Guide d'utilisation et normes de programmation

Paul-Louis George, Marina Vidrascu

IRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.(3) 954 90 20

Rapports Techniques

N° 42

GUIDE D'UTILISATION ET NORMES DE PROGRAMMATION

Paul Louis GEORGE
Marina VIDRASCU

Octobre 1984

GUIDE D'UTILISATION ET NORMES DE PROGRAMMATION

Paul Louis GEORGE
Marina VIDRASCU

Octobre 1984

Guide d'utilisation et normes de programmation du Club Modulef

Résumé

La bibliothèque Modulef est constituée d'un ensemble de programmes écrits en Fortran 77 permettant la résolution de nombreux problèmes par la méthode des Eléments Finis.

Nous développons les idées de base retenues pour construire ce code:

- gestion dynamique de la mémoire,
- notion de Structure de Données,
- idée de module, de sous-programme, d'algorithme,
- etc ...

Ces concepts ont été élaborés pour assurer la fiabilité, la portabilité et la possibilité d'évolution du code.

De plus, nous présentons un guide général d'utilisation des programmes et les spécifications conseillées pour leur écriture.

Abstract

Modulef is a general purpose finite element computer program written in Fortran 77.

We point out the basic ideas considered for building this library :

- dynamic memory allocation,
- notion of Data-base,
- idea of modules, subroutines and algorithms,
- etc...

These concepts have been developped to guarantee reliability, portability and possibility of modification and evolutions.

Furthermore we present a general guide to use the programs and specifications for writing them.

SOMMAIRE

INDEX ALPHABETIQUE DES PROGRAMMES DECRITS

INDEX ALPHABETIQUE DES PROGRAMMES

0. GENERALITES

L'utilisation de la bibliothèque MODULEF peut revêtir plusieurs formes selon le problème à traiter.

NIVEAU 1

Le problème peut être entièrement traité en utilisant les modules existants.

L'utilisateur doit alors appeler le module (ou l'enchainement de modules) désiré. Dans ce cas une connaissance superficielle du code et des idées générales de base est suffisante.

NIVEAU 2

Les modules existants de la bibliothèque permettent de traiter la plupart des étapes du calcul. Néanmoins certains calculs (et les programmes correspondants) ne sont pas prévus dans le code.

L'utilisateur doit alors enchaîner des modules existants puis développer ses PROPRES modules (en utilisant au maximum les modules, algorithmes et utilitaires existants) et en respectant les normes du code MODULEF. Cette solution, préconisée par le Club, nécessite une connaissance plus poussée de la structuration du code. Elle a, par contre, l'avantage de ne poser aucun problème d'interface. De plus, les nouveaux modules sont intégrables instantanément aux bibliothèques MODULEF.

NIVEAU 3

Il s'agit de l'utilisation simultanée de modules de MODULEF et d'autres programmes existants par ailleurs. Dans ce cas deux possibilités se présentent :

a) "chapeauter" les programmes existants qui contiennent des algorithmes et créer un MODULE. Dans ce cas le niveau 2 est retrouvé.

b) l'utilisateur réalise l'interface entre les programmes de MODULEF et ses propres programmes. Cette solution est généralement à bannir.
Les nouveaux programmes ainsi obtenus sont difficiles à intégrer à la bibliothèque, il y a souvent duplication de code.
Elle est néanmoins envisageable pour des problèmes très compliqués et particuliers, difficiles à généraliser.

Quelle que soit la solution choisie une connaissance assez approfondie du code s'avère nécessaire.

Cette brochure se propose donc de traiter ces 3 cas d'utilisation de la bibliothèque. On détaillera en particulier les notions utiles pour un travail au niveau 2.

1. IDEES ET DEFINITIONS DE BASE

1.1. Terminologie

Quel que soit le niveau d'utilisation du code Modulef, il est bon de connaître les quelques notions suivantes :

Le **code Modulef** se compose d'un ensemble de sous programme (en Fortran 77, à de rares exceptions près) qui réalisent différents algorithmes de calcul dans le cadre de la méthode des **éléments finis.**

Ces sous-programmes, selon le type de problème qu'ils traitent, sont regroupés dans des **bibliothèques** éditables.

Les particularités essentielles du code MODULEF sont :

a) la gestion dynamique
b) la modularité
c) la transportabilité.

a) La mémoire est gérée de manière **dynamique** dans un seul super-tableau par un ensemble de sous-programmes qui permettent de créer des tableaux mais aussi de réutiliser la place mémoire précédemment allouée à un tableau qui ne sera pas utilisé par la suite.

b) La bibliothèque ne contient pas des programmes capables de résoudre un problème d'un bout à l'autre mais des **modules.**

Un **MODULE** est un ensemble de sous-programmes réalisant l'algorithme correspondant à une étape importante du calcul et permettant la gestion des données nécessaires et des résultats obtenus.

Les modules sont conçus selon le principe de la **séparation** de **l'algorithme** numérique et de la **gestion** des paramètres d'entrée et de sortie.

En contrepartie de la modularité, il convient de définir les interfaces entre les modules : les **structures de données.**

Une **STRUCTURE DE DONNEE** ou S.D. est un ensemble de tableaux structurés par avance contenant les valeurs descriptives d'une étape importante du calcul. Le nombre des structures de données est assez réduit.

A chaque structure de donnée est associé un ensemble **d'outils de gestion** permettant de la manipuler.

Une structure de donnée servant à définir les paramètres d'entrée nécessaires à un module est une **structure de donnée d'entrée** (S.D.E.). A l'inverse si elle sert à recueillir les résultats d'un module c'est une **structure de donnée de sortie** (S.D.S.).

Les S.D. sont repérées par leur type et éventuellement le fichier les contenant (ex : un maillage est stocké dans une S.D. de type NOPO, pouvant résider sur un fichier repéré par son nom et/ou son numéro).

REMARQUE :

Chaque sous-programme du code est conçu en fonction de normes précises pour permettre :

- la réalisation d'interfaces
- une garantie de portabilité
- une garantie de fiabilité
- une maintenance aisée
- une communication et des modifications aisées.

1.2. Le langage MODULEF

Ce langage est FORTRAN 77. De plus quelques groupes de sous-programmes spécifiques concernent :

- la gestion dynamique des tableaux (traités au chapitre 4)
- les fichiers en accès direct (traités au chapitre 6)
- la gestion des caractères.

TRAITEMENT DES CARACTERES

L'utilisation des caractères respecte la convention suivante :

- **Un mot** contient **4 caractères cadrés à gauche.**

Les sous-programmes permettant de manipuler les caractères sont NUMALP, INCANO, INCAPA, CHAR4, ICHAR4.

i) FUNCTION NUMALP(N)

Cette fonction de type CHARACTER*1, fournit, à partir de l'entier N un caractère selon la règle :

N	-1	0	1	9	10	35
NUMALP(N)	'%'	'0'	'1'	'9'	'A'	'Z'

Si N < +1 ou N > 35 NUMALP(N) = '*'

ii) FUNCTION INCANO (NOM, N, IPOS)

Cette fonction, de type INTEGER permet de remplacer le IPOS-ième caractère de NOM par le caractère obtenu par NUMALP(N)

iii) FUNCTION INCAPA (NOM, NCAR, IPOS)

Cette fonction, de type INTEGER permet de remplcaer le IPOS-ème caractère de NOM par le caractère NCAR.

REMARQUE : INCANO(NOM, N, IPOS) = INCAPA (NOM, NUMALP(N),IPOS)

iv) FUNCTION CHAR4(I)

Effectue la conversion de l'entier I en CHARACTER*4

v) FUNCTION ICHAR4(CHAR)

Effectue la conversion de CHAR (CHARACTER*4) en entier.
Ces deux dernières fonctions sont également décrites dans la brochure 109, P. LAUG "Conversion de Modulef en Fortran 77", Janvier 1984.

1.3. Fil directeur de la résolution d'un problème MODULEF.

1.3.1. Quelques idées sur la conception descendante et la programmation ascendante

La résolution d'un problème par MODULEF se fait selon les principes de la conception descendante. Cette résolution comporte plusieurs étapes :

1) écrire **le système d'équations aux dérivées partielles** à résoudre
2) choisir une **formulation variationnelle**
3) écrire **l'algorithme** mathématique permettant de résoudre le problème
4) décomposer l'algorithme en un enchainement d'**opérateurs mathéma -tiques** conceptuellement simples.
5) associer à chaque opérateur mathématique **le ou les modules** informatiques qui le réalisent.

Une fois ces étapes franchies la résolution d'un problème **est un enchaînement de modules.**

REMARQUES :

a) Les étapes 1 à 4 sont indépendantes de MODULEF et de toute structure informatique ; elles sont communes à la résolution de tout problème du plus simple au plus compliqué.

b) L'enchaînement capable de résoudre un problème donné n'est pas unique.

c) A un même opérateur mathématique peuvent correspondre plusieurs modules qui correspondent à des structurations de données différentes. Par exemple à l'opérateur mathématique : **résolution d'un système linéaire symétrique par une méthode directe de Cholesky** correspondent deux opérateurs informatiques selon que la matrice réside en mémoire centrale ou en mémoire secondaire.

d) Suivant que le module correspondant à un opérateur mathématique donné existe ou pas on retombe sur le niveau d'utilisation 1 ou 2 définis précédemment.

AVANTAGES ET INCONVENIENTS DE LA CONCEPTION DESCENDANTE ET DE LA PROGRAMMATION ASCENDANTE

Avantages

- un problème compliqué est décomposé en une suite de sous-problèmes plus simples
- séparation des difficultés mathématiques et informatiques
- utilisations multiples d'un opérateur dans des contextes différents ce qui évite la duplication de code.
- fiabilité de la programmation
- facilité de comparaison des méthodes ou éléments différents : seul le module correspondant à un opérateur mathématique est à changer dans un enchaînement donné.
- un enchaînement souvent utilisé peut être figé en une boîte noire.

Inconvénients

Naturellement ces inconvénients sont largement compensés par les avantages indiqués plus haut et ils ne sont ressentis que pendant la phase d'expérimentation de Modulef.

- La résolution d'un problème simple nécessite l'assimilation des idées générales et l'appel de plusieurs modules
- L'inclusion dans la bibliothèque d'enchainements tout prêts, capables de résoudre des problèmes simples, réduit la souplesse d'utilisation et les différents choix offerts par l'ensemble de la bibliothèque. De tels enchainements sont, soit pauvres du point de vue des possibilités offertes, soit gigantesques du point de vue des ressources informatiques nécessaires.

1.3.2. **Résolution d'un problème**

En utilisant les notions de la conception descendante la résolution d'un problème par MODULEF se résume, après réalisation des étapes 1 à 4 de 1.2.1 à rechercher dans les bibliothèques les MODULES qui réalisent les opérateurs mathématiques mentionnés dans l'étape 4. Une liste complète des modules actuels est contenue dans la documentation 85 "Présentation du Club MODULEF". Cette documentation est mise à jour tous les ans au moment de l'Assemblée Générale du Club Modulef" (un fascicule des seules nouveautés de la dernière version étant disponible).

De façon générale pour retrouver les équations aux dérivées partielles et les différentes formulations variationnelles utilisées se reporter aux introductions des documentations concernées (ex 36, 59, 63, 76, 79, 88, 93, 97, 107). Pour les problèmes classiques de thermique ou élasticité linéaire ces mêmes renseignements sont contenus dans les introductions des documentations 100 et 101 (fiches techniques des éléments)

Un exemple complet de résolution d'un problème simple est fourni en annexe.

2. Utilisation par appel des modules des bibliothèques

Les modules existant permettent de résoudre entièrement le problème que l'on se pose. Il y a alors deux moyens de procéder :

2.1. Appel standard d'un (ou plusieurs) module(s)

En s'aidant de la documentation spécifique du module : l'utilisateur doit alors

- écrire un programme principal, en Fortran, qui :

i) déclare le super-tableau de travail nommé M ;

ii) initialise les variables nécessaires à la gestion dynamique de ce tableau par un appel au S.P. INITI (cf page 15) ;

iii) afin de réaliser l'alignement des réels double précision, déclare une variable double précision qu'il met, si au moins l'un des tableaux gérés dans M est de ce type, en équivalence avec M(1) ;

iv) donne les valeurs et paramètres d'entrée nécessaires ;

v) affecte les fichiers contenant les S.D.E. utiles, repérées par leur numéro et un paramètre de niveau ;

vi) idem pour les S.D.S.

les conventions sur les numéros de fichiers NF sont :

NF = 0 il n'y a pas de fichier ; le calcul s'effectue en mémoire centrale

NF > 0 fichier séquentiel de numéro logique NF

NF < 0 fichier en accès direct de numéro logique NF

si ces fichiers sont en accès direct, l'utilisateur doit l'indiquer (voir INIDIR, DEFDIR ..., brochure 42)

vii) réalise l'appel du module

- écrire les sous-programmes supplémentaires éventuellement utiles (FUNCTION, SUBROUTINE ...)
- compiler
- assigner les fichiers des S.D. et les bibliothèques
- linker (en segmentant éventuellement)
- "perforer" les cartes données si elles sont utiles (en général ces cartes sont inexistantes ou en nombre très réduit)
- l'exécution peut alors être lancée.

2.2. Appel conversationnel d'un (ou plusieurs) module(s)

Certaines étapes du calcul peuvent être activées de manière conversationnelle.

Cette méthode permet de :

- ne pas avoir de programme Fortran à écrire ;
- ne pas avoir à assigner les fichiers, opération maintenant possible grâce à Fortran 77 ;
- se laisser guider par un système de /questions/ réponses/ documentation automatique/ pour générer les données nécessaires au module.

Une brochure spécifique décrit ce mode d'activation des modules (cf brochure 108).

2.3. Remarque : boîte noire

Ouand un enchainement est destiné à être exécuté un certain nombre de fois, on peut en faire le link une seule fois et obtenir ainsi une **boîte noire** exécutable.

3. Utilisation nécessitant la création de nouveaux MODULES

Quand au moins une étape du calcul à effectuer n'existe pas dans le code Modulef, il faut programmer soi-même un ou plusieurs sous-programmes.

Il est souhaitable de faire ces développements dans le cadre des normes du Club Modulef.

Les chapitres suivants se proposent de présenter toutes les notions nécessaires à la compréhension des normes de programmation.

Les points traités, étant d'intérêt inégal selon le type de travail à effectuer, chaque chapitre est divisé en deux parties. Une première partie, permettant une lecture rapide (cf § Généralités dans la suite), présente les grandes lignes du point abordé, tandis qu'une deuxième partie donne tous les détails nécessaires.

Seront abordés successivement :

- la gestion dynamique de la mémoire dans le super tableau
- la notion de Structure de Donnée
- les fichiers en accès direct
- la distinction module/sous programme
- la spécification pour l'écriture d'un module (en particulier le le traitement des S.D.E. et des S.D.S.)
- les spécifications pour l'écriture des sous programmes.

4. La gestion dynamique

4.1. Généralités

La gestion dynamique de la mémoire centrale, dont l'utilité n'est plus à démontrer, permet de gérer tous les tableaux, dans un seul tableau appelé super tableau.

Cette gestion est basée sur l'emploi :

- du super tableau M
- d'une zone de travail (le common TRAVAI)
- de 8 sous-programmes utilitaires.

Avec ces objets, il est possible de créer, trouver, lire, supprimer, changer le nom des tableaux de la mémoire centrale et ainsi de la gérer au mieux.

Le super tableau est amené à contenir des tableaux de types différents (entier, réel, double précision ...)

Les différents tableaux sont accessibles par leurs adresses dans le super vecteur.

Pour ces deux raisons, il est nécessaire de tenir compte de règles de cohérence entre types (un tableau double précision doit commencer par un mot d'adresse paire).

Tous ces points sont détaillés dans les paragraphes suivants.

4.2. Outils de la gestion dynamique

4.2.1. Le super tableau M

Tous les tableaux utilisés par un module doivent être adressés dans le super tableau de travail M (à l'exception d'éventuels tableaux de taille réduite à quelques mots).

Comme M contient des tableaux de types différents il convient de respecter les règles suivantes :

- un tableau double précision (ou complexe) commence par un mot d'adresse mémoire paire.
- un tableau complexe double precision commence par un mot d'adresse multiple de 4.

Ces règles d'alignement sont respectées, à l'intérieur du vecteur M, par les utilitaires d'adressage. Il est néanmoins nécessaire que l'adresse du super tableau M soit elle même cohérente. Ceci est réalisé par les instructions

```
DOUBLE PRECISION DM
EQUIVALENCE (DM, M(1))
```

du programme d'appel du module (cf iii) 2.1.).

