR*KR,KC,CARAC)          !Sdexplo
      CALL PUTCST ('-iK','COMPLEXE',IBID,KR,CMPLX(0.,-KR),CARAC) !Sdexplo
*
*     Calcul de la matrice Somme(Omega) gradu.gradv -k2 uv dx
*
      CALL CLTERM ('RIGID',1,' ','MASSE',1,' ','ADEUV',1,IMPRES) !Assembl
*
*     Assemblage de la matrice de preconditionnement
*
      CALL DOASTR ('GDELTA',1,'-iK','CONSTANTE',0,'COMPLEXE')    !Sdexplo
      CALL ASMTRM ('PRECON',0,IMPRES)                            !Assembl
*
*     Second membre 'Elements Finis'
*
      CALL MATVEC ('PDELTA',1,'DPHIN',1,'NEUGAM',1,IMPRES)       !Assembl
      CALL CHRONO ('Assemblage E.F.',IMPPAL)                     !Utilite
*
```

```
*     Factorisation incomplete de la matrice de preconditionnement
*
      CALL FILU ('PRECON',0,'S','PRECON',0,IMPRES)                    !Syslin
*     Resolution incomplete du probleme a distance finie
      CALL FASV ('PRECON',0,'NEUGAM',1,'PHICAL',1,IMPSOL)             !Syslin
      CALL CHRONO ('Factorisation LU incomplete et resolution',IMPPAL)!Utilite
*
*     Ici sont deja connus ADEUV,1 - PDELTA,1 - GDELTA,1 - NEUGAM,1
*
      CALL PTITRE ('Formulation FOURIER standard')                    !Utilite
*
*     Calcul des noyaux de Green (Fourier)
*
      CALL CALNOY ('SIGMA','GAMMA',1,'FOURIER',1,'K','LAMBDA'
                  ,'MATGRE',1,'MATGRN',1)                             !Calgre
      CALL CHRONO ('Fonction de Green',IMPPAL)                        !Utilite
*
*     Calcul des termes de Couplage
*
      CALL COUMAT ('GDELTA',1,'MATGRN',1,1,'A MORT','PDELTA',1
                  ,'COUPLA',1,1,IMPRES)                               !Couplag
      CALL COUVEC ('GDELTA',1,'MATGRE',1,1,'A MORT','NEUGAM',1
                  ,'COUPLS',1,IMPRES)                                 !Couplag
      CALL CHRONO ('Termes de Couplage',IMPPAL)                       !Utilite
*
*     Assemblage des matrices E.F. standard et des termes de Couplage
*
      CALL DOASTR ('GDELTA',1,'LAMBDA','CONSTANTE',0,'COMPLEXE')      !Sdexplo
      CALL DOASTR ('COUPLA',1,'1/4PI','CONSTANTE',0,'REEL')           !Sdexplo
      CALL ASMTRM ('MATRIS',1,IMPRES)                                 !Assembl
*
*     Assemblage des seconds membres E.F. standard et de Couplage
*
      CALL DOASTR ('COUPLS',1,'1/4PI','CONSTANTE',0,'REEL')           !Sdexplo
      CALL ASVTRM ('SECMBR',1,IMPRES)                                 !Assembl
      CALL TBTUER ('COUPLS',1)                                        !Alodyn
      CALL CHRONO ('Assemblage du systeme',IMPPAL)                    !Utilite
*
*     Resolution par Gradient biconjugue preconditionne
*
      CALL GBCT ('MATRIS',1,'SECMBR',1,'PRECON',0,EPSIL,MAXTER
                  ,'PHICAL',1,IMPRES,*100)                            !Syslin
100   CALL CHRONO ('Resolution - Gradient Biconjugue',IMPPAL)         !Utilite
*
*     Comparaison avec une solution exacte
*
      CALL DFTERM ('PHICAL',1,'PHIEXA',1,IMPSOL*5)                    !Sdexplo
      CALL BYE ('C''est fini')                                        !Utilite
      END
```

4. La procédure 'utilisateur' FCTRM

```
*
      SUBROUTINE FCTRM (NDIM,POINT,NOMFCT,TYPFCT,NODONA,TYDONA
                       ,TYPDOA,MCASSO,IST,RST,CST,RESULT,CESULT)
*
* Auteur : D.Martin (Janvier 1992)
* Derniere modification : D.Martin le 14 Mars 1992
*
*
      CHARACTER*(*) NOMFCT,TYPFCT,NODONA,TYDONA,TYPDOA
      INTEGER       IST(*),MCASSO
      REAL          RST(*),POINT(*),RESULT(*)
      COMPLEX       CST(*),CESULT(*)
*
      REAL          CORM(3)
      COMPLEX       D1EM(3),K,SLXHLM
      CHARACTER     NAMDON*16,NAMFCT*16
*
      IF (TYDONA(:4).NE.'CONS') THEN
         NAMDON=NODONA
         NAMFCT=NOMFCT
         CALL BAISE ('*Fctrm (HELMZ3D)* La Donnee '//NAMDON(:10)
                     //' pour la fonction '//NAMFCT(:10)
                     //' n''est pas une CONSTANTE')
      ENDIF
      IF (TYPFCT(:4).NE.'COMP') THEN
         NAMFCT=NOMFCT
         CALL BAISE ('*Fctrm (HELMZ3D)* La Fonction '//NAMFCT(:10)
                     //' n''est pas a valeur COMPLEXE')
      ENDIF
      IF (TYPDOA(:4).EQ.'REEL') THEN
         K = RST(MCASSO)
      ELSEIF (TYPDOA(:4).EQ.'COMP') THEN
         K = CST(MCASSO)
      ENDIF
      IF (NOMFCT(:5).EQ.'DPHIN') THEN
         CESULT(1)   =-(0.,1.)*K*EXP ((0.,1.)*K*POINT(1))
         CESULT(2)   = 0.