Un tableau est défini par son :

- type (le type des valeurs qu'il contient)
- nom (identificateur de 4 caractères)
- adresse dans M
- longueur (en nombre de mots et non pas de variables !)

Ces caractéristiques (dont la signification est détaillée plus bas) sont contenues dans les **TABLES** : zone de travail réservée qui permet de gérer les tableaux.

4.2.2. La zone de travail (COMMON /TRAVAI/)

Cette zone de 1024 mots, le common TRAVAI, est réservée à la gestion des tableaux.

Elle contient les informations suivantes :

```
COMMON/TRAVAI/IM, LM, NOMTAB, NNN, NTYTAB(204), NTAB(204),
              IATAB(204), LTAB(204), NPERMU(204)
```

avec

IM : première adresse libre dans M après le dernier tableau adressé

LM : la dimension du vecteur de travail M

NOMTAB : le nombre de tableaux actuellement gérés dans M (il est possible de gérer jusqu'à 204 tableaux)

NNN : le paramètre d'impression des tables :

0 pas d'impression

1 impression de chaque opération affectant les tables.

Puis les tables, c'est-à-dire, la description de tous les tableaux

NTYTAB(I) : le type du tableau I

1 : entier, 2 : réel simple précision, 3 : logique, 4 : alpha-numérique, 5 : reél double précision, 6 : complexe, 7 : complexe double précision.

NTAB(I) : l'identificateur du tableau I, un nom de 4 caractères. Il est nécessaire d'éviter que des tableaux différents aient le même identificateur dans les tables.

Convention sur les noms à adopter :

INTERDIT : les 2 premiers caractères sont ceux du nom d'une S.D.

CONSEILLE : les tableaux temporaires utilisés par un module ont le nom qui commence par ':'

IATAB(I) : l'adresse dans M de la première valeur du tableau I
LTAB(I) : le nombre de mots du tableau I
NPERMU(I) : le rang dans M du tableau I

ces informations, nécessaires à la gestion automatique, ne sont jamais manipulées par le programmeur !

Ce common décrit ainsi l'ensemble des tableaux actifs dans le super vecteur M. Il permet en particulier de calculer le nombre de mots des zones restant disponibles pour y insérer de nouveaux tableaux. Il permet donc de gérer dynamiquemment la mémoire disponible en supprimant les tableaux devenus inutiles et en insérant, si possible, à leur place les nouveaux tableaux nécessaires.

L'identificateur (4 caractères) d'un tableau est choisi avec soin pour éviter les ambiguïtés, les conflits et pour faciliter la lisibilité des programmes.

4.2.3. Les sous programmes utilitaires

4.2.3.1. Descriptions des utilitaires

ADRESS, TROUVE, TUER, MAXTAM, READRE, RENOMM, IMATAB

i) notations : pour la suite

M est le vecteur de travail
LM est le nombre de mots de M
NTYP est le type du tableau considéré :
1 entier, 2 réel, 3 logique, 4 hollerith, 5 réel double précision
NOMT est son identificateur
IA son adresse mémoire dans M
L son nombre de mots

ii) ADRESS :

SUBROUTINE ADRESS (NTYP, NOMT, IA, L, M)

Cet utilitaire permet l'adressage dans M du tableau de nom NOMT.
Les paramètres d'entrée sont :
NTYP : le type du tableau
NOMT : son nom de 4 caractères
ex : NOMT = ICHAR4 ('TOTO')

IA : 0 le tableau est adressé dans une zone libre de M
de longueur supérieure ou égale à L.
0 < IA < LM l'adresse du tableau est forcée à IA
IA < 0 ou IA+L < LM l'adressage est impossible, un diagnostic
le signale et la liste des tables est obtenue

L : le nombre de mots du tableau

M : le super tableau

En sortie :

IA : adresse dans M du tableau

iii) TROUVE

SUBROUTINE TROUVE (NTYP, NOMT, IA, L, NUMC, M)

Cet utilitaire permet de récupérer l'adresse , la longueur et le type du tableau NOMT.

Paramètre d'entrée :
NOMT

En sortie on obtient :
IA, L, NTYP

Si IA = 0 le tableau n'a pas été trouvé dans les tables

L'utilisation de cet utilitaire est possible dans les sous programmes ou fonctions utilisateur ; elle est déconseillée dans les sous-programmes de la bibliothèque : on préfèrera passer les adresses des tableaux voulus plutôt que de les "trouver".

iv) TUER

SUBROUTINE TUER (NOMT, M)

Cet utilitaire permet de supprimer des tables le tableau de nom NOMT et de récupérer la place devenue disponible.

v) MAXTAM

SUBROUTINE MAXTAM(M, IA, LMAX)

Cet utilitaire permet, à tout moment, de chercher dans les tables la longueur maximum disponible.

En sortie :

IA et LMAX : l'adresse de la zone de longueur LMAX disponible

vi) READRE

SUBROUTINE READRE (NTYP, NOMT, IA, L, M, NRET)

Cet utilitaire permet de réadresser le tableau NOMT de type NTYP et de nombre de mots L.

Si le tableau existe déjà il y a comparaison entre :
- son adresse retrouvée IAR
- son adresse demandée IA

Si IA ≠ IAR l'ancien tableau est supprimé

Si IA = IAR on compare la longueur L demandée et la longueur LR retrouvée

Si L ≠ LR l'ancien tableau est supprimé et le nouveau est adressé

Si L = LR retour sans adressage, puisque celui-ci est correct

L'argument de sortie NRET vaut :

0 si le tableau a bien été adressé

1 s'il ne l'a pas été c'est-à-dire qu'il existait déjà avec les bonnes caractéristiques.

viii) RENOMM

SUBROUTINE RENOMM (NOMOLD, NOMNEW, IA, L, M)

Cet utilitaire permet de changer le nom d'un tableau.

viii) IMATAB

SUBROUTINE IMATAB(M)

Cet utilitaire permet de lister le contenu des tables.

4.2.3.2. Remarques sur l'utilisation des sous-programmes de la gestion dynamique

i) l'adresse IA en entrée, ne doit pas être une constante :

ex : IA = 0
CALL ADRESS (NTYP, NOMT, IA, L, M)
et non CALL ADRESS (NTYP, NOMT, 0, L, M)

Cette dernière instruction peut écraser la constante 0.

ii) L'unicité de l'identificateur est à la charge du programmeur. Il n'y a aucun contrôle dans ADRESS.

iii) pour NNN = 1, chaque utilitaire utilisé imprime les caractéristiques du tableau manipulé.

iv) avant d'utiliser ces utilitaires, il convient d'initialiser les tables par appel à INITI.

CALL INITI (M, LM, IMPRE, NNN)

avec

M : le super vecteur
LM : sa longueur
IMPRE : paramètre d'impression des données et résultats
NNN : paramètre d'impression des tables
0 : pas d'impression
1 : impression

v) l'adressage d'un tableau consiste à lui réserver dans M une zone. Affecter des valeurs au tableau est à la charge du programmeur.

vi) en raison des différents types des tableaux contenus dans M, on donne à M plusieurs identifications.

M, XM, DM, ... (entier, réel, double précision ...)

A l'appel CALL MODULE (M, M, M, ...) on fait correspondre SUBROUTINE MODULE(M, XM, DM, ...) avec les spécifications :

dimension M(*), XM(*)

double précision DM(*)

v) Pour affecter des valeurs au tableau TAB résidant dans M à l'adresse IA deux possibilités sont offertes :

- utiliser TAB directement dans M (resp XM, DM) par l'intermédiaire de son adresse
- utiliser TAB dans un sous-programme.

1) **Utilisation directement dans M, XM, DM**

Le tableau TAB n'existe pas physiquement en FORTRAN. Suivant le type du tableau (avec les conventions standard) nous avons la correspondance suivante :

variable logique	variable physique FORTRAN
J = TAB(I)	J = M(IA+I-1)
X = TAB(I)	X = XM(IA+I-1)
D = TAB(I)	D = DM($\frac{IA-1}{2}$ + I)

2) **Utilisation de TAB dans un sous-programme**

L'adresse du premier mot de TAB est passée au sous-programme. Dans le sous-programme l'utilisation de TAB est FORTRAN standard.

Exemple :

A l'appel CALL SP(..., M(IA), ...)
correspond

```
SUBROUTINE SP( ..., TAB, ...)
DIMENSION TAB(*)
```

et dans la subroutine SP la I-ème valeur de TAB est TAB(I).

ATTENTION

Dans ce type d'utilisation l'appel se fait en utilisant M(IA) quel que soit le type du tableau TAB. Il est aussi possible d'appeler par l'intermédiaire de :

CALL SP (..., XM(IA), ...)

SP (..., DM($\frac{IA+1}{2}$), ...) ou SP(...,DM($\frac{IA}{2}$) ...)

car $\frac{IA+1}{2} = \frac{IA}{2}$ puisque IA est pair

Il est conseillé d'utiliser au maximum les tableaux par l'intermédiaire des sous-programmes car la programmation des sous-programmes est alors standard (Fortran 77) et n'obéit à aucune norme supplémentaire.

4.3.3.3. **Un exemple commenté** (voir en annexe).

5. LES STRUCTURES DE DONNEES

5.1. Généralités

Une STRUCTURE DE DONNEES (**S.D.**) est un ensemble de tableaux descriptifs d'une étape importante de calculs. Ces tableaux sont regroupés et traités en bloc par des sous-programmes utilitaires qui constituent des moyens de gestion permettant de les manipuler simplement.

Une Structure de Données possède un **TYPE** (4 caractères). Les différents tableaux de la structure de nom donné '....' sont décrits dans la documentation 2. Les impératifs de la gestion dynamique des tableaux (unicité des noms des tableaux) rendent nécessaire l'utilisation d'un deuxième paramètre le **NIVEAU** pour distinguer les tableaux de deux S.D. de même nature (ex 2 maillages) qui résident simultanément en M.C.

L'intérêt des S.D. est, en particulier :
- d'éviter de fournir en entrée, au module, des quantités importantes d'informations (par ex : un maillage) ; en effet, il suffit de définir la S.D. qui contient ces valeurs et, grâce aux outils de gestion, on dispose des informations nécessaires. De même, en sortie d'un module, l'ensemble des tableaux des S.D. sont stockés simultanément
- de normaliser les résultats (ou les données) d'un module permettant ainsi des interfaces simples et fiables entre modules
- rendre la programmation des MODULES claire : chaque S.D. dispose d'outils de lecture, d'écriture, d'impression.

UTILISATION DES S.D.

En règle générale, un module travaille à partir de valeurs à fournir (option, paramètres, etc...) et des valeurs contenues dans une ou plusieurs S.D. dites S.D.E. Les résultats du module sont fournis sous forme de valeurs et d'une ou plusieurs S.D. dites S.D.S. (E : entrée ; S : sortie).

Lors de l'utilisation en tant que S.D.E. ou S.D.S. d'un module, une structure de donnée est repérée par le **numéro** (0 = mémoire centrale, ce fichier peut être d'accès séquentiel ou direct selon le cas) ou le nom du fichier éventuel qui la contient et le paramètre de **Niveau.**

5.2. Description d'une S.D.

Une S.D. est un ensemble de tableaux structurés par avance. Des outils de gestion sont associés.

Toute structure de données est constituée de tableaux. Le tableau p de la S.D. est entièrement accessible et manipulable par la connaissance des tableaux le précédant.

Compte-tenu des valeurs qu'elles contiennent, les S.D. ont été préalablement définies (cf brochure 2).

Ainsi : tout maillage sera stocké dans une S.D. de type NOPO, une matrice de stockage profil le sera dans une S.D. de type MUA.
etc...

Il existe deux types de S.D. :

Type 1 : tous les tableaux de la S.D. résident simultanément en mémoire centrale. Les premiers tableaux décrivent les suivants et permettent leur manipulation.

Type 2 : la S.D. est paginée (mémoire secondaire d'accès séquentiel ou direct) et seule une ou quelques pages résident simultanément en mémoire centrale. Ces premiers tableaux décrivent le mécanisme de stockage des autres tableaux et permettent leur manipulation.

5.3. Descriptif des tableaux d'une S.D.

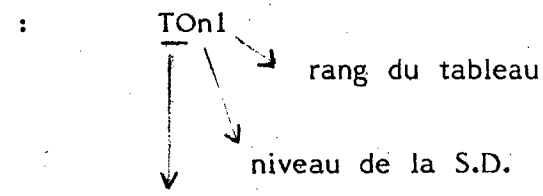
Chaque S.D. est constituée de plusieurs tableaux. Le nom de chaque tableau est fonction du nom de la S.D., de son niveau et du rang de ce tableau dans la S.D..

Dans tout ce qui suit le type de la S.D. est TOTO (type fictif).

Ainsi, pour une S.D. TOTO de niveau n qui contient 6 tableaux, les noms des tableaux sont les suivants :

TOn0, TOn1, TOn2, TOn3, TOn4, TOn5

c'est-à-dire :



REMARQUE :

Rappelons que la notion NIVEAU est liée à la gestion dynamique et n'est pas intrinsèque à la définition d'une S.D. C'est pourquoi dans la brochure 2 (description des structures de données) les noms des tableaux de la même S.D. TOTO seront :

TOT0, TOT1, TOT2, TOT3, TOT4, TOT5

mais lors de toute utilisation le 3ème caractère sera remplacé par le numéro de niveau.

On distingue les **"tableaux"** d'une structure de donnée, dont l'utilisation est prévue dès sa construction, des **"tableaux associés"**. Ces derniers sont des tableaux rajoutés à ceux de la S.D. par l'utilisateur selon ses propres besoins. Ils sont décrits par le deuxième tableau de la S.D. et seront gérés en même temps que ceux de la S.D. par les différents utilitaires.

i) le tableau TOn0 : il contient 32 entiers

Il comporte les informations générales relatives au problème traité et à la S.D.

NTITRE(20)	20 mots de 4 caractères	: titre du travail
NDATE(2)	2 mots de 4 caractères	: date
NOMCRE(6)	6 mots de 4 caractères	: nom du créateur
NOM DE LA S.D.	4 caractères	: ex 'TOTO'
NIVEAU DE LA S.D.	entier NITOTO	
un paramètre réservé	entier	
nombre de tableaux associés à la S.D.	entier NTOTO	

ii) le tableau TOn1 de 22 mots

Ce tableau contient le descriptif des tableaux associés à la S.D.

L'utilisation de ces tableaux n'est pas très répandue. (le paramètre final du tableau TOn0 permet de savoir s'ils existent ou pas).

En cas d'existence il contient les informations suivantes :

Boucle de 1 à NTTOTO

nom du tableau I	1 mot de 4 caractères NTAB
adresse dans M de ce tableau	IATAB
longueur (en mots) du tableau	LTAB
son type	NTYP
un commentaire sur son contenu	18 mots de 4 caractères

REMARQUE

Pour une S.D. stockée sur un fichier la variable IATAB ci-dessus n'a aucune signification. Elle sera automatiquement mise à jour par l'utilitaire de lecture de la S.D.

iii) Le tableau TOn2

Ce tableau contient les informations générales relatives aux valeurs stockées dans la S.D.

ex : nombre d'éléments
nombre de degrés de liberté
taille des tableaux qui suivent
etc...

iv) le tableau TOn3

Ce tableau contient généralement la description de la segmentation du tableau final.

v) le tableau TOn4

Ce tableau contient généralement les informations relatives au pointeur, s'il existe, du tableau suivant.

vi) le tableau final : ici TOn5

C'est ce tableau qui contient les valeurs intéressantes fournies selon le descriptif des tableaux précédents.

REMARQUES :

1) Selon les S.D. cette organisation, bien que fidèle à la description ci-dessus, peut différer légèrement.

2) Il convient de ne pas mélanger dans un même tableau des variables réelles en simple précision et en double précision ce qui peut nuire à l'alignement sur une adresse paire des variables à deux mots.

5.4. Les outils de gestion

La gestion des tableaux des S.D. se fait par l'intermédiaire d'objets spécifiques à chaque S.D. ou communs à toutes.

Ces objets sont :
- un **COMMON ASSOCIE** à la S.D. de nom ALTOTO
- un **tableau de sauvegarde** de ce common de nom TOn%
- des **sous-programmes utilitaires**

a) spécifiques permettant
- l'écriture ECTOTO
- la lecture LETOTO
- la recherche d'adresse RETOTO
- l'impression IMTOTO
- sous-programmes particuliers

b) généraux permettant :
- la manipulation des tableaux associés LETASD, RETASD, ECTASD, COTASD TUTASD
- l'utilisation aisée des S.D. INICSD, SAUCSD, RSTSDE, SAUSDS
- la copie d'une S.D. COPISD
- la destruction de la S.D. TUERSD

5.4.1. Le common associé

Son nom est constitué des 2 lettres AL suivies du nom de la S.D. (ici ALTOTO).

Il contient :

- Une variable entière dont le nom commence par les deux lettres NE et comporte le nom de la structure de données. Il représente le numéro de l'enregistrement logique accessible lorsque le support de la structure de données est un fichier séquentiel en mémoire secondaire ; il représente le numéro de l'enregistrement physique (page) accessible lorsque le support de la structure de données est un fichier d'accès direct en mémoire secondaire.

Cette variable doit avoir une valeur précisée avant chaque opération d'entrée ou de sortie. Son incrémentation est à la charge de l'utilisateur.

- Pour chacun des tableaux, le triplet suivant :

Nom du tableau, son adresse (IA ...) dans M, son nombre de mots (L ...).

Pour notre exemple :

```
COMMON/ALTOTO/NETOTO, NTOT0, IATOT0, LTOT0,
                      NTOT1, IATOT1, LTOT1,
                      NTOT2, IATOT2, LTOT2,
                      NTOT3, IATOT3, LTOT3,
                      NTOT4, IATOT4, LTOT4,
                      NTOT5, IATOT5, LTOT5.
```

REMARQUE

Lors de son utilisation par un module, lorsque les valeurs de ce COMMON doivent être transmises, en paramètre, à un sous-programme ce COMMON est mis en équivalence avec le tableau NCOMMO (de nom NZ suivi du nom de la S.D. ; ici NZTOTO de longueur égale à la longueur du COMMON). Cette équivalence est rendue nécessaire par le fait que sur certaines machines (IBM par exemple) le passage des paramètres par valeur ou par adresse n'est pas identique.

5.4.2. Le tableau de sauvegarde du Common

Son identificateur est constitué de la manière suivante :
- les deux premiers caractères du nom de la structure de données,
- le caractère donné par NUMALP (cf 1.2) pour le numéro de niveau,
- le caractère %,

Si l'on reprend l'exemple utilisé jusqu'ici, le tableau de sauvegarde du COMMON/ALTOTO a pour identificateur TO0% si TOTO est de niveau 0,

Le tableau de sauvegarde recueille les valeurs des variables du common étiqueté associé. L'utilisation de ce tableau est indispensable pour retrouver les tableaux d'une structure de données lorsque l'on utilise plusieurs jeux de données pour un même problème. Par l'intermédiaire de ce tableau, le commun associé est sauvegardé ou restauré. Tout cela se fait au moyen de deux sous-programmes utilitaires : INICSD et SAUCSD (voir ci-dessous).

5.4.3. Les sous programmes utilitaires

Notations

Dans ce qui suit :

M : le vecteur de travail,
NOMSD : variable entière représente l'identificateur de la S.D.,
NIVSD : numéro de niveau de la S.D.,
NCOMMO : tableau équivalent au commun étiqueté, qui en reproduit les valeurs au niveau du programme, (NZTOTO)
LCOMMO : nombre de mots de NCOMMO et du commun étiqueté,
NCSD : variable entière dont la valeur est l'identificateur du tableau de sauvegarde du commun étiqueté, (TO0%)
IACSD : adresse, dans M, du tableau de sauvegarde.
NFTOTO : numéro de support de la S.D. TOTO avec la convention
= 0 mémoire centrale
> 0 fichier séquentiel
< 0 fichier en accès direct.

Z La notion abusive de support NF = 0 est continuellement utilisée dans MODULEF par les traitements M.C. pour simplifier les notations. Il est clair que si NF = 0 il n'y a pas d'E/S !

5.4.3.1. Les sous-programmes utilitaires spécifiques

Les quatre sous-programmes spécifiques permettent les transferts des S.D. entre la mémoire centrale et un fichier (LETOTO, ECTOTO) ; la vérification qu'une S.D. est en mémoire centrale (RETOTO), ou l'impression de la S.D. (IMTOTO).

Les programmes IMTOTO sont des MODULES (à utiliser selon le paragraphe 2). **Les sous-programmes LE, EC, RE (TOTO) utilisent tous le COMMON (ALTOTO) associé à la S.D.**. La mise à jour de ce COMMON est à la charge du programmeur via certains utilitaires. Pour chaque S.D. ces programmes sont décrits dans la brochure 2 (description des S.D.). Les caractéristiques générales de ces sous-programmes sont détaillées ci-dessous :

REMARQUE :

Généralement ces sous-programmes ne sont pas appelés directement mais via RSTSDE, SAUSDS décrits en 5.4.3.2. ii).

i) le sous-programme d'écriture

ECTOTO : son nom commence par EC suivi du nom de la S.D.

SUBROUTINE ECTOTO (M, NFTOTO, NMOSD, NOPT)

Cet utilitaire permet d'écrire, sur le fichier de numéro séquentiel ou en accès direct /NFTOTO/, les tableaux de la S.D. concernée et les éventuels "tableaux associés". Ces derniers sont écrits par appel au sous-programme ECTASD décrit plus loin.

Pour une S.D. de type 1, le sous-programme écrit tous les tableaux de la S.D. suivant l'option NOPT ; en sortie NMOSD donne le nombre de mots transférés.

Pour une S.D. de type 2, seuls les tableaux non paginés sont écrits. L'écriture des pages suivantes est à la charge du programmeur (WRITE).

Chaque enregistrement débute par une variable entière LE qui contient le nombre de mots qui suivent. Cette convention permettra, de lire le contenu d'un enregistrement logique sans connaître, à priori sa longueur.

WRITE (NFTOTO) LE,(M(IA+I-1), I = 1, LE)

La variable NETOTO permet de compter le nombre d'enregistrement du fichier contenant la S.D.

ii) Le sous-programme de lecture

LETOTO : son nom commence par LE suivi du nom de la S.D.

SUBROUTINE LETOTO (M, NFTOTO)

Cet utilitaire permet de lire sur le fichier de numéro NFTOTO des tableaux de la S.D. concernée et les éventuels tableaux associés. Ces derniers seront lus par appel au sous-programme LETASD décrit plus loin.

Pour une S.D. de type 1, tous les tableaux sont adressés et lus en mémoire centrale dans le common.

Pour une S.D. de type 2, seuls les tableaux non paginés sont adressés et lus. Le transfert des pages des tableaux suivants est à la charge du programmeur, néanmoins ces tableaux sont adressés dans le super-tableau (la place ainsi réservée est suffisante pour contenir une page du dernier tableau). Le descriptif des tableaux est alors contenu dans le common.

Le transfert des pages du dernier tableau est à la charge du programmeur. Il est réalisé par :

READ (NFTOTO) LE, (M(IA+I-1), I=1, LE)

avec la convention vue en i)

Dans notre exemple IA est connu, c'est IATOT5.

iii) le sous-programme de recherche

RETOTO : son nom commence par RE suivi du nom de la S.D.

SUBROUTINE RETOTO (M, NFTOTO)

Cet utilitaire permet de rechercher dans M les tableaux de la S.D. TOTO, et les éventuels tableaux associés (par appel à RETASD décrit plus loin). NFTOTO ici vaut zéro.

iv) le sous-programme d'impression

IMTOTO : son nom commence par IM suivi du nom de la S.D.

SUBROUTINE IMTOTO(M, NFTOTO, NITOTO, IMPRE)

Cet utilitaire permet d'imprimer les tableaux de la S.D. TOTO contenus dans le fichier de numéro NFTOTO (≥ 0, ou > 0), et les éventuels tableaux associés. Ces derniers seront imprimés par appel au sous-programme IMTASD. Selon la valeur de IMPRE, plus ou moins d'impressions sont obtenues. Pour avoir la liste des tableaux imprimés dans chaque cas en fonction de IMPRE consulter la brochure 2.

v) Sous-programmes particuliers : Les différents programmes particuliers qui aident la manipulation d'une S.D. sont décrits dans la brochure 2.

5.4.3.2. Les sous-programmes utilitaires généraux

i) Sous-programmes utiles pour les tableaux associés

Six sous-programmes permettent de manipuler les tableaux associés à une S.D.

Notation : dans la suite :

M : le vecteur de travail,

NFSD : numéro d'unité logique du fichier supportant les tableaux associés

LPAGE : nombre de mots par page si le fichier est d'accès direct,

NOSD : identificateur du deuxième tableau de la S.D., tableau descriptif des tableaux associés

NTASD : nombre de tableaux associés.

SUBROUTINE ECTASD (M, NFSD, LPAGE, NEF, NOSD, NTASD, NMOTAS)

Permet d'écrire, sur le fichier de numéro d'unité logique NFSD, les tableaux associés à une S.D. En sortie, NEF est le numéro du dernier enregistrement écrit et NMOTAS est le nombre de mots transférés.

SUBROUTINE LETASD (M, NFSD, LPAGE, NEF, NOSD, NTASD)

Permet d'adresser dans M et de lire, sur le fichier de numéro d'unité logique NFSD, les tableaux associés à une S.D. En entrée, NEF est le numéro du dernier enregistrement lu. En sortie, toutes les adresses des tableaux associés sont mises à jour dans NOSD.

SUBROUTINE RETASD (M, NFSD, NOSD, NTASD)

Permet de retrouver dans M les tableaux associés à une S.D. Les tableaux sont décrits dans NOSD qui joue le même rôle que le tableau de sauvegarde du commun de la S.D. L'argument NFSD est formel et peut être annulé ...

SUBROUTINE TUTASD (M, NOSD, NTASD)

Permet de retirer des tables les tableaux associés à une S.D.

SUBROUTINE COTASD (M, NTAC, NTASD, NTAB1, IATAB1, LTAB1)

Permet de créer les tableaux associés à une S.D. NTAC est le nombre des tableaux associés à créer à partir de cartes de données. NTAB1, IATAB1, LTAB1 sont les caractéristiques, dans les tables, du deuxième tableau de la S.D., décrivant ces tableaux associés.

SUBROUTINE IMTASD (M, NOSD, NTASD)

Permet d'imprimer les tableaux associés à une S.D.

Remarque :

On associe un tableau à une structure de données lorsque, pour un problème particulier, son utilisation devient nécessaire dans un module.

En règle générale les sous-programmes LETASD, RETASD, ECTASD, IMTASD ne sont pas directement appelés par le programmeur. Ils sont utilisés automatiquement via LETOTO, RETOTO, ECTOTO, IMTOTO. De même TUTASD est souvent appelé via TUERSD. Cette façon de procéder permet de réaliser une gestion systématique et simultanée de l'ensemble de la S.D. (tableaux et tableaux associés) ; l'utilisateur n'a pas à s'occuper de l'existence ou non des tableaux associés).

ii) Sous-programmes généraux concernant les COMMONS et l'utilisation aisée des S.D.

Généralités sur INICSD, SAUCSD

Les sous-programmes LETOTO, RETOTO, ECTOTO travaillent avec les valeurs contenues dans le COMMON associé ALTOTO sans tenir compte du numéro de niveau de la S.D. Quand plusieurs structures de données de même type sont utilisées (TOTO de niveau n et TOTO de niveau m) l'image du COMMON /ALTOTO/ décrivant la S.D. TOTO de niveau n est contenue dans le tableau de sauvegarde TOn% ; celle du COMMON /ALTOTO/ décrivant la S.D. TOTO de niveau m dans le tableau TOm%.

L'initialisation du COMMON, la création des tableaux de sauvegarde ainsi que les transferts entre COMMON et tableaux sont à la charge de l'utilisateur. Elles peuvent être exécutées à l'aide de deux sous-programmes utilitaires INICSD, SAUCSD. (Les notations de 5.4.3. sont utilisées).

INICSD

SUBROUTINE INICSD (M, NOMSD, NIVSD, NCOMMO, LCOMMO, NCSD, IACSD)

Cet utilitaire génère l'identificateur NCSD (TOn%) du tableau de sauvegarde et cherche dans le super tableau, un tableau portant ce nom.

- le **tableau existe** : le common est chargé avec les valeurs retrouvées dans le tableau.

- le **tableau n'existe pas** : le COMMON est initialisé automatiquement : les identificateurs des noms des tableaux sont créés à partir du nom de la S.D. et de son niveau ; les autres valeurs sont mises à zéro (adresse et longueur). Le tableau de sauvegarde du COMMON **n'est pas créé.**

Après exécution de INICSD il est possible de travailler avec les tableaux de la S.D. à partir des valeurs descriptives du COMMON

SAUCSD

SUBROUTINE SAUCSD (M, NOMSD, NIVSD, NCOMMO, LCOMMO, NCSD, IACSD)

Cet utilitaire génère l'identificateur NCSD du tableau de sauvegarde et adresse ce tableau dans les tables.

Il transporte ensuite le contenu du common dans ce tableau.

REMAROUE sur l'utilisation de INICSD, SAUCSD

Les sous-programmes INICSD, SAUCSD contiennent les paramètres NCOMMO. Donc, conformément à la REMAROUE la section de 5.4.1., les sous-programmes qui les utilisent font apparaître les instructions :

```
COMMON/ALTOTO/ NETOTO, NTOTO, IATOTO, ... NTOT5, IATOT5, LTOT5
DIMENSION NZTOTO (19)
EQUIVALENCE (NETOTO, NZTOTO(1))
.
.
CALL INICSD (M, 'TOTO', NITOTO, NZTOTO, 19, NCTOTO, ICTOTO)
```

□

Généralités sur RSTSDE, SAUSDS

Les sous-programmes **RSTSDE** et **SAUSDS** sont des **sous-programmes clef** de MODULEF. Ils sont **utilisés par tous les MODULES.** C'est par leur intermédiaire que s'effectuent tous les transferts des S.D. entre la mémoire centrale et les fichiers (lecture par LE..., écriture par EC...) et controles (par RE...). Leurs fonctions sont symétriques. L'un est associé aux opérations d'entrée et contrôle : RSTSDE, l'autre aux opérations de sortie : SAUCSD. En résumé seront "traitées" par RSTSDE les S.D.E. et par SAUSDS les S.D.S.

RSTSDE

But :

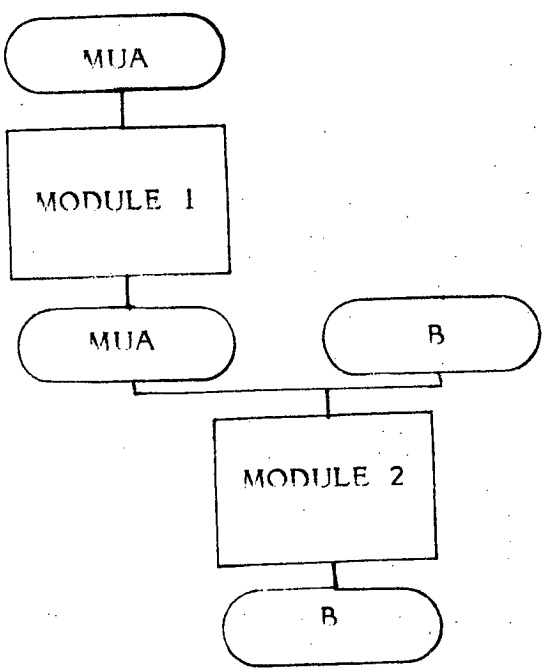
Les S.D. constituent les interfaces des modules. Ouand un module doit utiliser une S.D.E. plusieurs cas de figure sont possibles. En fait, l'état des COMMONS, l'existence ou non du tableau de sauvegarde dépendent du contexte spécifique ou est appelé le module donc ne peuvent pas être connus à priori par le programmeur du module. Le but du sp RSTSDE est d'effectuer toutes les taches pour qu'une S.D. soit utilisable (les tableaux en MC, le COMMON à jour).

. De plus le sous-programme effectue une vérification de cohérence entre les données du tableau 0 de la S.D. et celles du COMMON/TRAVA1/.

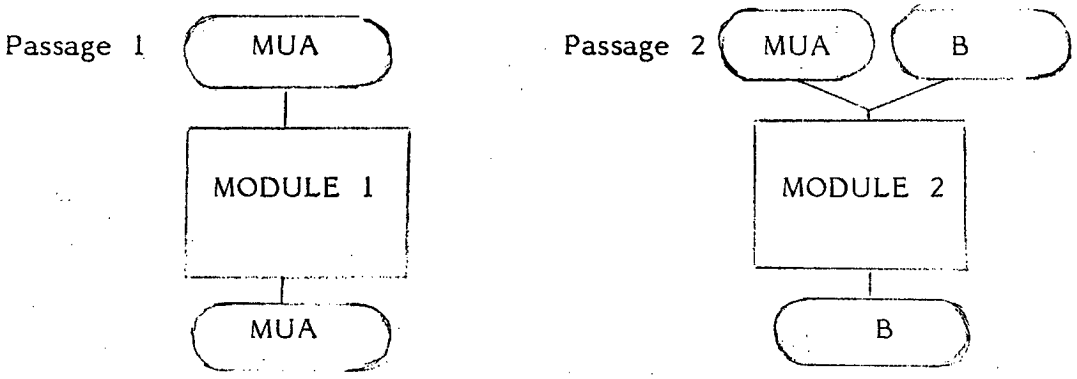
Un exemple simple :

Décomposons la résolution d'un système linéaire par une méthode directe en deux étapes : la factorisation (MODULE 1) et la descente-remontée (MODULE 2). Soit à résoudre le système linéaire Ax = b. A est contenu dans une S.D. MUA, b dans une S.D. B.