         CESULT(NDIM)= 0.
      ELSEIF (NOMFCT(:6).EQ.'PHIEXA') THEN
         CESULT(1) = SLXHLM (K,POINT)
      ENDIF
                                                          END !Fctrm
```

– Mai 2000 –

Annexe 1. Intégrands des termes 'Éléments finis'

N.B. À l'instant où nous mettons sous presse, cette liste peut ne pas être complète !

Dans ces tableaux

- D désigne un domaine quelconque (volume, surface, ligne) dont x est le point générique.
- $\int_D P_1 u \, P_2 v \, dx$ désigne une intégrale de la forme $\int_D [f(x)] \, P_1 u(x) \, P_2 v(x) \, dx$ où P_1 et P_2 sont deux polynômes différentiels et $[f(x)]$ représente un éventuel coefficient variable (voir la définition d'une donnée associée au terme).
- u correspond à l'*inconnue en colonne*.
- v représente une quelconque fonction-test (dite *inconnue en ligne*).
- 'α' ou 'β' est l'un des caractères 1, 2 ou 3 et ∂_α représente la dérivée par rapport à la $\alpha^{ème}$ variable d'espace.
- ∂_n représente la dérivée normale pour une intégrale de bord.

A.1.1. Intégrales pour inconnues scalaires

Nom générique (mot-clé)	Type d'intégrale
V	$\int_D v \, dx$
D'α'V	$\int_D \partial_\alpha v \, dx$
UV	$\int_D uv \, dx$
D'α'UV	$\int_D \partial_\alpha uv \, dx$
UD'α'V	$\int_D u \partial_\alpha v \, dx$
D'α'UD'β'V	$\int_D \partial_\alpha u \partial_\beta v \, dx$
GRADGRAD	$\int_D \overrightarrow{\text{grad}}\, u \cdot \overrightarrow{\text{grad}}\, v \, dx$
DNV	$\int_D \partial_n v \, dx$
UDNV	$\int_D u \partial_n v \, dx$
DNUV	$\int_D \partial_n uv \, dx$
DNUDNV	$\int_D \partial_n u \partial_n v \, dx$

A.1.2. Intégrales pour inconnues vectorielles

Dans le tableau suivant

- $\vec{n}$ représente le vecteur normal unitaire extérieur domaine de calcul D.
- u_ℓ désigne la $\ell^{ème}$ composante de $\vec{u}$ et $u_{\ell,\alpha}$ la dérivée partielle $\dfrac{\partial u_\ell}{\partial x_\alpha}$.