1) Toute la résolution est faite dans un même passage. Dans ce cas, le "programme d'appel" peut être schématisé par :



2) La résolution est faite dans deux passages différents qui peuvent être schématisés par :



Il est clair que dans ce deuxième cas de figure la S.D.S. MUA issue de MODULE 1 doit être stockée sur fichier pour être utilisée lors d'un autre passage par MODULE 2.

Nous nous intéressons à l'état du COMMON/ALMUA/, à l'existence ou non du tableau de sauvegarde MUn% en entrée du MODULE 2 ou, autrement dit, quelles sont les tâches à accomplir pour pouvoir utiliser MUA.

Cas 1

La S.D. est en mémoire centrale (puisque créée précédemment par le MODULE 1), donc le tableau de sauvegarde MUn% existe. Dans ce cas il faut : a) charger le tableau MUn% dans le COMMON (par INICSD par exemple)

b) vérifier que tous les tableaux de la S.D. sont bien en M.C. par REMUA. Ensuite la S.D. MUA est utilisable par le MODULE 2.

Cas 2

MODULE 2 est le premier module à exécuter, il n'y a donc rien en mémoire. Dans ce cas il faut :

a) initialiser le COMMON/ALMUA/. Une initialisation automatique est faite par INICSD. Dans des cas particuliers (très rares dans la pratique) l'utilisateur souhaite faire sa propre initialisation du COMMON. Ceci est possible. Il suffit de créer et remplir, à sa guise le tableau MUn%. Ensuite un appel de INICSD permet d'initialiser le COMMON .

b) lire la S.D. MUA par LEMUA

c) créer le tableau associé au COMMON MUn% et le remplir par SAUSDS.

Ensuite, la S.D. MUA est utilisable par le MODULE 2. D'un point de vue pratique la distinction entre le cas 1 et 2 est faite par l'intermédiaire du paramètre NFSD (=0 cas 1, ≠ 0 cas 2).

L'appel du sous-programme RSTSDE

SUBROUTINE RSTSDE (M, NOMSD, NFSD, NIVSD, NCOMMO, LCOMMO,
LE..., RE..., NCSD, IACSDE)

où LE..., RE... sont les noms des sous-programmes de lecture et contrôle associés à la S.D. de nom NOMSD. (Si NOMSD='TOTO' il s'agit de LETOTO, RETOTO)

SAUSDS

Le but de ce sous-programme est d'écrire sur fichier la S.D.S. d'un module si NFSD ≠ 0 et de mettre à jour ou créer (par appel à SAUCSD) le tableau associé au COMMON.

SUBROUTINE SAUSDS (M, NOMSD, NFSD, NIVSD, NCOMMO, LCOMMO,
NCSD, IACSD, EC..., NMOSD, NOPT)

REMARQUES SUR L'UTILISATION DE RSTSDE ET SAUSDS

. Les sous-programmes RSTSDE et SAUSDS utilisent le paramètre NCOMMO (remarque de 5.4.1.).

. Parmi les paramètres il y a des sous-programmes (EC..., RE..., LE...), ceux-ci doivent être déclarés en EXTERNAL dans les sous-programmes utilisant RSTSDE ou SAUSDS.

En résumé, si un programme utilise la S.D. TOTO via RSTSDE ou/et SAUSDS les instructions suivantes vont apparaître :

```
COMMON /ALTOTO/NETOTO, NTOTO ... LTOT5   (cf 5.4.1.)
DIMENSION NZTOTO(19)
EQUIVALENCE (NETOTO, NZTOTO(1))
.
.
EXTERNAL LETOTO, RETOTO, ECTOTO
.
.
CALL RSTSDE (M, 'TOTO', NFTOTO, NITOTO, NZTOTO, 19, LETOTO,
             RETOTO, NCTOTO, ICTOTO)
.
.
CALL SAUSDS (M, 'TOTO', NFTOTO, NITOTO, NZTOTO, 19, NCTOTO,
             ICTOTO, ECTOTO, NMOSD, NOPT)
```

. Dans tout ce qui suit, un appel à RSTSDE est appelé une RESTAURATION D'UNE S.D. ; un appel à SAUSDS est appelé une SAUVEGARDE D'UNE S.D.. Ces appellations sont abusives si NF = 0 ; dans ce cas il n'y a pas d'E/S.

□

iii) La copie d'une S.D. COPISD

```
SUBROUTINE COPISD (M, NOMSD, NTAB, NISD1, NC1, IANC1, NISD2,
                   NC2, IANC2)
```

Cet utilitaire permet de copier les NTAB premiers tableaux de la S.D. NOMSD de niveau NISD1, dans les NTAB premiers tableaux de la S.D. de même nom, mais de niveau NISD2. Le tableau de sauvegarde du commun de la S.D. d'entrée est NC1, d'adresse M(IANC1). Le sous-programme initialise le tableau de sauvegarde NC2 du commun de la S.D. de sortie, NC2 et IANC2 étant des arguments de sortie

Les paramètres NC1, IANC1, NC2, IANC2 sont éventuellement remis à jour.

Cela permet de réaliser une modification dans une structure de données sans avoir à la recharger en entier.

iv) La destruction de la S.D. TUERSD

SUBROUTINE TUERSD (M, NOMSD, NISD, NOPT)

Cet utilitaire libère en mémoire centrale la place occupée par les tableaux de la structure de données et efface leur trace dans les tables. Le tableau de sauvegarde du commun associé est lui-même effacé.

Si NOPT = 0 on conserve dans M les tableaux associés
Si NOPT = 1 on efface aussi les tableaux associés.

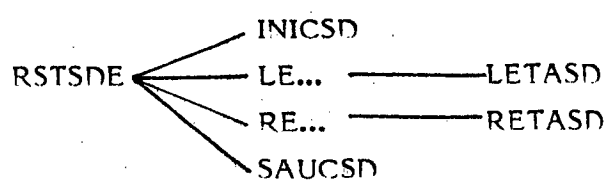
L'appel à cet utilitaire ne se fait jamais dans un module. C'est au programmeur d'un enchainement de modules à déterminer, dans son programme principal, le moment où la structure de données doit être détruite. Bien entendu, si une sauvegarde a préalablement été réalisée, la structure de données peut être reconstituée par la suite par la lecture (LE...) du fichier.

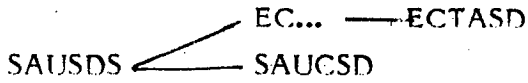
Si des tableaux associés existent, il est possible de faire appel à TUERSD avec l'option NOPT = 0 et appeler ensuite TUER pour chacun des tableaux associés à effacer.

5.4.4. REMAROUES SUR L'UTILISATION DES SOUS-PROGRAMMES DE GESTION

A chaque S.D. sont associés des utilitaires spécifiques ou généraux qui sont appelés directement ou par l'intermédiaire d'autres utilitaires. Une synthèse sur l'utilisation courante de ces utilitaires semble nécessaire pour permettre de répondre à la question **"qui (programme ou utilisateur) appelle quoi et à quel moment".**

Les arbres d'appel ci-dessus répondent partiellement à la question





En règle générale :

. LE..., EC..., RE..., RETASD, LETASD, TUTASD ne sont jamais appelés directement ; les 3 premiers sont indiqués comme paramètres de RSTSDE (SAUSDS) via l'EXTERNAL.

. RSTSDE, SAUSDS sont utilisés par le programmeur d'un MODULE pour toute restauration et/ou la sauvegarde de S.D.

. INICSD est directement utilisé par le programmeur d'un MODULE dans deux situations particulières qui correspondent à la création d'une S.D. nouvelle :

a) une S.D. n'existe pas. Elle sera créée par le module MODULE. L'initialisation automatique du COMMON se fait par appel à INICSD. Ensuite les tableaux seront adressés après calcul des longueurs. Ainsi toutes les valeurs du COMMON (noms, adresses, longueurs) seront initialisées.

b) une S.D. est créée à partir d'une autre de même type par copie COPISD et l'utilisateur désire se servir, dans la suite, de la S.D. nouvellement créé. La mise à jour du COMMON associé se fait encore par appel à INICSD.

. SAUCSD est utilisé directement dans des cas particuliers où le COMMON est encore modifié après sauvegarde de la S.D. (exemple : une S.D. de type 2 : après sauvegarde le 1er mot du tableau de sauvegarde (TOn%) contient le nombre d'enregistrements déjà écrits : mais, quand toutes les pages de la S.D. seront écrites cette valeur sera modifiée).

. TUERSD n'est pas appelé dans les MODULES mais, en dehors, dans les enchainements de MODULES. En effet la notion de S.D. encore utile est liée à un enchainement précis.

. IM... est généralement utilisé en dehors des MODULES.

5.4.5. **Documentation**

La brochure 2 (S.D.) contient la documentation relative aux S.D. En particulier elle décrit :

- le common associé à chaque S.D.
- les tableaux des S.D. et leur contenu.
- les utilitaires propres à chaque S.D.

6. LES FICHIERS EN ACCES DIRECT

En FORTRAN 66, les instructions permettant l'utilisation des fichiers en accès direct ne sont pas les mêmes sur toutes les machines. Pour avoir une utilisation standard (à savoir portable) des programmes Fortran qui utilisent de tels fichiers, il faut utiliser la bibliothèque MEFDIR décrite dans la brochure 42 : "Gestion des fichiers en accès direct (MEFDIR)". Cette bibliothèque contient des utilitaires **non transportables** permettant d'utiliser de façon standard les fichiers en accès direct.

A l'heure actuelle cette bibliothèque existe pour plusieurs types d'ordinateurs.

En FORTRAN 77, les instructions permettant l'utilisation de l'accès direct sont standard. La bibliothèque MEFDIR devient donc inutile. En fait pour avoir la continuité les programmes de lecture de MEFDIR sont transformés (font appel à un READ standard) etc... Avec ce système les anciens modules restent **inchangés** : les **nouveaux** peuvent être écrits en FORTRAN standard.

7. MODULE - SOUS-PROGRAMME - ALGORITHME

7.1. Généralités

Un MODULE est un ensemble de sous-programmes qui réalisent une étape importante de calcul.

Le **module** est la réalisation informatique d'un opérateur mathématique. Lors de la création d'un module il faut respecter au mieux les principes suivants :

i) chaque opérateur mathématique (voir 1.2.1.) doit être le plus spécifique possible de façon à ce que le module puisse être utilisé dans des contextes multiples

ii) séparation de l'algorithme numérique et de la gestion des tableaux et S.D.

iii) éviter les cartes de données : préférer le passage des valeurs par tableaux.

Précisons rapidement les points i) et ii) :

i) Spécificité de chaque opérateur :

A l'opérateur "résolution d'un système linéaire de matrice stockée profil par CHOLESKY" on va préférer la suite d'opérateurs :

1) construction des pointeurs de la matrice profil
2) assemblage de la matrice profil
3) prise en compte des conditions aux limites
4) factorisation
5) descente-remontée.

Ce choix répond au souci de modularité et permet une grande souplesse d'utilisation. Par exemple dans le cas de la résolution d'un système itératif où la matrice est constante au cours des itérations les opérateurs 1 à 4 seront utilisés une seule fois ; le 5 autant de fois qu'il y a d'itérations.

ii) séparation de l'algorithme numérique et de la gestion

Le respect du principe de la séparation de l'algorithme numérique et de la gestion a l'avantage de rendre les programmes fiables et plus faciles à lire. D'autre part, les sous-programmes qui contiennent l'algorithme sont écrits en FORTRAN standard ce qui permet de les utiliser en tant que partie d'une **bibliothèque de programmes.**

Cette séparation est possible dans le cas des structures de données de type 1 mais est impossible dans le cas des S.D. de type 2 qui contiennent un tableau paginé. En effet, dans ce cas, pour retrouver les valeurs intéressantes, il faut tenir compte de la structuration des données.

En règle générale la structuration d'un module est représentée dans la figure ci-dessous :

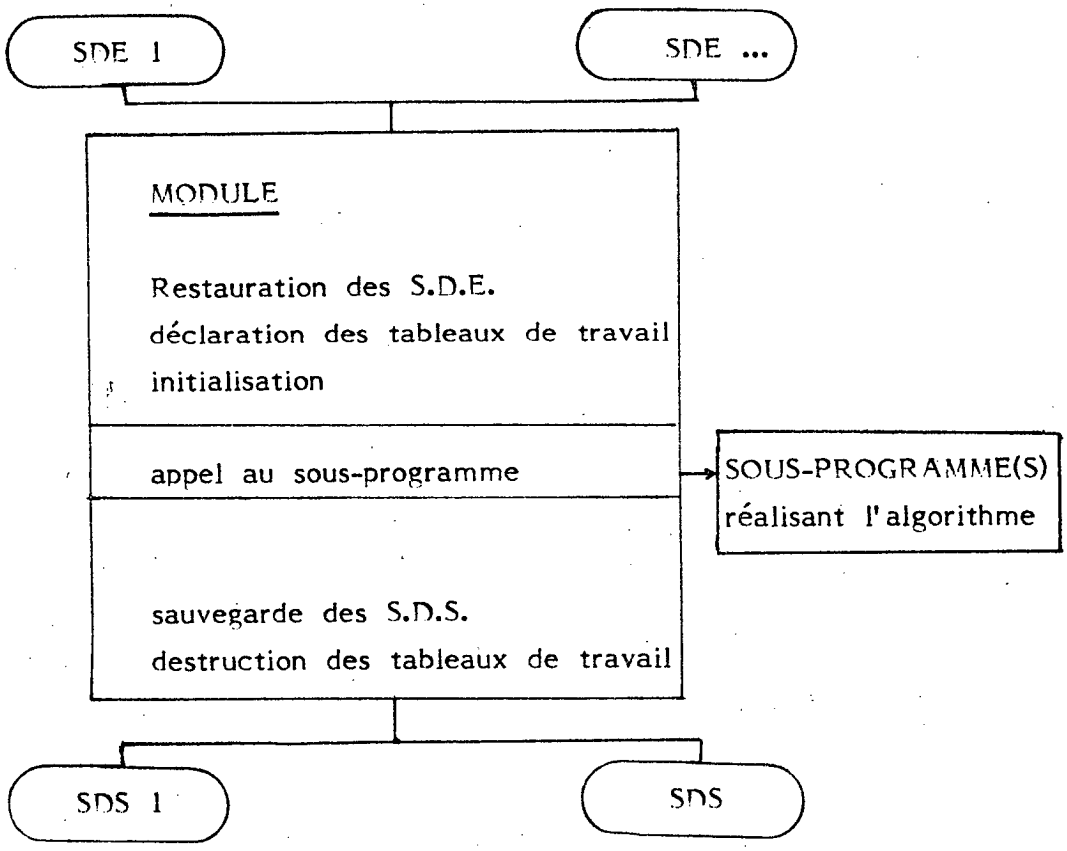


Figure 1

7.2. Normes de programmation d'un module

Respecter les normes de programmation d'un module signifie respecter les principes i), ii), iii) de 7.1. et utiliser la gestion dynamique pour les tableaux.

Dans la suite nous présentons les différentes étapes qui apparaissent dans la figure 1 du 7.1. et nous décrivons les utilitaires nécessaires à leur réalisation.

7.2.1. Les commons

7.2.1.1. Description

Les trois COMMONS suivants apparaissent dans la plupart des modules existants, mais **ne sont plus nécessaires et sont même à éviter** pour les nouveaux développements.