Nom générique (mot-clé)	Type d'intégrale
`V = F.V`	$\int_D \overrightarrow{f} \cdot \overrightarrow{v}\, dx$
`UV = U.V`	$\int_D \vec{u} \cdot \vec{v}\, dx$
`U^V`	$\int_D \vec{u} \wedge \vec{v}\, dx = \int_D u_1 v_2 - u_2 v_1\, dx$ (2d)
`GRADGRAD`	$\int_D \overrightarrow{\text{grad}}\, \overrightarrow{u} \cdot \overrightarrow{\text{grad}}\, \overrightarrow{v}\, dx$
`ROTROT`	$\int_D \text{rot}\ \vec{u}\ \text{rot}\ \vec{v}\, dx$ (2d), $\int_D \overrightarrow{\text{rot}}\, \overrightarrow{u} \cdot \overrightarrow{\text{rot}}\, \overrightarrow{v}\, dx$ (3d)
`DIVDIV`	$\int_D \text{div}\ \vec{u}\ \ \text{div}\ \vec{v}\, dx$
`EIJEIJ`	$\int_D \varepsilon_{ij}(\vec{u})\, \varepsilon_{ij}(\vec{v})\, dx$ où $\varepsilon_{ij}(\vec{u}) = (u_{i,j} + u_{j,i})/2$
`UDIVV`	$\int_D u\ \text{div}\ \vec{v}\, dx$
`DIVUV`	$\int_D \text{div}\ \vec{u}\ v\, dx$
`UGRADV`	$\int_D \overrightarrow{u} \cdot \overrightarrow{\text{grad}}\, v\, dx$
`GRADUV`	$\int_D \overrightarrow{\text{grad}}\, u \cdot \overrightarrow{v}\, dx$
`ROTROTBT0`(1)	$\int_D u_{2,1}v_{2,1} + u_{1,2}v_{1,2} - u_{2,1}v_{1,2} - u_{1,2}v_{2,1} + u_{3,1}v_{3,1} + u_{3,2}v_{3,2}\, dx$ (1_1)
`ROTROTBT1`(1)	$\int_D u_{1,1}v_3 + u_3 v_{1,1} + u_{2,2}v_3 + u_3 v_{2,2}\, dx$ (1_2)
`ROTROTBT2`(1)	$\int_D u_1 v_1 + u_2 v_2\, dx$ (1_3)
`DIVDIVBT0`(1)	$\int_D u_{1,1}v_{1,1} + u_{1,1}v_{2,2} + u_{2,2}v_{1,1} + u_{2,2}v_{2,2}\, dx$ (1_4)
`DIVDIVBT1`(1)	$\int_D u_1 v_{3,1} + u_{3,1}v_1 + u_2 v_{3,2} + u_{3,2}v_2\, dx$ (1_5)
`DIVDIVBT2`(1)	$\int_D u_3 v_3\, dx$ (1_6)
`EIJEIJBT0`(2)	$\int_D \varepsilon_{ij}^{\beta_0}(\vec{u})\, \varepsilon_{ij}^{\beta_0}(\vec{v})\, dx$ où $\varepsilon_{ij}^{\beta_0}(\vec{u}) = (u_{i,j} + u_{j,i})/2$ avec $u_{i,3} = 0$
`U`α`V`α(3)	$\int_D u_\alpha v_\alpha\, dx$
`U`α`,`α`V`α`,`α(3)	$\int_D u_{\alpha,\alpha} v_{\alpha,\alpha}\, dx$
`Ui,i,Vj,j`(4)	$\int_D u_{1,1}v_{2,2} + u_{2,2}v_{1,1}\, dx$ (4_1)
`Ui,j,Vj,i`(4)	$\int_D (u_{1,2} + u_{2,1})(v_{1,2} + v_{2,1})\, dx$ (4_2)

$(RotRot)$ Matrice élémentaire symétrique dont chaque 'coefficient'$_{ij}$ a pour intégrand :

$$\begin{pmatrix} w_{i,2}w_{j,2} & -w_{i,2}w_{j,1} \\ -w_{i,1}w_{j,2} & w_{i,1}w_{j,1} \end{pmatrix}^{(2d)} \quad \begin{pmatrix} w_{i,2}w_{j,2}+w_{i,3}w_{j,3} & -w_{i,2}w_{j,1} & -w_{i,3}w_{j,1} \\ -w_{i,1}w_{j,2} & w_{i,3}w_{j,3}+w_{i,1}w_{j,1} & -w_{i,3}w_{j,2} \\ -w_{i,1}w_{j,3} & -w_{i,2}w_{j,3} & w_{i,1}w_{j,1}+w_{i,2}w_{j,2} \end{pmatrix}^{(3d)}$$

$(DivDiv)$ Matrice élémentaire symétrique dont chaque 'coefficient'$_{ij}$ a pour intégrand :

$$\begin{pmatrix} w_{i,1}w_{j,1} & w_{i,1}w_{j,2} \\ w_{i,2}w_{j,1} & w_{i,2}w_{j,2} \end{pmatrix}^{(2d)} \qquad \begin{pmatrix} w_{i,1}w_{j,1} & w_{i,1}w_{j,2} & w_{i,1}w_{j,3} \\ w_{i,2}w_{j,1} & w_{i,2}w_{j,2} & w_{i,2}w_{j,3} \\ w_{i,3}w_{j,1} & w_{i,3}w_{j,2} & w_{i,3}w_{j,3} \end{pmatrix}^{(3d)}$$

(1) Intégrales particulières pour la modélisation 2d des équations de Maxwell avec dépendance harmonique en temps et en la dernière variable d'espace (inconnue vectorielle à 3 composantes).