COMMON /TRAVAI/
COMMON /TRAVA1/
COMMON /UNITES/

i) COMMON /TRAVAI/ IM, LM, NOMTAB, NNN, NTYTAB(204), NTAB(204), IATAB(204), LTAB(204), NPERMU(204)

C'est le COMMON associé à la gestion dynamique. Il a été décrit en 4.2.2. La seule variable qui peut être modifiée par l'utilisateur est NNN. Elle est initialisée par INITI.

ii) COMMON /TRAVA1/ NTITRE(20), NDATE(2), NOMCRE(6), LONRE, IMPRE, IMPLEC, INILEC, IFIL

où

NTITRE(20)	contient les 80 caractères du titre du travail
NDATE(2)	contient les 8 caractères de la date du travail
NOMCRE(6)	contient les 24 caractères du nom de l'utilisateur
LONREE	est un paramètre (LONgueur des REEls) qui est égal à 1, en simple précision, 2 en double précision.
IMPRE	est un paramètre d'impression des informations de chaque module. Sa valeur peut varier de 0 à 10 ; c'est au programmeur du module à en faire l'usage qui lui semble convenable, de manière à rendre progressivement plus explicites les impressions intermédiaires en cas de recherche d'erreur

IMPLEC est le paramètre d'impression pour le format libre, tel que
IMPLEC 0, il n'y a pas d'impression
IMPLEC = 1, chaque carte lue est imprimée
IMPLEC = 2, de plus, chaque donnée lue est imprimée.

INILEC est un pointeur permettant de compter les cartes lues ; il doit être égal à zéro au départ et ne doit pas être modifié par l'utilisateur.

IFIL correspond à une réservation de paramètre.

iii) COMMON /UNITES/LECTEU, IMPRIM, INTERA, IMPFIC, PADUNI(28)

avec

LECTEU numéro logique du lecteur (initialisé dans INITI)

IMPRIM numéro logique de l'imprimante (initialisé dans INITI)

INTERA interactivité 0 batch
1 interactif
2 graphique

IMPFIC numéro logique du deuxième fichier de résultats.
Cette valeur est initialisée dans INITI. En version standard elle est 99.

REMAROUES

. Si l'utilisateur désire conserver les "sorties" d'un module sur un fichier plutôt que sur l'imprimante il peut modifier la valeur d'IMPRIM en donnant le numéro logique du fichier de sortie.

. IMPFIC permet une utilisation similaire. En fait il sera possible d'utiliser deux "sorties" une sur IMPRIM, l'autre sur IMPFIC cette dernière à utiliser plus particulièrement en cas de besoin.

7.2.1.2. Les utilitaires

Certaines variables de ces COMMONS peuvent être consultées ou modifiées par le programmeur ce qui peut se faire soit **directement,** soit en utilisant les **fonctions** LIBCMD et INFO décrites dans "Conversion de Modulef en Fortran 77", P. LAUG brochure n°109. Il est recommandé d'utilser cette dernière possibilité dont l'intérêt est que :

il n'est plus nécessaire de faire figurer les COMMONS ;

si le contenu des commons vient à changer seules ces fonctions devront être modifiées.

7.2.2. Manipulation des structures de données

7.2.2.1. Généralités

Le traitement de plusieurs S.D. de même nom est présenté au paragraphe 7.2.2.2.

La restauration des S.D. se fait par appel du sous-programme RSTSDE (cf 54.3.2. ii)) ; la **sauvegarde** par appel de SAUSDS (cf 5.4.3.2. et ii)).

La création d'une S.D. (qui sera une S.D.S. du module) : le module va contenir le COMMON associé à la S.D. (cf 5.4.1.). Pour créer une S.D. il faudra :

i) **initialiser** le COMMON (via INICSD cf 5.4.3.2. ii) par exemple)

ii) **adresser** les tableaux de la S.D. La longueur des tableaux est généralement connue en fonction des paramètres d'entrée du module et/ou des valeurs contenues dans les S.D.E. Si, éventuellement, la taille d'un tableau n'est pas connue elle est majorée soit à l'aide des paramètres connus, soit en utilisant toute la place disponible (cf MAXTAM 4.2.3.1. v)).

iii) **remplir** les tableaux de la S.D. Les "initialisations" (en général le contenu du tableau 2 de la S.D.) peuvent être faites dans le module. Le(s) tableau(x) significatifs sont remplis dans les sous-programmes qui contiennent l'algorithme.

iv) **réajuster** la taille des tableaux majorés (par exemple READRE cf 4.2.3.1. vi))

Attention : ne pas oublier cette étape, surtout lorsque MAXTAM a été utilisé.

v) **sauvegarder** la S.D. via SAUSDS

REMARQUES

Les étapes iv), v) sont parfois interversées. C'est, en particulier le cas lorsque une S.D. de type 2 est utilisé.

Soit une S.D. TOTO de type 2 dont le dernier tableau (TOT5 dans l'exemple) est segmenté. Les premiers tableaux doivent être sauvegardés **avant** appel de l'algorithme. Les différentes pages du dernier tableau seront écrites dans le sous-programme qui renferme l'algorithme.

Dans ce cas, le **réajustement** du tableau se fait après la **sauvegarde.** Il faut, donc, après réajustement, faire **une mise à jour du tableau associé** à la S.D. par appel de SAUCSD.

Les utilitaires utilisés fréquemment lors de la création d'une nouvelle S.D.

SUBROUTINE INTAB0 (NOMSD, NIVSD, NTASD, NTABO)

Cet utilitaire permet de remplir le tableau 0 de la S.D.

où

NOMSD : est le nom de l'identificateur de la structure de données
NIVSD : est son numéro de niveau
NTASD : est le nombre de ses tableaux associés
NTABO : est, en sortie, le premier tableau de la structure de données

SUBROUTINE COTASD (M, NTAC, NTASD, NTAB1, IATAB1, LTAB1)

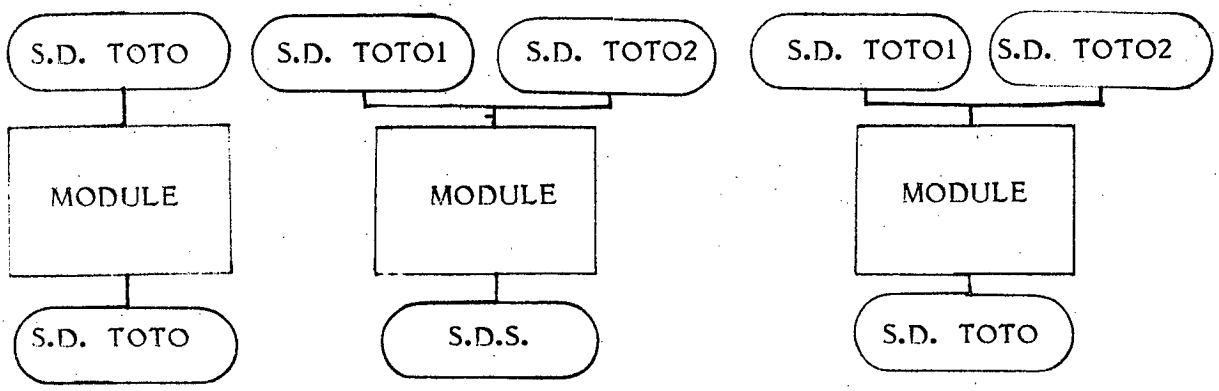
Cet utilitaire permet de créer NTAC tableaux associés, lus sur des cartes de données. Cette possibilité est offerte à l'utilisateur lors de chaque **création** d'une S.D. par l'intermédiaire du paramètre NTAC.

NTAC : nombre tableaux associés (souvent NTAC=NTASD sauf cas exceptionnel où le programmeur associe des tableaux par programme).
NTAB1 : nom du tableau de la S.D.
IATAB1 : adresse du premier tableau
LTAB1 : longueur du 1er tableau (22*NTASD)

Le sous-programme COTASD **adresse** et **remplit** le tableau 1 de la S.D. et il adresse et lit les tableaux associés.

7.2.2.2. Traitement de plusieurs S.D. de même nom par un module

Plusieurs cas de figure sont envisageables. Ils sont présentés dans la figure 1.



Cas A Cas B Cas C

Figure 1

Quel que soit le cas de figure la règle est toujours la même : Deux S.D. de même nom **sont confondues** en M.C. si elles **ont le même niveau** et sont **distinctes** si elles ont **des niveaux différents.**

Cette règle découle des conventions adoptées pour donner les noms des tableaux des S.D: (rappel TOni) → numéro du tableau (cf 5.3)

niveau

nom S.D.

REMARQUES

i) Dans tous les cas à chaque S.D. sont associées deux paramètres entiers : **le numéro de support et le niveau.**

ii) Quand une seule S.D. de nom TOTO apparait dans un module pour utiliser ses tableaux il est possible de travailler soit avec le **COMMON associé** soit avec **le tableau de sauvegarde** (rappelons que les utilitaires travaillent avec le COMMON).

iii) Quand plusieurs S.D. de même nom apparaissent il faut travailler avec le **tableau de sauvegarde** (TOn%) ou avec le COMMON mais en faisant les transferts tableau de sauvegarde COMMON (via INICSD, SAUCSD) chaque fois que c'est nécessaire.

iv) Certains modules acceptent que deux S.D. de même nom aient le même niveau, d'autres pas. Ces renseignements sont contenus dans la documentation spécifique à chaque module.

En règle générale, deux **S.D. de même nom peuvent avoir le même niveau si l'algorithme le permet.**

EXEMPLE : factorisation d'une matrice ; les matrices initiale et factorisée peuvent coïncider physiquement.

Dans ce cas le module peut être utilisé soit avec :
- deux S.D. de même niveau (la S.D.S. **écrase** la S.D.E.)
- deux S.D. de niveaux différents (il y a effectivement 2 matrices en M.C.) dans ce deuxième cas l'appel de l'algorithme est précédé par la copie (via COPISD) de la S.D.E. de niveau n1 dans la S.D.S. de niveau n2.

C'est à l'utilisateur qui réalise l'enchainement des modules de choisir quelle est la configuration la mieux adaptée à ses besoins.

7.2.2.3. Le support d'une S.D.

Toute utilisation d'une S.D. par un module nécessite la donnée d'un **numéro de support** (NFTOTO). Rappelons la convention sur les numéros de support :

NFTOTO = 0 traitement en MC.

> 0 la S.D.E. sera lue sur le fichier séquentiel de numéro logique NFTOTO
la S.D.S. sera sauvegardée sur le fichier séquentiel de numéro logique NFTOTO

< 0 la S.D.E. sera lue sur le fichier en accès direct de numéro logique -NFTOTO
la S.D.S. sera écrite sur le fichier en accès direct de numéro logique -NFTOTO.

Si un fichier en accès direct est utilisé il doit être **ouvert** avant toute opération et **fermé** en fin d'utilisation.

Si le fichier en accès direct est le support d'une **S.D.E. l'ouverture** se fait dans le sous-programme LETOTO par contre si c'est une S.D.S. cette opération doit être effectuée dans le module.

Pour la description des utilitaires concernant l'accès direct se reporter à P. LAUG documentation n°42.

7.2.3. Les tableaux de travail

Avant l'appel de l'algorithme, le programmeur d'un module doit recenser et adresser tous les tableaux de travail dont il aura besoin par la suite. Pour le faire il convient de :

i) choisir un nom (de préférence commençant par ':')

ii) initialiser l'adresse à zéro

iii) calculer la longueur du tableau

iv) adresser le tableau (via ADRESS ou READRE)

Si la taille d'un tableau ne peut pas être calculée (ou majorée) il faut adresser ce tableau en dernier et lui consacrer toute la place disponible (via MAXTAM) puis faire in réajustement.

v) le tableau sera passé en paramètres des différents sous-programmes par l'intermédiaire de son adresse M(IATAB) ce qui permet une utilisation standard dans le sous-programme.

REMARQUE : il ne faut jamais utiliser IM (première adresse libre derrière les tableaux, valeur contenue dans le COMMON/TRAVAI/) pour adresser un tableau. L'unique utilisation directe de ce paramètre (qui figure dans les impressions de l'adressage) est qu'il permet de connaître la taille mémoire nécessaire pour un enchainement déterminé. (Dans ce cas, attention à l'utilisation de MAXTAM !)

vi) En fin de module tous les tableaux de travail seront détruits (par TUER) et la place est libérée.

7.3. EXEMPLE : un exemple de MODULE commenté figure en annexe.

8. SPECIFICATIONS POUR L'ECRITURE D'UN MODULE OU D'UN SOUS-PROGRAMME

8.1. Généralités

Il ne s'agit pas de contraindre l'utilisateur ! Les quelques règles qui suivent ont comme but de rendre aisée la lecture des programmes. Cet objectif demande un minimum de standardisation.

8.2. La présentation

Tout module ou sous-programme débute par la carte SUBROUTINE (ou FUNCTION) suivie de **cartes commentaires** qui contiennent le **but** du sous-programme et la **signification des paramètres.** Il convient de bien séparer les paramètres d'entrée, de sortie et ceux dont la valeur est modifiée par le sous-programme.

8.3. Les commentaires

La lecture d'un programme est facilitée par la présence de commentaires. Les commentaires doivent être clairs et significatifs. Il est souhaitable que la seule lecture de commentaires donne les grandes lignes développées dans l'algorithme.

8.4. Les mnémoniques

Les noms **des programmes doivent** être le plus simple et le plus parlant possible. Avant de donner un nom à un programme il convient de s'assurer que ce nom n'existe pas déjà. Il est possible de faire cette opération en consultant le **listing des procédures** fournis sur chaque bande et qui contient la liste complète des sous-programmes et des bibliothèques auxquelles ils appartiennent.

Les mnémoniques qui sont utilisées dans les programmes doivent être les mêmes que ceux utilisés dans la documentation du module et, dans la mesure du possible, respecter les "habitudes" MODULEF. (Le nombre d'éléments sera désigné par NE plutôt que NELEM, NEL etc...).

8.5. Les diagnostics d'erreurs

Si une erreur est détectée, le format qui la signale doit contenir le nom du sous-programme et, si possible le remède et être clair.

8.6. Les astuces de programmation

Eviter au maximum les astuces de programmation ou alors bien les expliquer !

ANNEXE

I EXEMPLE DE RESOLUTION D'UN PROBLEME CONCRET PAR MODULEF

Un exemple simple va nous permettre de fixer les notations et d'expliciter plus en détail la marche à suivre pour la résolution d'un problème à l'aide de MODULEF.

Soit à résoudre le problème de Dirichlet homogène suivant :

1. EQUATIONS AUX DERIVEES PARTIELLES

$$(P) \quad \begin{cases} -\Delta u + a\,u = f \text{ dans } \Omega \in \mathbb{R}^2 \\ u = 0 \qquad\qquad \text{sur } \Gamma \end{cases}$$

avec $f \in L^2(\Omega)$, $a \geq 0$ p.p $a \in L^\infty(\Omega)$ où Ω est un ouvert régulier de $\mathbb{R}^2$ de frontière Γ.

2. FORMULATION VARIATIONNELLE .

Une formulation variationnelle possible de ce problème est :

$$(P') \quad \begin{cases} \text{Trouver } u \in H_0^1(\Omega) \text{ tel que} \\ \displaystyle\int_\Omega (\Delta u\, \Delta v + a\,u\,v)\,dx \quad \int_\Omega f\,v\,dx \;\; \forall\, v \in H_0^1(\Omega) \end{cases}$$

Pour tous les détails concernant l'analyse et les solutions de (P) et (P') consulter, par exemple, "The Finite Element Method For Elliptic Problems", Ph. CIARLET, North-Holland 1978.

3. L'ALGORITHME MATHEMATIQUE

A première vue un problème de ce type n'est pas complètement résolu dans MODULEF. Une analyse plus fine permet de l'assimiler à un problème de type THERMIQUE.

Brièvement, la décomposition de ce problème, en utilisant la formulation classique "en déplacements" et une écriture matricielle, est la suivante : on cherche une solution du problème discret :

$$(P'') \quad \begin{cases} \text{Trouver } u_h \in V_{0h} \text{ tel que} \\ \int_\Omega (\nabla u_h \nabla v_h + a\, u_h\, N_h)\, dx = \int_\Omega f\, v_h \in \forall\, v_h \in V_{0h} \end{cases}$$

où les ensembles V_h, V_{0h} sont définis par :

(1) $V_h = \{v_h \mid v_h \in \mathscr{C}^0(\bar{\Omega}),\quad v_h|_T = P_k,\ \forall\, T \in \mathscr{T}_h\}$

(2) $V_{0h} = \{v_h \mid v_h \in V_h\ ,\ V_h|_\Gamma = 0\}$

et P_k est un polynôme (de degré $\leq$ k, dépendant du choix de l'élément fini).