Matrices élémentaires symétriques dont chaque 'coefficient'$_{ij}$ a pour intégrand :

$$\begin{pmatrix} w_{i,2}w_{j,2} & -w_{i,2}w_{j,1} & 0 \\ -w_{i,1}w_{j,2} & w_{i,1}w_{j,1} & 0 \\ 0 & 0 & \begin{matrix} w_{i,1}w_{j,1} \\ +w_{i,2}w_{j,2} \end{matrix} \end{pmatrix}^{(1_1)} \quad \begin{pmatrix} 0 & 0 & w_iw_{j,1} \\ 0 & 0 & w_iw_{j,2} \\ w_{i,1}w_j & w_{i,2}w_j & 0 \end{pmatrix}^{(1_2)} \quad \begin{pmatrix} w_iw_j & 0 & 0 \\ 0 & w_iw_j & 0 \\ 0 & 0 & 0 \end{pmatrix}^{(1_3)}$$

$$\begin{pmatrix} w_{i,1}w_{j,1} & w_{i,1}w_{j,2} & 0 \\ w_{i,2}w_{j,1} & w_{i,2}w_{j,2} & 0 \\ 0 & 0 & 0 \end{pmatrix}^{(1_4)} \quad \begin{pmatrix} 0 & 0 & w_{i,1}w_j \\ 0 & 0 & w_{i,2}w_j \\ w_iw_{j,1} & w_iw_{j,2} & 0 \end{pmatrix}^{(1_5)} \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & w_iw_j \end{pmatrix}^{(1_6)}$$

(2) Intégrales particulières pour la modélisation 2d des équations de l'élasticité avec dépendance harmonique en temps et en la dernière variable d'espace(inconnue vectorielle à 3 composantes).

Matrice élémentaire symétrique dont chaque 'coefficient'$_{ij}$ a pour intégrand :

(3) Matrices élémentaires symétriques dont chaque 'coefficient'$_{ij}$ a pour intégrand :

$$\begin{pmatrix} w_iw_j\delta_{\alpha,1} & 0 & 0 \\ 0 & w_iw_j\delta_{\alpha,2} & 0 \\ 0 & 0 & w_iw_j\delta_{\alpha,3} \end{pmatrix} \qquad \begin{pmatrix} w_{i,1}w_{j,1}\delta_{\alpha,1} & 0 & 0 \\ 0 & w_{i,2}w_{j,2}\delta_{\alpha,2} & 0 \\ 0 & 0 & w_{i,3}w_{j,3}\delta_{\alpha,3} \end{pmatrix}$$

(4) Intégrales particulières de la décomposition des matrices élémentaires de de l'élasticité 2d (matrices ci-dessus incluses).

Matrices élémentaires symétriques dont chaque 'coefficient'$_{ij}$ a pour intégrand :

$$\begin{pmatrix} 0 & w_{i,1}w_{j,2} \\ w_{i,2}w_{j,1} & 0 \end{pmatrix}^{(4_1)} \qquad \begin{pmatrix} w_{i,2}w_{j,2} & w_{i,2}w_{j,1} \\ w_{i,1}w_{j,2} & w_{i,1}w_{j,1} \end{pmatrix}^{(4_2)}$$

A.1.3. Intégrales de bord pour inconnues vectorielles

Nom générique (mot-clé)	Type d'intégrale
`V.N`	$\int_D \vec{v} \cdot \vec{n}\, dx$
`V^N (2d)`	$\int_D \vec{v} \wedge \vec{n}\, dx$
`U^N.V^N (2d)`	$\int_D \vec{u} \wedge \vec{n} \cdot \vec{v} \wedge \vec{n}\, dx$
`U.NV.N`	$\int_D \vec{u} \cdot \vec{n}\ \vec{v} \cdot \vec{n}\, dx$
`GRADGRADTG`	$\int_D \overrightarrow{\mathrm{grad}}_T\ \overrightarrow{u} \cdot \overrightarrow{\mathrm{grad}}_T\ \overrightarrow{v}\ dx = \int_D \overrightarrow{n} \wedge (\overrightarrow{\mathrm{grad}}\, u \wedge \overrightarrow{n}) \cdot \overrightarrow{n} \wedge (\vec{g}rad \wedge \overrightarrow{n})\, dx$
`NDIVU.V`	$\int_D \vec{n}\ \mathrm{div}\ \vec{u} \cdot \vec{v}\, dx$
`N^ROTU.V`	$\int_D \vec{n} \wedge \overrightarrow{\mathrm{rot}}\ \overrightarrow{u} \cdot \vec{v}\, dx$
`U.NV`	$\int_D \overrightarrow{u} \cdot \vec{n}\ v\, dx$

Annexe 2. Types des termes 'Valeurs nodales'

N.B. Cette liste peut ne pas être complète !

Dans le tableau suivant

- $\vec{n}$ représente le vecteur normal unitaire extérieur au nœud n_i du domaine (de bord) et n_α sa composante αème
- $f(n_i)$ (resp. $\vec{f}(n_i)$) représente les valeurs de la fonction scalaire f (resp. la fonction à valeur vectorielle $\vec{f}$) en tous les nœuds i d'un domaine. Ces valeurs sont retournées par la procédure 'utilisateur' FCTRM (voir la définition d'une donnée associée au terme).