Les espaces fonctionnels utilisés dans (P") sont **de dimension finie.**

NOTATIONS

. $\mathscr{T}_h$ est une triangulation de $\bar{\Omega}$ élément courant T ($\bar{\Omega} = \bigcup_{T \in \mathscr{T}_h} T$)

. dim $V_h = \tilde{N}$: $\tilde{N}$ désigne donc le nombre total de degrés de liberté

. $\{\tilde{u}\}$ désigne l'ensemble des degrés de liberté (numérotation globale)

. $\{u_T\}$ désigne les degrés de liberté de l'élément T (numérotation locale)

. L'application $B_T : R^{\tilde{N}} \to R^N$, définie sur chaque élément, associe à l'ensemble des numéros "locaux" des d.l. de l'élément T, l'ensemble correspondant des numéros "globaux" de ces mêmes d.l.

$$\{u_T\} = [B_T]\ \{\tilde{u}\}$$

. N_h est l'ensemble des degrés de liberté appartenant à la frontière

. $\tilde{i} \in \tilde{N}$ (où $\tilde{i} \in N_h$) sont des notations qui signifient qu'un degré de liberté de numéro global $\tilde{i}$ appartient à la triangulation (respectivement à la frontière).

. Si on considère une base de V_h telle que $w_i(j) = \delta_{ij}$ alors :

$$u_h(x) = \sum_{\tilde{i}=1}^{\tilde{N}} u_h(\tilde{i}).\, w_i(x) = \sum_{\tilde{i} \in \tilde{N}-N_h} u_h(\tilde{i})\, w_{\tilde{i}}(x) \quad \text{car } u|_\Gamma = 0$$

En notation vectorielle ($\tilde{u_i}$ étant les composantes de u_h sur la base w_i)

$\{\tilde{u}\}$ sécrit :

$$\{\tilde{u}\} = \left[u_{\tilde{1}}, \ldots\, u_{\tilde{i}}, \ldots\, u_{\tilde{N}-N_h}, 0 \ldots 0\right]$$

en supposant la numérotation telle que les degrés de liberté situés sur la frontière soient numérotés en dernier.

De plus, comme la restriction de u_h à un triangle est un polynôme

$$u_h|_T = \left[P(x)\right] \{u_T\}$$

avec

$$\left[P\right] = \left[P_1, \ldots\, P_N\right]$$

avec N = dimension de l'espace = nombre de degrés de liberté (3 pour des polynômes d'un élément de degré 1 sur un triangle, 4 sur un quadrilatère ...)

. $$\left[Du_h\right] = \begin{bmatrix} \dfrac{\partial u_h}{\partial x} \\ \dfrac{\partial u_h}{\partial y} \end{bmatrix}$$

d'où $\left[Du_h\right] = \left[DP\right] \{u_T\}$

avec $\left[DP\right] = \begin{bmatrix} \frac{\partial p_1}{\partial x} & \cdots\cdots & \frac{\partial p_N}{\partial x} \\ \frac{\partial p_1}{\partial y} & \cdots\cdots & \frac{\partial p_N}{\partial y} \end{bmatrix}$

Avec ces notations le problème (P") s'écrit :

Trouver $\{\tilde{u}\}$ qui vérifie $\tilde{u}_{\tilde{i}} = 0 \quad \tilde{i} \in N_h$

(3)
$$\begin{cases} {}^t\{\tilde{v}\} \left(\sum_{T \in \mathcal{T}_h} {}^t\left[B_T\right] \left(\int_T \left({}^t\left[DP\right]\left[DP\right] + a\, {}^t\left[P\right]\left[P\right] \right) d\Omega \right) \left[B_T\right] \{\tilde{u}\} \right) = \\ {}^t\{\tilde{v}\} \left(\sum_{T \in \mathcal{T}_h} {}^t\left[B_T\right] \left(\int_T {}^t\left[P\right] f\, d\Omega \right) \right) \quad \text{pour tout } \{\tilde{v}\} \text{ tel que } \tilde{v}_{\tilde{i}} = 0 \; \tilde{i} \in N_h \end{cases}$$

A ce stade, le passage de la formulation (P") à la formulation (3) est tout à fait général **(indépendant de MODULEF).** Il reste à exhiber les **opérateurs mathématiques** correspondants, puis à trouver leurs équivalents dans MODULEF.

Suivant la documentation (100) : (INTRODUCTION : pages 11-13) il est loisible de réaliser dans Modulef les calculs élémentaires suivants :

$$\int_T {}^t\left[DP\right]\left[DP\right] d\Omega = \left[K_T\right] \quad \text{avec} \quad \left[K\right] = I \;,\; g = 0$$

$$\int_T {}^t\left[P\right]\left[P\right] d\Omega = \left[M_T\right] \quad \text{avec} \quad \rho = 1$$

$$\int_T {}^t\left[P\right] f\, d\Omega = \{b_T\} \quad \text{avec} \quad f^{\Gamma} = 0$$

Ceci est la **définition** d'un **élément fini** dans MODULEF.

En combinant ces différents calculs élémentaires, on implémente ainsi la formulation (3), ce qui montre que ce problème discret

. peut être entièrement résolu par MODULEF

. est assimilé à un problème de THERMIQUE.

La documentation (100) contient la liste et la description des éléments finis de type THERMIQUE disponibles.

Avec les notations ci-dessus, (3) devient :

Trouver $\{\tilde{u}\}$ qui vérifie $\tilde{u}_{\tilde{i}} = 0 \quad \tilde{i} \in N_h$

$$(4) \qquad {}^t\{\tilde{v}\}\Big(\sum_{T\in\mathcal{T}_h} {}^t\left[B_T\right]\left(\left[K_T\right] + a\left[M_T\right]\right)\{\tilde{u}\} = {}^t\{\tilde{v}\}\Big(\sum_{T\in\mathcal{T}_h} {}^t\left[B_T\right]\{b_t\}\Big) \ ;$$

c'est un système linéaire dont la matrice est symétrique, définie positive.

Désignons par $\left[A_T\right]$ une **matrice élémentaire** attachée au triangle T. A l'ensemble de ces matrices $\left[A_T\right]$ on associe la matrice :

$$(4') \qquad \left[\tilde{A}\right] = \sum_{T\in\mathcal{T}_h} {}^t\left[B_T\right]\left[A_T\right]\left[B_T\right],$$

ce qui constitue l'opération **d'assemblage.** Cette opération est indépendante de la structure de $\left[A_T\right]$.

REMARQUE : Il y a deux possibilités dans cet exemple :

Si $\tilde{A}$ est la matrice du système à résoudre, on peut l'écrire :

$$(i) \qquad \left[\tilde{A}\right] = \sum_{T\in\mathcal{T}_h} {}^t\left[B_T\right]\left[K_T\right]\left[B_T\right] + a\sum_{T\in\mathcal{T}_h} {}^t\left[B_T\right]\left[M_T\right]\left[B_T\right] = \left[K\right] + a\left[M\right]$$

ou bien

$$(ii) \begin{cases} \left[A_T\right] = \left[K_T\right] + a\left[M_T\right] \\ \left[\tilde{A}\right] = \sum_{T\in\mathcal{T}_h} {}^t\left[B_T\right]\left[A_T\right]\left[B_T\right] \end{cases}$$

D'un point de vue mathématiques, ces formules sont identiques. Elles peuvent déboucher sur deux réalisations informatiques différentes.

Dans les deux cas il faut calculer les **matrices élémentaires de masse** $\left[M_T\right]$ et les **matrices élémentaires de rigidité** $\left[K_T\right]$ puis réaliser :

i) - l'assemblage de $\left[M_T\right]$ qui donne $\left[M\right]$

- l'assemblage de $\left[K_T\right]$ qui donne $\left[K\right]$

- la combinaison de deux matrices (qui ont un stockage particulier)

ou :

ii) - la combinaison linéaire des matrices élémentaires

- l'assemblage de la matrice obtenue précédemment.

L'une ou l'autre des deux possibilités ci-dessus sera adoptée en fonction de la nature du problème à traiter.

□

Après assemblage de la matrice du système, la résolution de (4) s'écrit :

Trouver $\{\tilde{u}\}$, vérifiant $\tilde{u}_{\tilde{i}} = 0 \quad \tilde{i} \in N_h$, tel que

(5) $\quad {}^t\{\tilde{v}\}\left[\tilde{A}\right]\{\tilde{u}\} = {}^t\{\tilde{v}\}\left[\tilde{B}\right], \quad \forall\{\tilde{v}\}$ vérifiant $\tilde{v}_{\tilde{i}} = 0 \quad \forall_{\tilde{i}} \in N_h$

Rappelons que, sur la base $\{w_i\}$ de V_h, $\{\tilde{u}\} \in V_{oh}$ s'écrit :

(6) $\quad \{\tilde{u}\} = \left[\tilde{u}_1, \tilde{u}_2 \ldots, u_{\tilde{N}-N_h}, 0 \ldots 0\right]$

On considère alors une partition de $\tilde{A}$ et de $\tilde{B}$

$$\left[\tilde{A}\right] = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{bmatrix} \qquad \left[\tilde{B}\right] = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

où $\tilde{A}_{11}$ est une matrice $(\tilde{N}\text{-}N_h) \times (\tilde{N}\text{-}N_h)$.

Avec cette décomposition, en tenant compte des propriétés de $\{\tilde{u}\}$ et $\{\tilde{v}\}$ qui sont dans V_{0h}, (6) est équivalent à :

$$(7) \qquad \underbrace{\begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ 0 & I \end{bmatrix}}_{\tilde{A}_1} \{\tilde{u}\} = \begin{bmatrix} B_1 \\ 0 \end{bmatrix}$$

Donc pour calculer $\{\tilde{u}\}$ il faut résoudre (7) ou bien :

$$(8) \qquad \left[\tilde{A}_{11}\right] \{\tilde{u}_L\} = \left[B_1\right]$$

où u_L sont les $\tilde{N}\text{-}N_h$ premières composantes de $\{\tilde{u}\}$ (les N_h dernières sont nulles).

REMARQUES :

. D'un point de vue mathématique la relation (7) implique (8). Pour obtenir u , il est possible de résoudre (7) ou (8). D'un point de vue pratique (informatique) on préfère (7) à (8) car (8) nécessite une deuxième numérotation des degrés de liberté ou ceux bloqués ne sont pas comptés (ou alors sont effectivement numérotés en dernier ce qui est incompatible avec une minimisation de la largeur de bande.

. Le calcul direct de $\left[\tilde{A}_1\right]$ (la matrice de (7)) par assemblage des $T \in \mathcal{T}_h$ nécessite une distinction entre les triangles $T \in \mathcal{T}_h$ tels que $T \cap \Gamma = \phi$ et $T \cap \Gamma \neq \phi$ pour prendre en compte la condition $u_h|_\Gamma = 0$.

Il est donc plus simple de calculer d'abord $\left[\tilde{A}\right]$ et de passer ensuite à $\left[\tilde{A}_1\right]$. Ce passage est appelé **prise en compte des conditions aux limites.**

□

L'analyse précédente permet d'extraire **les opérateurs mathématiques.** Le problème se décompose donc en **plusieurs étapes :**

I) maillage du domaine
II) choix des éléments finis
III) données physiques, création des tableaux élémentaires
IV) résolution du problème linéaire
V) visualisation des résultats

Remarque : I) et II) sont issus de la formulation variationnelle choisie.

Chaque **étape** se décompose **en sous-étapes**

Nous présenterons dans la suite brièvement **la décomposition** de chaque étape et le choix **des modules** qui permettent la réalisation informatique de l'opérateur mathématique.

Définissons d'abord la **terminologie** MODULEF.

INCONNUE VARIATIONNELLE : inconnue du problème continu

DEGRE DE LIBERTE (d.l.) inconnue du système discret (ici du système linéaire)

MNEMONIQUE associé à un d.l. pour le définir, en clair Ex VN valeur au noeud, DX dérivée par rapport à x... La liste complète figure dans le tableau MAI8 de la S.D. MAIL (cf brochure 2 : Description des Structures de données.

NOEUD (NNO) : support des degrés de liberté

POINT (NP) : "point" permettant de définir la géométrie de l'élément

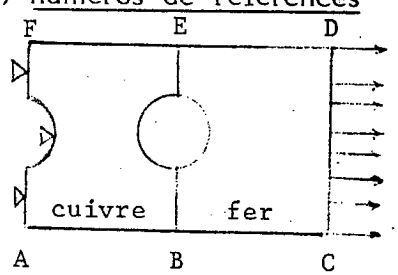
SOMMET : sens géométrique usuel

NUMERO DE SOUS-DOMAINE : numéro associé à chaque matériau permettant de distinguer des sous-domaines de caractéristiques différentes ou des traitements différents d'un même matériau.

NUMERO DE REFERENCE : notion similaire à celle de sous-domaine mais s'appliquant cette fois-ci aux frontières (surfaces dans (IR^3), lignes dans (IR^2 ou IR^3). Ces numéros permettent de traiter "en bloc" des conditions aux limites, des efforts ou bien de définir des lignes (surfaces) courbes.

EXEMPLES :

1) numéros de références



La pièce de la figure ci-dessus est encastrée sur la frontière AF et tirée sur la frontière CD. Elle est composée de deux matériaux et a un trou circulaire.

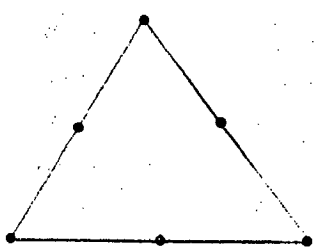
Il y aura **2 sous-domaines** 1 cuivre domaine ABEF
2 fer domaine BCDE

au moins **3 numéros de référence** 1 pour définir le cercle qui délimite le le trou et la ligne qui partage le domaine en deux sous-domaines à des fins de post-traitement.
2 pour définir la portion encastrée de la frontière
3 pour définir la droite DC où un effort sera appliqué.

Remarque :

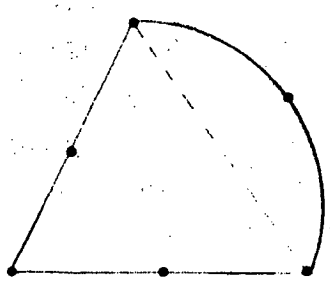
Les lignes AC et FD n'ont pas de numéro de référence car aucunes conditions particulières ne s'y appliquent.

2) noeuds et points



Triangle P2 droit

$$\begin{cases} NNO = 6 \\ NPO = 3 \end{cases}$$



Triangle P2 courbe

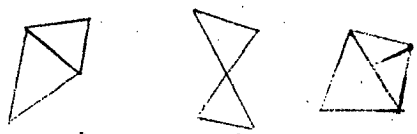
$$\begin{cases} NNO = 6 \\ NPO = 6 \end{cases}$$

4. ASSOCIATION OPERATEUR MATHEMATIQUE - OPERATEUR INFORMATIQUE

I) Maillage du domaine

Cette étape est la plus simple en tant qu'opérateur mathématique. La discrétisation doit respecter les règles générales d'une triangulation d'éléments finis :

- $T_1 \cap T_2 =$ { soit l'ensemble vide / soit 1 sommet, i.e., / soit 1 arête



oui oui non

- pas d'angle trop petit.

L'implémentation correspondante est un problème non trivial qui même lorsqu'on dispose de tous les opérateurs nécessaires dans la bibliothèque s'avère très coûteuse en moyens humains. Un maillage bien adapté est une condition indispensable à l'obtention de bons résultats.

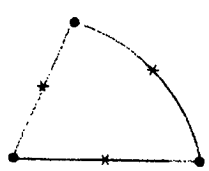
La technique de maillage adoptée dans MODULEF est une technique arborescente : le domaine est décomposé en parties géométriquement simples. Ces parties sont ensuite manipulées à l'aide d'opérateurs simples (symétrie, rotation, translation ...) et ensuite recollées jusqu'à obtention du maillage final.

La technique de maillage ainsi que les différents modules disponibles sont décrits dans le manuel APNOPO doc 104.

Le résultat de cette étape est une **S.D. NOPO.** Cette S.D. contient la description du maillage : les noeuds (sommets ou non) ont été générés ainsi que les coordonnées des sommets. L'interpolation n'a pas encore été prise en compte ; ce sera l'objet de l'étape suivante :

EXEMPLE :

Si l'élément **triangle P2 courbe** (cf exemple 2 précédent) est utilisé après cette étape



Les noeuds 1 à 6 sont générés ainsi que les coordonnées des sommets (maillage). Dans l'étape suivante les coordonnées des noeuds milieux seront calculées (choix de l'élément fini).

Le tracé du maillage est réalisé à l'aide du module TRNOPO (cf brochure 98).

II. Choix des éléments finis

Cette étape permet de prendre en compte les interpolations des fonctions. Elle peut être réalisée par le module COMACO brochure 13. Deux cas peuvent se présenter :

i) l'élément fini choisi existe dans l'une des bibliothèques :
THER (les éléments de thermique sont décrits dans la brochure 100)
ELAS (les éléments d'élasticité linéaire sont décrits dans la brochure 101)
ELNL (les éléments d'élasticité non linéaire sont décrits dans la brochure 97).

ii) l'élément choisi n'existe pas . Alors :
. soit il est à intégrer dans l'une des bibliothèques THER, ELAS, ELNL, FLUI. Après cette opération, décrite dans le brochure 95, on retombe sur le cas i).
. soit il peut être décrit directement dans COMACO à l'aide de cartes de données.

Le résultat de cette étape est constitué des deux **S.D. MAIL et COOR.** La S.D. MAIL contient la description du maillage (noeuds, points ...) et de l'interpolation (nombre et nom des inconnues variationnelles ; des mnémoniques etc...). La S.D. COOR contient les coordonnées des points (sommets ou non). Attention la S.D. COOR en sortie de COMACO ne contient pas les coordonnées des noeuds. Reprenons l'exemple des éléments "triangle P2 droit" et "triangle P2 courbe" : dans le cas du premier, la S.D. COOR contient les coordonnées des sommets dans le cas du deuxième des sommets et milieux de côtés.

III) Données physiques, création des tableaux élémentaires

Le but de cette étape est de créer les tableaux élémentaires $[M_T]$, $[K_T]$, $[b_T]$. Ce calcul nécessite la connaissance des données physiques qui décrivent le matériau, les efforts, les températures etc.... Dans le cas de l'exemple choisi il s'agit de **f** quantité qui intervient au second membre et **a** au premier membre. Toutes ces données physiques doivent être connues au moment où le module de création des tableaux élémentaires est activé.

Pour l'exemple traité il y a deux possibilités :

i) les **données** sont des constantes. Dans ce cas le module THERCT (brochure 56) peut être utilisé.

ii) les **données** ne sont pas constantes. Dans ce cas elles peuvent être fournies par sous-programme ou tableau. La description de la façon dont seront fournies les données est indiquée en utilisant les modules COMILI, COFORC. Les tableaux élémentaires seront ensuite construits par le module THELAS. La description de ces modules est contenue dans la brochure 14.

Dans tous les cas, il est possible de fournir ces caractéristiques par sous-domaine et/ou numéro de référence ce qui minimise le nombre des données. Le résultat de cette étape est une **S.D. TAE** qui contient, pour tous les éléments du maillage les tableaux élémentaires associés.

REMARQUES :

i) Cette étape varie en fonction de la nature du problème à traiter. Les modules THELAS, THERCT, ELASCT réalisent cette étape dans le cas de problèmes de thermique et élasticité linéaire. Ces mêmes modules peuvent être adaptés pour la résolution d'autres problèmes.

D'autres modules réalisent cette étape pour des problèmes spécifiques (ex COTAE en élasticité non linéaire en grandes déformations (97) ; BIHAP1, BIHAP2 pour un problème biharmonique (93) ...).

ii) Pour traiter un problème nouveau il suffit souvent de construire la S.D. TAE à partir des S.D. MAIL, COOR et des caractéristiques des matériaux.

IV) **Résolution du problème linéaire**

Cette étape se décompose en plusieurs sous-étapes. Les sous-étapes varient en fonction de la méthode de résolution choisie mais les principes de base restent les mêmes. La bibliothèque MODULEF contient un grand nombre de méthodes de résolution de systèmes linéaires.

i) des méthodes directes (cf brochures 21, 24, 62) avec la matrice résidant en MC ou en mémoire secondaire.

ii) des méthodes itératives (cf brochures 26 et 102).

Dans tous les cas la matrice associée à une discrétisation par éléments finis est une matrice creuse. Pour minimiser la place mémoire occupée un stockage approprié de la matrice est utilisé.

. Pour les méthodes directes (sauf hypermatrices brochure 62) un **stockage profil** est utilisé (cf description de la S.D. MUA dans brochure 2).

. Pour les méthodes itératives un **stockage morse** (cf description de la S.D. AMAT brochure 2) où seuls les coefficients non nuls sont stockés).

Le problème de **choix** de la méthode de résolution du système linéaire est non trivial et dépend fortement de la nature du problème à résoudre. Dans le cas de systèmes simples **toutes** les méthodes donnent de bons résultats. Quand il y a beaucoup de systèmes à résoudre (par exemple pour un processus itératif) il y a des arguments en faveur des deux types de méthodes (directes ou itératives)

i) pour une **méthode directe** si la matrice est constante elle est **factorisée** une fois pour toutes ensuite une résolution se limite à une **descente-remontée**

ii) pour une **méthode itérative** une bonne approximation de la solution est connue (à l'itération précédente ou au pas de temps précédent ...) donc le nombre d'itérations nécessaire est **réduit.**

Prenons l'exemple simple de la résolution du système linéaire par une méthode directe de CHOLESKY en MEMOIRE CENTRALE pour analyser les différentes sous-étapes.

IV.1. Construction des pointeurs

Cette étape construit, en explorant tous les éléments du maillage, les pointeurs nécessaires au stockage profil et calcule la taille mémoire occupée par la matrice. Ainsi le calcul sera lancé uniquement si la place mémoire disponible est suffisante.

Cette étape est réalisée par le module PREPAC (brochure 21). Le résultat est une **S.D. MUA** qui ne contient pas le dernier tableau (la matrice) mais uniquement les pointeurs.

IV.2 Assemblage

En partant de la S.D. MUA précédemment créée et de la S.D. TAE qui contient les tableaux élémentaires les modules d'assemblage ASMAPC (pour les matrices), ASMBMC (pour les seconds membres) réalisent les opérations (cf (4), (4'))

$$\begin{cases} \left[\tilde{A}\right] = \sum_{T \in \mathcal{T}_h} {}^t\left[B_T\right] \left[A_T\right] \left[B_T\right] \\ \left[\tilde{B}\right] = \sum_{T \in \mathcal{T}_h} {}^t\left[B_T\right] \left[b_T\right] \end{cases}$$

Le résultat de cette étape est la structure de données **MUA** (qui contient la matrice du système), et la structure de données **B** (qui contient les seconds membres).

IV.3 Prise en compte des conditions aux limites bloquées

Deux types de conditions aux limites peuvent apparaître :

i) celles qui sont prises en compte dans la formulation variationnelle, par exemple des conditions Neumann ou Fourier dans un problème de thermique, formulation classique (cf brochure 100 : introduction).

ii) celles qui ne sont prises en compte dans la formulation variationnelle, par exemple les conditions de Dirichlet.

Le but de cette étape, qui se décompose en 2 sous-étapes, est de prendre en compte ce dernier type de conditions aux limites.

IV.3.1. Construction de la S.D. BDCL

La S.D. BDCL contient une description des conditions aux limites bloquées (numéros des degrés, valeur). Elle est construite par un des modules COBDC2 (brochure 57) ou COBDCL (brochure 18).

Le module COBDCL construit la S.D. BDCL d.l. par d.l. à partir de cartes de données.

Le module COBDC2 construit cette même S.D. en utilisant des notions globales : les numéros de référence ; les noms des inconnues variationnelles. Il est plus simple d'emploi que COBDCL mais moins général, en particulier il ne traite pas le cas des conditions aux limites en relation linéaire. Ce cas, traité par COBDCL est particulièrement utile pour les conditions périodiques.

REMARQUES :

i) Si le problème traité contient des conditions aux limites en relation linéaire, cette sous-étape doit être réalisée **avant** l'étape IV.1 car le profil de la matrice est modifié.

ii) Le module COBDC2 utilise une S.D. COOR qui contient les coordonnées des noeuds. Elle peut être obtenue par le module CORNOE (cf brochure 63).

□

VI.3.2. **Prise en compte effective des conditions aux limites**

Il s'agit, avec les notations (6'), (7) de passer de $\left[\tilde{A}\right]$ à $\left[A_1\right]$

Il y a plusieurs techniques de prise en compte des c.l. Elles sont décrites (pour les matrices profil) dans la brochure (21) : module CLIMPC. Conceptuellement nous pouvons distinguer 2 classes :

i) une méthode de pénalisation. Son point de départ est l'égalité "informatique" suivante :

VTG + a = VTG si VTG >> a (par ex VTG=10^{20} et a = 1)

De façon pratique pour cette méthode on part de $\left[\tilde{A}\right]$ et $\left[\tilde{B}\right]$ de (6') et on remplace les **coefficients diagonaux** de A_{22} par VTG et les coefficients de B_2 par VTG.U_0 où

(9) $U_0 = \begin{bmatrix} u_1 \\ \\ u_{Nh} \end{bmatrix}$ vecteur des valeurs de blocage (dans notre ex. 0 cf (7))

Une "variante" consiste à remplacer les coefficients diagonaux de A_{22} par

$a_{ii} = a_{ii} * VTG$ au lieu de $a_{ii} = VTG$

et ceux du second membre B_2

$b_i = U_i * a_{ii} * VTG$ au lieu de $b_i = u_i * VTG$.

ou $i \quad N_h$

Dans les deux cas le résultat est le même.

Avantages et inconvénients

. Dans la première variante, la **valeur** initiale du coefficient diagonal est perdue.

. Dans la seconde variante, il faut connaître la valeur du coefficient diagonal pour la prise en compte des conditions aux limites sur le second membre. Il faut donc faire les deux prises en compte des c.l. simultanément ce qui peut être un inconvénient pour une méthode itérative où la matrice est constante.

En résumé : <u>la variante 1</u> correspond à une prise en compte séparée des c.l. sur les matrices et seconds membres avec (brochure 21 PROFIL/CLIMPC: 2,3).

. matrices NIVO = 4
VTG grand

. second membre NIVO = 1
VTG = au précédent

<u>la variante 2</u> conditions aux limites sur MUA et B
NIVO = 2
VTG = grand.

ii) <u>la méthode directe</u>

L'idée est de passer, réellement de (6') à (7). Remarquons que dans (7) une propriété importante de la matrice est perdue : **la symétrie.** C'est d'autant plus important que, pour une matrice symétrique seule une demi-matrice est stockée.

Il faut donc passer de (6') à (7') équivalent de (7) qui conserve la symétrie.

Dans le cas général (d.l. bloqués à une valeur U_0 (9) nulle ou non) si on considère une partition de $\{\tilde{u}\}$

$\{\tilde{u}\} = [U_1, U_0]$ où U_1 est le vecteur des d.l. libres
U_0 est le vecteur des d.l. bloqués

(7) s'écrit :

$$(7) \qquad \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ 0 & I \end{bmatrix} \begin{bmatrix} U_1 \\ U_0 \end{bmatrix} = \begin{bmatrix} B_1 \\ U_0 \end{bmatrix}$$

ce qui est, évidemment équivalent à :

$$(7') \qquad \begin{bmatrix} \tilde{A}_{11} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} U_1 \\ U_0 \end{bmatrix} = \begin{bmatrix} B_1 - \tilde{A}_{12} \cdot U_0 \\ U_0 \end{bmatrix}$$

Dans la pratique ce qui est calculé c'est (7'). C'est, avec les notations de la brochure 21 PROFIL/CLIMPC:2,3 l'option

NIVO = 3, VTG = 1.

REMARQUE :

Les différentes techniques ci-dessus sont indépendantes du mode de stockage de la matrice et seront donc retrouvées pour les autres types de stockage (ex : module CLIMGC pour les matrices morses).

IV.4. Factorisation de Cholesky

Cette étape permet de calculer la matrice L triangulaire qui vérifie

$$A = {}^tL\, L$$

et est réalisée par le module CHOLPC (brochure 21).

Le résultat est une **S.D. MUA** qui contient la matrice factorisée.

IV.5. Descente-remontée

La descente-remontée effectue la résolution de deux systèmes linéaires de matrice triangulaire. Elle est réalisée par le module DRCHPC (brochure 21).

Le résultat est une structure de donnée **B** qui contient le résultat.

COMMENTAIRES

La résolution du système linéaire est décomposée en plusieurs étapes pour rendre l'utilisation la plus simple possible. En effet :

- si plusieurs systèmes qui ont la même matrice sont résolus la factorisation est faite une seule fois.
- pour assembler une matrice de masse et une matrice de rigidité la préparation (calcul des pointeurs) est faite une seule fois.

etc...

v) **Interprétation des résultats**

L'étape de post-traitement est particulièrement importante. Il est en effet difficile d'interpréter les résultats à partir de l'image de la S.D. B. Les outils essentiels dont on dispose sont :

- calcul des normes d'erreur et présentation de la solution exacte et calculée (quand la solution analytique du problème est connue) : module NORME cf brochure 81
- tracé des isovaleurs (dans le cas de l'exemple choisi) ou des contraintes, vitesses, déformées ... cf brochure 96.

Dans le cadre du post-traitement peuvent être utlisés aussi les modules de modification d'une S.D. NOPO (cf brochure 99), et d'une S.D. B. En effet, si le problème présente des symétries le calcul peut être fait uniquement sur une partie du domaine. Il est néanmoins agréable de pouvoir présenter les résultats sur l'ensemble du domaine.

5. CONCLUSIONS

Le but de cette annexe est de montrer comment trouver dans les bibliothèques les modules capables de résoudre un problème donné, mais il ne s'agit en aucun cas de montrer comment utiliser un module donné. Pour cette dernière démarche nous renvoyons aux brochures correspondantes. Des exemples d'enchainements sont fournis dans la bibliothèque TEST et commentés dans la brochure de tests (87).

EXEMPLE DE MODULE

Cet exemple traite le Module PIGRA, module developpé a l'universite de Pavie, dont le but est de régulariser un maillage.

De haut en bas, on remarque :

- l'instruction SUBROUTINE (paramètres)
- le but du Module
- les paramètres d'entrée et de sortie (le cas échéant)
- le nom du ou des programmeurs, la date
- les déclarations : type, dimension, external