- $\partial f/\partial n(n_i)$ représente les valeurs de la dérivée normale de la fonction f en tous les nœuds i d'un domaine (de bord).
- $f[n_i]$ ou ($\overrightarrow{f}[n_i]$) représente les valeurs de la fonction f aux nœuds i d'un domaine fournies par le tableau qui est la donnée associée au terme.

Nom générique (mot-clé)	Type de valeurs nodales	Type de l'inconnue
F(1)	$f(n_i)$ ou $\vec{f}(n_i)$	scalaire ou vectoriel
DF/DN(2)	$\partial f/\partial n(n_i)$	scalaire
TABF(3)	$f[n_i]$ ou $\vec{f}[n_i]$ est donné par tableau	scalaire ou vectoriel
NF	$\vec{n}f(n_i)$	scalaire
NαF	$n_\alpha f(n_i)$	scalaire
N.F(4)	$\overrightarrow{n}\,.\,\overrightarrow{f}(n_i)$	vectoriel
N^F	$\overrightarrow{n} \wedge \overrightarrow{f}(n_i)$	vectoriel
F^N	$\overrightarrow{f}(n_i) \wedge \overrightarrow{n}$	vectoriel
N^(N^F)	$\overrightarrow{n} \wedge (\overrightarrow{n} \wedge \overrightarrow{f})(n_i)$	vectoriel
N(N.F)	$\overrightarrow{n}(\overrightarrow{n}\,.\,\overrightarrow{f})(n_i)$	vectoriel

(1) valeur par défaut des termes 'Valeurs nodales'.

(2) équivaut au mot-clé DERIVEE [NORMALE], ce dernier n'étant conservé que pour raison de compatibilité. La valeur $\partial f/\partial n(n_i)$ est calculée comme $\overrightarrow{n}\,.\,\overrightarrow{\mathrm{grad}}\,f(n_i)$, les composantes de $\overrightarrow{\mathrm{grad}}\,f$ étant retournées par la procédure 'utilisateur' DFCTRM.

(3) équivaut au mot-clé FOURNI, ce dernier n'étant conservé que pour raison de compatibilité.

(4) fournit une alternative au calcul de $\partial f/\partial n(n_i)$; la procédure 'utilisateur' FCTRM doit retourner dans ce cas le vecteur $\overrightarrow{\mathrm{grad}}\,f(n_i)$. Il devient alors inutile de programmer la procédure DFCTRM.

Annexe 3. Types des termes 'Condition Essentielle'

Nom générique (mot-clé)	Type de condition essentielle	Type de l'inconnue (u)
'u'=G	$\varphi = g^{(1)}$ ou $[K]\vec{u} = \vec{g}^{(1)}$	scalaire ou vectoriel
'u'.N=G	$([K]\vec{u}).\vec{n} = g^{(1)}$	vectoriel
'u'^N=G	$([K]\vec{u}) \wedge \vec{n} = \vec{g}^{(1)}$	vectoriel

'u' désigne le nom d'une inconnue préalablement déclarée (par exemple à l'aide de la directive INCONNUE).
K désigne le nom d'un tenseur préalablement calculé (par exemple de type VALEURS NODALES à valeurs tensorielles).

(1) les valeurs de la fonction scalaire g ou des composantes de la fonction vectorielle $\vec{g}$ en tous les nœuds i d'un domaine sont calculées à l'aide de la procédure FCTRM (voir la définition d'une donnée associée au terme).

Annexe 4. Types des termes 'Condition de transmission'

Nom générique (mot-clé)	Type de condition de transmission	Type des inconnues (u_i)
'u1'='u2'	$\varphi_1 = \varphi_2 + g$ ou $[K_1]\vec{u}_1 = [K_2]\vec{u}_2 + \vec{g}$	scalaire ou vectoriel
'u1'.N='u2'.N	$([K_1]\vec{u}_1).\vec{n} = ([K_2]\vec{u}_2).\vec{n} + g$	vectoriel
'u1'^N='u2'^N	$([K_1]\vec{u}_1) \wedge \vec{n} = ([K_2]\vec{u}_2) \wedge \vec{n} + \vec{g}$	vectoriel

'u_i' désigne le nom d'une inconnue préalablement déclarée (par exemple à l'aide de la directive INCONNUE).
K_i désigne le nom d'un tenseur préalablement calculé (par exemple de type VALEURS NODALES à valeurs tensorielles).

(1) les valeurs de la fonction scalaire g ou des composantes de la fonction vectorielle $\vec{g}$ en tous les nœuds i d'un domaine sont calculées à l'aide de la procédure FCTRM (voir la définition d'une donnée associée au terme).

– Mai 2000 –

Annexe 5. Liste des directives et mots-clés

Mot-Clé	Directive	Objet	Page
ECRITURE	*utilitaire*	changement d'unité d'impression principale	. . . C_1. 3
WRITE	*utilitaire*	changement d'unité d'impression principale	. . . C_1. 3
TITRE	*utilitaire*	insertion de titre dans le fichier principal d'impression	. . . C_1. 3
LECTURE	*utilitaire*	changement d'unité de lecture des directives	. . . C_1. 3
AVEC ECHO	LECTURE	écho des données sur le fichier des impressions secondaires	. . . C_1. 3
WRITE	*utilitaire*	changement d'unité de lecture des directives	. . . C_1. 3
WITH ECHO	WRITE	écho des données sur le fichier des impressions secondaires	. . . C_1. 3
STOP	*utilitaire*	met fin à l'exécution du programme	. . . C_1. 4
ARRET	*utilitaire*	met fin à l'exécution du programme	. . . C_1. 4
ALLOCATION	*générale*	initialisation de l'allocation dynamique	. . . C_1. 5
QUANTITE	ALLOCATION	modification du nombre maximum initial de tableaux gérés	. . . C_1. 5
TABLEAUX	ALLOCATION	définition des buffers d'entrée/sortie	. . . C_1. 5
FICHIERS	ALLOCATION	définition des fichiers de la mémoire secondaire	. . . C_1. 5
RUMEURS	ALLOCATION	active ou non le fichier contenant l'état des tableaux	. . . C_1. 5
ACTIVES	ALLOCATION	état du fichier des 'RUMEURS'	. . . C_1. 5
INACTIVES	ALLOCATION	état du fichier des 'RUMEURS'	. . . C_1. 5
MOUCHARD	ALLOCATION	active (BAVARD) ou non (MUET) le fichier contenant les opérations de gestion des tableaux	. . . C_1. 5
BAVARD	ALLOCATION	état du 'MOUCHARD'	. . . C_1. 5
MUET	ALLOCATION	état du 'MOUCHARD'	. . . C_1. 5
STRUCTURE	*générale*	modification des tailles initiales des structures	. . . C_1. 6
DONNEES	STRUCTURE	mot-clé optionnel	. . . C_1. 6
INCONNUE	STRUCTURE	modif. du nombre initial d'inconnues et du nombre maximal initial de caractères par nom d'inconnues	. . . C_1. 6
DOMAINE	STRUCTURE	modification du nombre maximum initial de domaines géométriques et du nombre maximal initial de caractères par nom de domaine	. . . C_1. 6
DONNEE	STRUCTURE	modification du nombre maximum initial de données et du nombre maximal initial de caractères par nom de donnée	. . . C_1. 6
TERME	STRUCTURE	modification du nombre maximum initial de termes et du nombre maximal initial de caractères par nom de terme	. . . C_1. 6
CONSTANTE	STRUCTURE	modification du nombre maximum initial de constantes	. . . C_1. 6

NOMBRE	STRUCTURE	nombre initial d'articles d'une structure	. . . C_1. 6
LONGUEUR	STRUCTURE	nombre initial de caractères d'un tableau de noms	. . . C_1. 6
GEOMETRIE	*générale*	déclenche la lecture du fichier de maillage	. . . C_1. 8
MAILLAGE	*générale*	déclenche la lecture du fichier de maillage	. . . C_1. 8
LECTURE	MAILLAGE	permet la lecture sur un fichier dont le nom est donné par une chaîne de caractères	. . . C_1. 8
UNITE	MAILLAGE	mot-clé optionnel = FICHIER	. . . C_1. 8
FICHIER	MAILLAGE	mot-clé optionnel = UNITE	. . . C_1. 8
FORMAT	MAILLAGE	définition du format de lecture des coordonnées ou des numéros globaux des points du maillage	. . . C_1. 8
LECTURE	MAILLAGE	mot-clé optionnel	. . . C_1. 8
COORDONNEES	MAILLAGE	déclenche la lecture du format de lecture des coordonnées des points des éléments du maillage	. . . C_1. 8
NUMEROTATION	MAILLAGE	déclenche la lecture du format de lecture des numéros globaux des points des éléments du maillage	. . . C_1. 8
COMMENTAIRE	MAILLAGE	AVEC ou SANS COMMENTAIRE indique que des lignes 'commentaire' sépare les données des éléments du fichier de maillage	. . . C_1. 8
IMPRESSION	MAILLAGE	permet l'impression par élément des coordonnées et numéros globaux des points du maillage	. . . C_1. 8
DESCRIPTION	MAILLAGE	directive obligatoire déclenchant la lecture des données globales du maillage (doit être suivi du mot GLOBALE)	. . . C_1. 9
VARIABLES	MAILLAGE	déclenche la lecture des noms des variables d'espace (doit être suivi du mot ESPACE et des chaînes définissant les noms des variables d'espace)	. . . C_1. 9
NOMBRE	MAILLAGE	déclenche la lecture du nombre d'éléments du maillage (doit être suivis du mot ELEMENTS et du nombre d'éléments)	. . . C_1. 9
BLOC	MAILLAGE	déclenche la lecture du nombre d'éléments d'un bloc d'éléments du maillage (est nécessairement suivi du mot définissant le type géométrique, du nombre d'éléments du bloc puis du mot-clé ELEMENT)	. . . C_1. 9
ELEMENTS	MAILLAGE	mot-clé obligatoire suivant le mot-clé BLOC	. . . C_1. 9
DOMAINE	MAILLAGE	permet de définir un nouveau domaine géométrique, doit être suivi du nom du domaine et de la liste de ses constituants – liste de numéros d'éléments ou liste de couples (numéro d'élément, numéro de face ou d'arête)	. . . C_1. 10
ELEMENT	MAILLAGE	définit les constituants d'un domaine	. . . C_1. 10
FACE	MAILLAGE	définit les constituants d'un domaine de bord	. . . C_1. 10
ARETE	MAILLAGE	définit les constituants d'un domaine de bord	. . . C_1. 10
POINT	MAILLAGE	définit les constituants d'un domaine ponctuel	. . . C_1. 10

FIN	MAILLAGE	termine la lecture des domaines et du fichier de maillage	. . . C_1. 11
INCONNUE	*générale*	déclenche la lecture du nom et du type des inconnues du problème	. . . C_1. 19
VALEURS	INCONNUE	inconnue 'éléments finis' standard (nécessairement suivi du mot-clé NODALES)	. . . C_1. 19
NODALES	INCONNUE	voir mot-clé VALEURS	. . . C_1. 19
SCALAIRE	INCONNUE	définit le type d'une (fonction) inconnue du problème	. . . C_1. 19
VECTORIEL	INCONNUE	définit le type d'une inconnue du problème	. . . C_1. 19
COMPOSANTES	INCONNUE	nombre de composantes d'une inconnue vectorielle	. . . C_1. 19
INTERPOLATION	INCONNUE	définit le type d'interpolation d'une (fonction) inconnue du problème	. . . C_1. 19
ISOPARAMETRIQUE	INCONNUE	interpolation isoparamétrique	. . . C_1. 19
LAGRANGE	INCONNUE	interpolation de Lagrange	. . . C_1. 19
SPECTRALE	INCONNUE	inconnue pour le méthode des éléments finis localisés	. . . C_1. 19
FONCTIONS	INCONNUE	définit le rang de la troncature de l'opérateur frontière dans la méthode des éléments finis localisés. Doit être précédé du mot NOMBRE et suivi du mot PROPRES	. . . C_1. 19
TERMES	INCONNUE	nombre de focntions dans développement des fonctions spectrales en somme d'exponentielles. Doit être précédé du mot NOMBRE et suivi du mot FONCTION	. . . C_1. 19
SUPPLEMENTAIRE	INCONNUE	définition d'une inconnue non valeurs nodales	. . . C_1. 19
SYMETRIE	*générale*	définition de symétrie du problème par rapport à un plan ou un axe de coordonnées	. . . C_1. 21
ANTISYMETRIE	*générale*	définition d'antisymétrie du problème par rapport à un plan ou un axe de coordonnées	. . . C_1. 21
QUADRATURE	*générale*	modification du degré de précision des formules de quadrature	. . . C_1. 22
GAUSS	QUADRATURE	définit le type d'une formule de quadrature	. . . C_1. 22
DEGRE	QUADRATURE	définit le degré d'une formule de quadrature	. . . C_1. 22
NODALE	QUADRATURE	permet le choix d'une formule de quadrature sur les nœuds d'un élément fini	. . . C_1. 22
CALCUL	*générale*	permet de définir les calculs à effectuer sur un domaine	. . . C_1. 23
DOMAINE	CALCUL	mot-clé optionnel	. . . C_1. 23
IMPRESSION	CALCUL	niveau d'impression pour un domaine de calcul, pour un terme,etc.	. . . C_1. 23
QUADRATURE	CALCUL	modifie la valeur par défaut du schéma de quadrature numérique d'un terme éléments finis (est nécessairement suivi du mot-clé DEGRE)	. . . C_1. 23
TERME	CALCUL	définition d'un nouveau terme éléments finis	. . . C_1. 23

MATRICE	CALCUL	définition d'un terme non assemblé	. . . C_1. 23
ELEMENTAIRE	CALCUL	définition d'un terme non assemblé	. . . C_1. 23
NIVEAU	CALCUL	niveau d'un terme	. . . C_1. 23
INCONNUE	CALCUL	définit le nom des inconnues attachées à un terme	. . . C_1. 23
COLONNE	CALCUL	défini l'inconnue en colonne pour un terme matriciel	. . . C_1. 23
LIGNE	CALCUL	défini l'inconnue en ligne pour un terme matriciel	. . . C_1. 23
INTEGRAND	CALCUL	mot-clé précédent le nom d'un intégrand pour le calcul d'un terme éléments finis	. . . C_1. 23
DONNEE	CALCUL	mot-clé optionnel permettant de définir le nom d'une donnée affectée à un terme	. . . C_1. 23
ASSOCIEE	CALCUL	ce mot-clé optionnel précédé du mot-clé DONNEE permet de définir le nom d'une donnée associée à une donnée de type FONCTION	. . . C_1. 23
VALEURS	CALCUL	nécessairement suivi du mot-clé NODALES, ce mot permet de déclarer le calcul d'un terme de type valeurs nodales	. . . C_1. 26
DONNEE	CALCUL	mot-clé optionnel permettant de définir le nom d'une donnée affectée à un terme	. . . C_1. 26
ASSOCIEE	CALCUL	ce mot-clé optionnel précédé du mot-clé DONNEE permet de définir le nom d'une donnée associée à une donnée de type FONCTION	. . . C_1. 26
CONDITION	CALCUL	ce mot-clé permet de définir un terme de type condition essentielle ; il est nécessairement suivi du mot-clé ESSENTIELLE ou TRANSMISSION	. . . C_1. 28
ESSENTIELLE	CALCUL	déclaration d'une condition essentielle	. . . C_1. 28
DONNEE	CALCUL	mot-clé optionnel permettant de définir le nom d'une donnée affectée à un terme	. . . C_1. 28
ASSOCIEE	CALCUL	ce mot-clé optionnel précédé du mot-clé DONNEE permet de définir le nom d'une donnée associée à une donnée de type FONCTION	. . . C_1. 28
TRANSMISSION	CALCUL	déclaration d'une condition de transmission	. . . C_1. 31
DONNEE	CALCUL	mot-clé optionnel permettant de définir le nom d'une donnée affectée à un terme	. . . C_1. 31
ASSOCIEE	CALCUL	ce mot-clé optionnel précédé du mot-clé DONNEE permet de définir le nom d'une donnée associée à une donnée de type FONCTION	. . . C_1. 31
NORMALES	CALCUL	permet le calcul des normales aux nœuds d'un domaine de bord	. . . C_1. 33
AUTRES CALCULS	*générale*	permet de définir les termes autres	. . . C_1. 34
TERME	AUTRES CALCULS	définition d'un nouveau terme (doit être suivi du nom du terme)	. . . C_1. 34
IMPRESSION	AUTRES CALCULS	niveau d'impression pour un domaine de calcul, pour un terme,etc.	. . . C_1. 34

ASSEMBLAGE	*générale*	déclenche la lecture du type de l'assemblage et de la liste des termes constituants d'un terme assemblé	. . . C_1. 35
COMPOSE	ASSEMBLAGE	mot-clé indiquant un assemblage simple (valeur par défaut)	. . . C_1. 35
MULTI-INCONNUE	ASSEMBLAGE	assemblage de termes avec inconnues différentes	. . . C_1. 35
DONNEE	*générale*	permet de définir le nom et la valeur d'une donnée de type constante	. . . C_1. 37
LONGUEUR	DONNEE	fixe la longueur d'une données de type tableau	. . . C_1. 37

– Mai 2000 –

Annexe 6. Index des procédures

GBCT	gradient bi-conjugué [préconditionné]	C_2. 26
GCS	gradient conjugué 'squared' [préconditionné]	C_2. 27
FASV	résolution d'un système linéaire à matrice factorisée	C_2. 27
PGAUSS	condensation statique dans un système linéair	C_2. 28
VPINIT	initialisation des vacteurs propres	C_2. 28
VPMISE	calcul de valeurs/vecteurs propres, méthode des sous-espaces	C_2. 28
DFTERM	comparaison de deux termes vectoriels uni-inconnue	C_2. 29
BYE	fin d'un programme	C_2. 30
TBCREE	création d'un tableau	C_3. 1
TBTUER	destruction d'un tableau	C_3. 1
TBSAVE	mise en état de sauvegarde différée d'un tableau	C_3. 2
TBRR1	recherche d'un tableau	C_3. 2
TBRR*l*	recherche de plusieurs tableaux	C_3. 2
TBAR1	demande de l'adresse d'un tableau en mémoire centrale	C_3. 2
TBAR*l*	demande des adresses de plusieurs tableaux en mémoirecentrale	C_3. 2
TBTYPE	retourne le type de déclaration d'un tableau	C_3. 3
TBARTI	recherche du nombre d'articles d'un tableau	C_3. 3
TBAJST	ajustement de la taille d'un tableau	C_3. 3
STIMPR	impression de l'état des super-tableaux	C_3. 3
PRRUME	impression du tableau des rumeurs	C_3. 3
FCTRM	procédure utilisateur pour définir les données de type fonction	C_4. 2
DFCTRM	procédure utilisateur pour définir les données de type gradient de fonction (obsolète)	C_4. 5