  On remarque en particulier :
  - le tableau NZNOPO
  - le COMMON ALNOPO (page 23)
  - l'équivalence entre ce COMMON et le tableau NZNOPO
- les formats : impression d'entrée (FORMAT 100) si IMPRE non nul
  de sortie (FORMAT 101) si IMPRE non nul
  impressions particulières pour IMPRE plus grand
- la restauration en M.C de la S.D.E NOPO à partir du fichier de numero NFNOPO (page 7 pour NFNOPO, 31 pour RSTSDE)
- préparation de la sauvegarde de la S.D.S NOPO (page 29 pour INICSD, page 34 pour COPISD)
- initialisation des valeurs nécessaires à partir des tableaux de la S.D.E maintenant actifs
- adressage des tableaux (pages 12, 14, 4)
- création d'un tableau associé à la S.D NOPO (page 20) et description de ce tableau dans le tableau NOP1 (page 21)
- appel du ou des algorithmes (Les tableaux sont passés par leur première adresse dans le super tableau (pages 16-17, 40, 48-49)).Chaque opération est commentée
- destruction des tableaux devenus inutiles (page 14)
- sauvegarde de la S.D.S par SAUSDS (page 33)
- impression de fin de module

```
      SUBROUTINE PIGRA(M,NFNOPE,NINOPE,NFNOPS,NINOPS)
C     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     BUT :  MODIFIER UNE S.D.E. NOPO POUR CREER UNE TRIANGULATION
C     ---    DU TYPE AIGU.
C            POUR UNE DESCRIPTION DETAILLEE CF :
C            " L.D.MARINI-P.PIETRA: PIGRA, A PROGRAM FOR GENERATING A
C            MESH OF THE ACUTE TYPE. (IAN-CNR, PAVIA)"
C     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     PARAMETRES D ENTREE :
C     ---------------------
C     M       : SUPER TABLEAU DE TRAVAIL
C     NFNOPE  : NUMERO DU FICHIER DE LA S.D.E. NOPO
C     NINOPE  : NIVEAU DE LA S.D.E. NOPO
C     NFNOPS  : NUMERO DU FICHIER DE LA S.D.S. NOPO
C     NINOPS  : NIVEAU DE LA S.D.S. NOPO
C     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     PROGRAMMEURS :   L.D.MARINI - P.PIETRA IAN-CNR, PAVIA DECEMBRE 1983
C     ......................................................................
      CHARACTER*4 CHAR4,NOM
      CHARACTER*72 ICOMM,KINFO*80
      DIMENSION NZNOPO(19),M(*)
      EXTERNAL ECNOPO,LENOPO,RENOPO
      COMMON/ALNOPO/NENOPO,NOPO,IANOPO,LENOPO,NOP1,IANOP1,LENOP1,
     &                    NOP2,IANOP2,LENOP2,NOP3,IANOP3,LENOP3,
     &                    NOP4,IANOP4,LENOP4,NOP5,IANOP5,LENOP5
      EQUIVALENCE(NENOPO,NZNOPO(1))
      DATA ICOMM/'1ERECOMP.=NBRE MAX DE VOISINS.I-EME COMP.=NO. DE REFER
     &ENCE DU SOMMET I-1'/
C
  100 FORMAT(1X,78('&')/' MODULE PIGRA : ',A/1X,78('&'))
  101 FORMAT(1X,78('&')/' FIN DU MODULE PIGRA '/1X,78('&'))
  102 FORMAT(/' RESULTATS SUR LE FICHIER ',I6/)
C
      IMPRE = IINFO('BAVARD')
      IF ( IMPRE .NE. 0 ) WRITE(IINFO('I'),100) KINFO('TITRE')
C
C     RESTAURER EN MC LE SD NOPO
C     --------------------------
      CALL RSTSDE(M,'NOPO',NFNOPE,NINOPE,NZNOPO,19,LENOPO,RENOPO,
     &            NCNOPE,ICNOPE)
C
C     POUR LA SAUVEGARDE
C     ------------------
      IF (NINOPE .NE. NINOPS) THEN
         CALL COPISD(M,'NOPO',6,NINOPE,NCNOPE,ICNOPE,NINOPS,NCNOPS,
     &               ICNOPS)
         CALL INICSD(M,'NOPO',NINOPS,NZNOPO,19,NCNOPS,ICNOPS)
      ELSE
         NCNOPS = NCNOPE
         ICNOPS = ICNOPE
      END IF
C
      NE    = M(IANOP2+4)
      NOE   = M(IANOP2+14)
      NP    = M(IANOP2+21)
      NCOPNP = M(IANOP2+3)
C
      IANF  = 0
      LNF   = NP + 1
      NTYP  = 1
      NOM   = 'IFRO'
      CALL READRE(NTYP,'IFRO',IANF,LNF,M,NN)
C
C     ASSOCIER A NOPO UN TABLEAUX SI NOEUDS ET POINTS COINCIDENT
C     ----------------------------------------------------------
      IF ( NCOPNP .EQ. 1 ) THEN
```

```
      NT     = M(IANOP0+31)
      NTAC   = NT+1
      LE     = LENOP1
      LENOP1 = LENOP1+22
      NTY    = 1
      IAA    = IANOP1
      IANOP1 = 0
      CALL READRE(NTY,CHAR4(NOP1),IANOP1,LENOP1,M,NN)
      CALL TRTATA(M(IAA),M(IANOP1),LE)
      L1       = IANOP1+22*NT-1
      M(L1+1) = ICHAR4(NOM)
      M(L1+2) = IANF
      M(L1+3) = LNF
      M(L1+4) = NTYP
      CALL TRCHTA(ICOMM,M(L1+5),18)
      M(IANOP0+30) = M(IANOP0+30)+1
      M(IANOP0+31) = NTAC
    END IF
C
      IANEVE = 0
      LNEVE  = NP+1
      CALL READRE(NTY,':NEV',IANEVE,LNEVE,M,NN)
C
C   CONSTRUCTION DU TABLEAU NEVE DE NP+1 VALEURS
C   NEVE(1) = 0 , NEVE(I+1)-NEVE(I) = NOMBRE D ELEMENTS CONTENANT LE
C   SOMMET I ----------------------------------------------------
      CALL TAV1(NE,NP,NCOPNP,M(IANOP5),LNEC,M(IANEVE))
      IANEC  = 0
      CALL READRE(NTY,':NEC',IANEC,LNEC,M,NN)
      LNVOI  = LNEC+NP
      IAVOI  = 0
      CALL READRE(NTY,':NVO',IAVOI,LNVOI,M,NN)
C
C   CONSTRUCTION DU TABLEAU NVOI CONTENANT LES SOMMETS VOISINS D-UN
C   SOMMET
C   MODIFICATION DE NEVE DEVIENT NEVE(1) = 0  NEVE(I+1)-NEVE(I)
C   NOMBRE DE SOMMETS VOISINS DU SOMMET I
C   ----------------------------------------------------------
      CALL TAV2(NE,NP,NCOPNP,LNEC,M(IANOP5),M(IANEVE),M(IANEC),M(IAVOI))
      CALL TUER(':NEC',M)
C
C   CONSTRUCTION DU TABLEAU IFRO DE NP+1 VARIABLES
C   IFRO(1) = LARGUEUR DE BANDE IFRO(I+1) = NO DE REFERENCE DU POINT I
C   ----------------------------------------------------------------
      CALL APFRO(NP,NE,M(IANOP5),NCOPNP,M(IANF),NP1,M(IANEVE),LDEP)
      IAIND  = 0
      CALL READRE(NTY,':IND',IAIND,NP1,M,NN)
      IADEP  = 0
      CALL READRE(NTY,':DEP',IADEP,LDEP,M,NN)
C
C   CONSTRUCTION DES TABLEAUX IND ET NDEP. IND EST L'EQUIVALENT DE NVOI
C   POUR LES POINTS INTERNES; IND EST SON POINTEUR
C   ----------------------------------------------------------------
      CALL COMPAT(NP,NP1,LDEP,M(IANEVE),M(IAIND),M(IADEP),M(IAVOI),
     &            M(IANF+1))
      CALL TUER(':NVO',M)
C      ------------------------------------------
C      ------      APPEL DE L'ALGORITHME      ------
C      ------------------------------------------
      CALL PIGRA1(M(IANOP4),M(IAIND),M(IANEVE),M(IADEP),NP1,NE,
     &            M(IANOP5),NP,NCOPNP)
C
C     SAUVEGARDE
C     ----------
      NOPT   = 0
      CALL SAUSDS(M,'NOPO',NFNOPS,NINOPS,NZNOPO,19,NCNOPS,
```

```
&             ICNOPS,ECNOPO,NOPT,NMOT)
 IF ( IMPRE .GT. 2 ) CALL IMNOPO(M,0,NINOPS,IMPRE)
 CALL TUER(':IND',M)
 CALL TUER(':DEP',M)
 CALL TUER(':NEV',M)
 IF(NCOPNP .NE. 1) CALL TUER(NOM,1)
 IF ( IMPRE .GT. 2 ) WRITE(IINFO('I'),102) NFNOPS
 IF ( IMPRE .NE. 0 ) WRITE(IINFO('I'),101)
 END
```

LISTE DES PUBLICATIONS MODULEF

(1) PL. GEORGE. M. VIDRASCU,
Guide d'utilisation et Normes de programmation. Mars 1984.

(2) A.PERRONNET,
Description des structures de données du Club Modulef. Février 1979.

(12) A. MARROCCO,
GEL3D1.Module de maillage tridimensionnel automatique. Décembre 1978.

(13) D. LEROY,
Description du maillage interpolation, Module COMACO. Mai 1979.

(14) M. VIDRASCU,
Modules de création des tableaux élémentaires associés aux éléments.
Novembre 1978.

(18) F. BAY, A. PERRONNET, M. VIDRASCU,
Construction de la structure de données BDCL décrivant les conditions aux limites "FORCEES". Modules COBDCL, COBDC2, COBDC3. Octobre 1984.

(20) A. PERRONNET,
Le module CONDL1. Janvier 1980.

(21) A. PERRONNET,
Quelques résolutions directes des systèmes linéaires de matrice profil.
Octobre 1984.

(22) PL. GEORGE, A. PERRONNET, M. VIDRASCU,
L'assemblage et la manipulation d'une S.D. B. Octobre 1984.

(24) P. JOLY,
Méthode frontale. Juin 1978.

(26) A. PERRONNET, X. DESROCHES,
Quelques résolutions itératives des systèmes linéaires par gradient conjugué avec préconditionnement ou non. Octobre 1984.

(30) A. PERRONNET,
Manuel d'utilisation des modules de calcul des valeurs et vecteurs propres $\{x\}$ du problème $([K] - \lambda [M]) \; x = 0$.
Juillet 1979.

(31) A. PERRONNET.
Méthodes de calcul de valeurs λ et vecteurs propres x du problème :
$([K] - \lambda [M]) \; x = 0$. Application à l'intégration des problèmes transitoires.

(36) F. THOMASSET, P. CAUSSIGNAC ,
Equations de Navier Stokes bidimensionnelles : modules NSNCEV,NSNCST,TRSD.

(38) H. FOERSTER.
TR2D01, TR2D02 to solve the Dirichlet problem for Helmholtz' equation,
G.M.D. no 47, BONN.

(39) H. FOERSTER,
TR2D04, TR2D05 to solve the Neumann and periodic boundary value problems for Helmholtz' equation, G.M.D. no 53, BONN.

(40) H. FOERSTER, H. REUTERSBERG,
TR2D11 to solve the modified biharmonic boundary value problem, G.M.D. no 50, BONN.

(41) C. CHINOSI, G. FUMAGALLI, L.D. MARINI, A. QUARTERONI, G. SACCHI
User manual for the programs HYBREF, HYBREC, DEPER, DEPLAR, CLOTO, DEVES, PAVIA 1978.

(42) P. LAUG,
Gestion des fichiers en accès direct, MEFDIR, Manuel d'utilisation. Juin 1977.

(44) P. LAUG,
Lecture de cartes-données en format libre : manuel d'utilisation. Janvier 1979.

(46) Y. DEPEURSINGE,
Gestion de la mémoire secondaire selon un algorithme de mémoire virtuelle dynamique : le système MEMVIR, Ecole Polytechnique de Lausanne, Suisse. Janvier 1980.

(47) Manuel d'utilisation du logiciel graphique Fortan 3D, Centre de CAO-ENSTA-INRIA.

(49) F. FORGES,
Notice d'utilisation de TRI-SUP, Septembre 1978.

(52) P.L. GEORGE,
Résolution numérique de problèmes évolutifs, Thèse de 3ème cycle, Université PARIS VI, 20 Juin 1980.

(54) A. TAL-NIR, M. BERCOVIER,
Implémentation of the Multigrid method "MLAT" in the finite element method. The Hebrew University Jerusalem Report 80/101.

(56) M. VIDRASCU,
Création de la S.D. TAE, Modules THERCT ELASCT, Mars 1981.

(58) M. VIDRASCU,
Construction de la structure de données COOR des coordonnées des noeuds, Module CORNOE, Avril 1981.

(59) D. MARINI,
Description de quelques éléments de plaques, Intégration dans Modulef, Rapport de Recherche INRIA no 59, Mars 1981.

(62) M. VAN-INGELANDT,
Structure de données HYMA et modules associés, Université de Lille. Octobre 1980.

(63) P.L. GEORGE,
Problèmes évolutifs, Juin 1981.

(64) Utilisation des algorithmes de résolution, leur fonction, leurs paramètres, leurs commentaires, Mai 1981.

(65) Mémento des modules : de leurs bibliothèques, de leur fonction, de leurs paramètres, de leurs commentaires, Mai 1981.

(66) Utilisation de la gestion dynamique des tableaux dans Modulef, Mai 1981.

(67) Mémento des procédures, Juillet 1981.

(70) M. VAN INGELANDT,
Module SSPAHM, calcul de valeurs propres avec S.D. HYMA, Université de LILLE. Mai 1981.

(72) A. FUSCIARDI, P. MARCHIORO, A. MORGANA,
The solution of complementary Systems, module COMPL - COMPRX, Décembre 1980.

(73) The Club Modulef : A library of computer procedures for finite element analysis. Avril 1982.

(74) J. ROMAN.
Trois modules de traitement des structures de données AMAT et B .
Résolution de systèmes linéaires de matrice symétrique définie positive par méthode directe - Modules LOFACT, NUFACT, DESREM, Université de Bordeaux 1, Octobre 1980.

(76) A. BERMUDEZ, J.M. VIANO,
Quelques méthodes numériques en élastoplasticité. Mai 1981.

(77) F. PISTRE, R. PIERROT, J. VAZEILLES,
COLIBRI, Modules COLIBR, COLIB2, Mars 1982, remplace la publication no 9.

(78) M. VAN INGELANDT,
Calcul des valeurs et vecteurs propres par une méthode de LANCZOS itérative, Module LANCMF, mai 1982, Université de LILLE.

(79) M. SERMANGE,
Physique des plasmas : modules de calcul d'équilibres MHD axisymétriques, Février 1982.

(82) J. ROMAN,
Renumérotation des noeuds d'interpolation d'un maillage plan d'éléments finis à l'aide du théorème de séparation de LIPTON et TARJAN, Octobre 1980.

(83) P. LAUG,
Implementation de la bande Modulef. Mai 1982.

(84) F. PISTRE,
Maillage tridimensionnel à partir d'un maillage bidimensionnel Module MA2D3D, Mai 1982.

(85) Présentation du Club Modulef. Mai 1984

(86) A. MARROCCO,
Bibliothèque de tracés,TRACOU, TRIANG, EQUI, TRINUM, KOUPE, COUPE, LIGNES, TRF. Octobre 1979.

(87) D.STEER, M. VIDRASCU,
Quelques exemples de tests, test de thermique T1, test d'élasticité E1, test en mécanique des fluides F1. Mai 1982.

(88) P. CAUSSIGNAC, Y.DEPEURSINGE, Y.JACCARD,
Manuel d'utilisation du module NSPOAX de résolution des équations de NAVIER-STOKES stationnaires axisymétriques, mai 1982, EPFL - Lausanne.

(89) J.M. VIANO,
Calcul des contraintes et visualisation des zones plastiques en élastoplacité bi et tridimensionnelle, Modules ELAPLA, VISPLA, Avril 1982.

(90) M. VIDRASCU,
Description du calcul des contraintes par élément, Module STRESS.
Avril 1982, remplace le no 28.

(92) L. DELLA CROCE, G. SACCHI,
Résolution de systemes de complémentarité : Les modules AMATMS et RELAX. mai 1982, Pavia.

(93) M. VIDRASCU,
Résolution du problème de Dirichlet pour l'opérateur biharmonique par une méthode d'éléments finis mixtes, modules : BIHAP1, BIHAP2, DEDIRI, DEREFR, FACMAF,GRADCO, PRCOL. Mars 1983.

(94) P. LAUG.
Les fonctions interprétées. Rapport Technique INRIA no 38, Juin 1984.

(95) P.L. GEORGE, A. PERRONNET,M. VIDRASCU,
Intégration d'un nouvel élément fini, création d'une bibliothèque d'éléments finis. Avril 1983.

(96) P.L. GEORGE, F. PISTRE,
Postraitement. Tracé des isovaleurs, des contraintes, des vitesses, des coupes. Modules ISOSOL, TRAISO. TABSTR, TRASTR, TRAVIT, COUPEB, TRCOUP, Juin 1983.

(97) P. LE TALLEC. M. VIDRASCU,
Plasticité non linéaire : quelques modules de calcul en grandes déformations, Modules COBDNO, COTAE, GDEAXI,GDEFIN,GDEPLA,GRAFEL, Super-modules PRELA2,PRELA3. Mars 1983.(remplace le no 75)

(98) F. PISTRE,
Tracé d'une S.D. NOPO, Module TRNOPO, Juin 1983.

(99) P.L. GEORGE. P. LAUG, F. PISTRE,
Construction et modification de la Structure de Données NOPO. Mai 1984.

(100) Fiches Techniques "Eléments finis", Bibliothèque Thermique. Avril 1983. (remplace le no 61).

(101) Fiches Techniques "Eléments Finis", Bibliothèque d'Elasticité. Avril 1983. (remplace le no 61).

(102) P. JOLY,
"Méthodes de gradient conjugué pour résoudre des systèmes linéaires non symétriques. Les modules RESMIN, BIGRAD et TCHEBY".Juin 1983. (remplace le no 55).

(103) Y. DEPEURSINGE, S. JEANDREVIN, Ph. CAUSSIGNAC,
"GRELFI : Editeur de type éléments finis". Mai 1983.

(104) P.L. GEORGE,
Le module APNOPO. mailleur bidimensionnel du Club Modulef. Mai 1984.

(106) C. RODRIGUEZ,
Torsion elastoplastique et ecoulement d'un fluide de Bingham. avril 1984

(107) C. CONCA, D. STEER,
"Résolution des équations bidimensionnelles de Navier Stokes pour un fluide incompressible et visqueux en régime stationnaire, Modules NSKINC et COTABM". Juin 1983.

(108) P.L. GEORGE,
Utilisation conversationnelle de Modulef . Programmes principaux . Programmes conversationnels. Mai 1984

(109) P. LAUG,
Conversion de Modulef en FORTRAN 77. Rapport Technique INRIA no 34 fevrier 1984

(110) L.D. MARINI P. PIETRA
Solution of filtration problems through porous media . The programs
PIGNA DAMIAN, PLOTDAM. IAN-CNR PAVIA. May 1984

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique