CENTRE NATIONAL D'ETUDES SPATIALES

MSLIB Fortran 90

CS

Nomenclature : **M-MU-0-103-CIS**

Edition : 03 Date: 25/10/2000
Révision: 08 Date: 15/02/2005

Volume 3

Caractéristiques principales et conventions d'utilisation de la MSLIB Fortran 90

Rédigé par : Guylaine PRAT	le : CS (SI/Espace/FDS)	
Validé par : Guylaine PRAT Anne MAZZIETTI-ERSA (ingénieur qualité)	le : CS (SI/Espace/FDS) CS (SI/Espace)	
Pour application : Franck REINQUIN Hervé MADIEU	le : CNES (DCT/SB/OI)	

18 avenue Edouard Belin
31401 TOULOUSE CEDEX 9
Tél: 05 61 27 31 31

DIFFUSION INTERNE CNES

Observations

Voir la note nomenclaturée M-NT-0-18-CN:
"*Liste de diffusion de la documentation utilisateur MSLIB*".

DIFFUSION EXTERNE CNES

Observations

Voir la note nomenclaturée M-NT-0-18-CN:
"*Liste de diffusion de la documentation utilisateur MSLIB*".

BORDEREAU D'INDEXATION

CONFIDENTIALITE : NC

MOTS-CLES :
MSLIB, Fortran 90, caractéristiques principales, conventions d'utilisation.

TITRE : Volume 3 - Caractéristiques principales et conventions d'utilisation de la MSLIB Fortran 90

AUTEUR : Guylaine PRAT

RESUME :
Ce document présente les caractéristiques principales de la bibliothèque de mécanique spatiale MSLIB Fortran 90, ainsi que les conventions d'utilisation de la bibliothèque.
En outre, ce document explique comment sont renseignées les notices d'utilisation des routines et comment elles doivent être interprétées.

SITUATION DU DOCUMENT : Création

VOLUME :	PAGES : 59	PLANCHES :	FIGURES :	LANGUES : F

CONTRAT : Marché 779/Cnes/2001/8929 BC4500009860

SYSTEME HOTE : Frame6/MSLIB

MODIFICATION

ETAT DOCUMENT				PAGES REVISEES	
ED.	REV.	DATE	REFERENCE ORIGINE (pour chaque édition)	ETAT PAGE *	NUMERO DES PAGES
01	00	11/08/98	M-MU-0-103-CN Rédacteur : E. Le Dé avec la participation de G. Prat		Création
02	00	16/09/99	M-MU-0-103-CN Rédacteur : E. Le Dé avec la participation de S. Vresk	M	Voir barres de modifications
03	00	25/10/00	M-MU-0-103-CN Rédacteur : E. Le Dé avec la participation de V. Lépine et G. Prat	M	Ajout d'un nouveau type dérivé *tm_quat* (§ 2.3.2). Ajout de nouveaux vocabulaires (annexe 1) et de codes retour (annexe 2)
03	01	17/04/01	M-MU-0-103-CIS Rédacteur : G. Prat avec la participation de E. Le Dé	M	Ajout de nouveaux champs pour le type dérivé *tm_code_retour* (§ 2.3.2 et § 2.4)
03	02	10/04/02	M-MU-0-103-CIS Rédacteur : M.Hazak avec la participation de E. Le Dé et G.Prat	M	Création du parameter *pm_version_MSLIB90* (§ 2.2.4); ajout attribut sequence (voir § 2.3.1); ajout plage de codes retour (annexe 2)
03	03	03/03/03	M-MU-0-103-CIS Rédacteur : G. Prat avec la	M	Ajout d'un nouveau code retour (annexe 2)
03	04	30/09/03	M-MU-0-103-CIS Rédacteur : B. Revelin et G. Prat	M	§ 2. (référence volume 5) Ajout de codes retour (annexe 2)
03	05	05/12/03	M-MU-0-103-CIS Rédacteur : B. Revelin et G. Prat	M	Création de parameters liés aux planètes (§ 2.2.2) Ajout de codes retour (annexe 2)
03	06	27/05/04	M-MU-0-103-CIS Rédacteur : V. Lépine et G. Prat	M	Ajout de nouveaux vocabulaires (annexe 1) et de codes retour (annexe 2)
03	07	24/09/04	M-MU-0-103-CIS Rédacteur : G. Prat	M	Ajout de nouveaux vocabulaires (annexe 1)
03	08	15/02/05	M-MU-0-103-CIS Rédacteur : G. Prat	M	Ajout de nouveaux vocabulaires (annexe 1) et de codes retour (annexe 2)

* I = Inséré S = Supprimé M = Modifié

Sommaire

Index

P

Documents de référence

DR1

Règles de programmation et de conception pour la MSLIB Fortran 90
M-ST-0-44-CN
Edition 2 Révision 0
E. Le Dé, G. Prat, J.C. Bergès

DR2

Volume 1
Contenu de la documentation utilisateur de la MSLIB Fortran 90
M-MU-0-101-CIS
G. Prat

DR3

Report of the IAU / IAG working group on cartographic coordinates and rotational elements of the planets and satellites: 2000
P.K. Seidelmann, V.K. Abalakin, M. Bursa, M.E. Davies, C. De Bergh, J.H. Lieske, J.Oberst, J.L. Simon, E.M. Standish, P. Stooke, P.C. Thomas

Documents applicables

DA1

Spécification de la gestion des anomalies pour la MSLIB Fortran 90
M-ST-0-79-CN
Edition 2 Révision 0
E. Le Dé, G. Prat, JC. Agnèse, JC. Bergès, R. Epenoy, D. Hautesserres, P. Legendre

DA2

Volume Z
Utilitaires de traitement des codes retour
MSLIB Fortran 90
M-MU-0-120-CIS
G. Prat

1 Introduction

Ce document présente les caractéristiques principales de la bibliothèque de mécanique spatiale MSLIB Fortran 90, ainsi que les conventions d'utilisation de la bibliothèque.

En outre, ce document explique comment sont renseignées les notices d'utilisation des routines et comment elles doivent être interprétées.

2 Qu'est ce que la MSLIB Fortran 90 ?

La MSLIB Fortran 90 est une bibliothèque de mécanique spatiale qui met à la disposition des utilisateurs différents types de "composants" logiciels.

Pour chaque logiciel de mécanique spatiale, l'assemblage de ces différents "composants" logiciels permet d'en réaliser une partie.

Ces "composants" logiciels sont :

- des routines de calcul avec leurs modules interfaces,
- des définitions de sous-types réel et sous-types entier,
- des définitions de parameters,
- des définitions de types dérivés (structure de données),
- un système de gestion des anomalies.

La MSLIB Fortran 90, c'est aussi une façon de structurer les données, que ce soit avec des vecteurs, des tableaux ou des types dérivés ; et c'est aussi une façon de concevoir un logiciel.

Dans les paragraphes qui suivent, nous allons approfondir tous ces points.

2.1 Les routines de calcul de la MSLIB Fortran 90

La MSLIB Fortran 90 propose aux utilisateurs un ensemble de routines de calcul.

Toutes ces routines sont des subroutines au sens Fortran 90, il n'y a pas de fonction. Ces dernières sont interdites dans la MSLIB Fortran 90 car en toute rigueur, elles ne peuvent pas retourner un code retour pour indiquer si le calcul s'est bien passé.

Les routines de la MSLIB Fortran 90 offrent une séquence d'appel normalisée et simple.

Les règles appliquées pour la bibliothèque MSLIB Fortran 90 sont les suivantes :

- Les routines sont nommées sur 4 à 20 caractères répartis de la manière suivante :
  - 1er caractère : le préfixe MSLIB "`m`" (1 caractère),
  - 2ème caractère : le code du thème auquel appartient la routine (1 caractère),
  - les caractères suivants : mnémonique libre (2 à 18 caractères).

  Le code MSLIB qui préfixe toutes les routines de la MSLIB permet d'éliminer le risque d'homonymie, sous réserve bien entendu que l'utilisateur élimine le caractère "`m`" en première position de sa propre nomenclature pour les noms de ses routines, de ses modules, de ses parameters, de ses variables, de ses types dérivés ...

  Le code du thème est imposé en deuxième position du nom de la routine pour permettre une localisation rapide de la routine.

A titre d'information, voici quelques exemples de code pour désigner un thème :

- la lettre "`C`" correspond au thème "Constantes",
- la lettre "`D`" correspond au thème "Date, Durée",
- la lettre "`E`" correspond au thème "Extrapolation d'orbite",
- etc...

Nota : le document [**DR2**] donne la liste et le contenu de tous les thèmes.

- Au niveau de la séquence d'appel, les paramètres d'appel de la routine sont classés de la manière suivante :
  - entrées obligatoires,
  - sorties obligatoires (autre que le code retour),
  - le code retour,
  - entrées facultatives,
  - sorties facultatives.

  Le code retour est utilisé comme séparateur vis à vis des paramètres d'appel obligatoires et des paramètres d'appel facultatifs.

  En outre, le code retour est un paramètre obligatoire qui est systématiquement présent dans la séquence d'appel de chaque routine de la MSLIB Fortran 90 (Règle Pas-m(4) de [**DR1**]).

- Afin d'améliorer la lisibilité, les paramètres d'entrées/sorties et les paramètres de travail sont interdits (Règles Pas-m(5) et Pas-m(7) de [**DR1**]).

- Les noms des routines et les noms des paramètres d'appel sont construits en utilisant ce qu'on appelle le vocabulaire MSLIB ; ainsi avec un peu d'habitude de la MSLIB Fortran 90, il est facile de deviner ce que signifie le nom d'une routine ou celui d'un paramètre d'appel.

  Par exemple "`inv_apla`" représente la constante physique qui s'appelle inverse de l'aplatissement. En annexe 1, nous donnons la liste complète du vocabulaire MSLIB ainsi que sa signification.

Il est à noter que le projet MSLIB se réserve la possibilité de modifier l'ordre et le nombre des entrées facultatives, ainsi que l'ordre et le nombre des sorties facultatives. Ceci ne nuit pas à la compatibilité ascendante compte-tenu qu'en Fortran 90, on peut nommer les paramètres d'appel ; en outre ceci permet de faire évoluer les routines. Cette possibilité est appréciable car la MSLIB Fortran 90 est appelée à "vivre" de nombreuses années.

En cas de modification de l'ordre et du nombre de paramètres facultatifs, l'utilisateur en sera informé par l'intermédiaire du volume 5 de la documentation utilisateur ("***Historique des évolutions de la MSLIB Fortran 90***", **M-MU-0-105-CIS**).

2.2 Les parameters de la MSLIB Fortran 90

La MSLIB Fortran 90 met à la disposition des utilisateurs un ensemble de parameters, appelés "`parameters MSLIB`".

Ces parameters peuvent être regroupés en cinq catégories :

- ceux qui interviennent dans la définition du "sous-type réel MSLIB" et du "sous-type entier MSLIB" (voir paragraphe 2.2.1),
- ceux qui peuvent être utilisés indépendamment de l'utilisation des routines de la MSLIB (parameters utilisés pour les constantes mathématiques, physiques ou informatiques),
- ceux qui servent de clés de sélection au niveau de la séquence d'appel de certaines routines,
- ceux qui servent pour connaître le niveau de version du binaire MSLIB utilisé,
- et ceux qui servent pour la gestion des anomalies (voir paragraphe 2.4).

En règle générale, ces parameters sont de type réel ou entier.

Les noms de ces parameters commencent tous par "`pm_`" ("`p`" pour paramètre et "`m`" pour MSLIB). Comme pour le préfixe "`m`", ce préfixe permet d'éliminer le risque d'homonymie, sous réserve que l'utilisateur n'utilise pas pour les noms de ses routines, de ses modules, de ses parameters, de ses variables, de ses types dérivés, ... , des noms qui commencent par le préfixe "`pm_`".

Comme pour les noms des routines et des paramètres d'appel, les noms des parameters sont construits en utilisant ce qu'on appelle le vocabulaire MSLIB (voir Annexe 1).

Enfin à titre exceptionnel, le projet MSLIB se réserve la possibilité de modifier la valeur d'un parameter, celui-ci gardant bien entendu la même signification physique, mathématique ou fonctionnelle. En cas de modification de la valeur d'un parameter de type physique ou mathématique, l'utilisateur en sera informé par l'intermédiaire du volume 5 de la documentation utilisateur ("***Historique des évolutions de la MSLIB Fortran 90***", **M-MU-0-105-CIS**).

Nota : Vous pouvez utiliser les parameters MSLIB de la même manière que vous utilisez vos propres parameters, à la condition qu'en début de votre sous-programme, vous ayez mis l'instruction "`use mslib`" (cf paragraphe 2.5).

2.2.1 Les parameters utilisés pour la définition du "sous-type réel MSLIB" et du "sous-type entier MSLIB" (parameters `pm_reel` et `pm_entier`)

Les types réel et entier du langage Fortran 90 se subdivisent chacun en un certain nombre de variantes appelées sous-types.

Pour le type réel, les sous-types correspondent à une précision donnée et/ou un intervalle de définition (par exemple [-10^{308}, $+10^{308}$]).

Pour le type entier, les sous-types correspondent à un intervalle de définition (par exemple [-32768, +32768]).

Selon la façon dont on utilise ces sous-types, il est possible d'écrire un code source Fortran 90 parfaitement portable avec la garantie que le sous-type réel et le sous-type entier sélectionnés à la compilation correspondent bien à votre besoin et ceci quel que soit la machine.

Ainsi pour la bibliothèque MSLIB Fortran 90, nous avons défini le "sous-type réel MSLIB" et le "sous-type entier MSLIB" :

- Le "sous-type réel MSLIB" est défini de telle manière que les réels déclarés avec ce sous-type sont représentés en machine avec un nombre de bits suffisants pour assurer au minimum 13 décimales significatives au niveau de la représentation décimale.

  Un réel x du "sous-type réel MSLIB" est à déclarer de la manière suivante :

  ```
  real(pm_reel)::x
  ```

  sachant qu'au niveau d'un des modules de la MSLIB Fortran 90, nous avons défini `pm_reel` comme suit :

  ```
  integer,parameter :: pm_reel = selected_real_kind(13)
  ```

- Le "sous-type entier MSLIB" est défini de telle manière que les entiers déclarés avec ce sous-type permettent de représenter tous les entiers appartenant à l'intervalle [-10^9, 10^9].

  Un entier `n` du "sous-type entier MSLIB" est à déclarer de la manière suivante :

  ```
  integer(pm_entier):: n
  ```

  sachant qu'au niveau d'un des modules de la MSLIB Fortran 90, nous avons défini `pm_entier` comme suit :

  ```
  integer,parameter :: pm_entier = selected_int_kind(9)
  ```

Le "sous-type réel MSLIB" est utilisé pour tous les réels de la MSLIB (les variables, les constantes, les paramètres d'appel, les variables internes, ...).

Le "sous-type entier MSLIB" est utilisé uniquement lorsqu'on manipule des entiers qui sont susceptibles d'être très grand. A ce jour, il n'est utilisé que pour les entiers manipulant le nombre de jours juliens écoulés depuis une date donnée ; cette date pouvant être, par exemple, le 31 décembre 1899 à 12 heures ou le 1er janvier 1950 à 0 heure.

Pour les autres entiers, on utilise les sous-types entier par défaut des compilateurs et on fait l'hypothèse qu'ainsi les entiers sont codés sur au moins 16 bits, ce qui permet de représenter tous les entiers appartenant à l'intervalle [-32768, +32768].

Nota : sur les machines disponibles au CNES en 1997, compte-tenu de la façon dont est défini le "sous-type réel MSLIB", la MSLIB Fortran 90 fonctionne en double précision sur SUN et en simple précision sur CRAY.

2.2.2 Liste des parameters de la MSLIB Fortran 90 pouvant être utilisés indépendamment de l'utilisation des routines

Voici la liste des parameters pouvant être utilisés indépendamment de l'utilisation des routines :

Type	Nom du parameter	Signification mathématique, physique ou informatique
réel	pm_pi	π
réel	pm_deux_pi	$2\,\pi$
réel	pm_pi_sur2	$\pi/2$
réel	pm_deg_rad	constante de conversion des degrés en radians ($\pi/180$)
réel	pm_rad_deg	constante de conversion des radians en degrés ($180/\pi$)
réel	pm_ua	unité astronomique (en **km**)
réel	pm_vit_lum	vitesse de la lumière dans le vide (en m/s)
réel	pm_i_critique_non_retro	inclinaison critique non rétrograde (rad) solution de $1 - 5 \times (\cos i)^2 = 0$
réel	pm_i_critique_retro	inclinaison critique rétrograde (rad) solution de $1 - 5 \times (\cos i)^2 = 0$
réel	pm_r_equa_GRS1980	rayon équatorial terrestre dans le système GRS 1980 (en m)
réel	pm_inv_apla_GRS1980	inverse de l'aplatissement terrestre dans le système GRS 1980
réel	pm_apla_GRS1980	aplatissement terrestre dans le système GRS 1980
réel	pm_mercure_r_equa_UAI	rayon équatorial de Mercure dans le système UAI (en m) d'après [**DR3**]
réel	pm_mercure_apla_UAI	aplatissement de Mercure dans le système UAI d'après [**DR3**]
réel	pm_venus_r_equa_UAI	rayon équatorial de Vénus dans le système UAI (en m) d'après [**DR3**]
réel	pm_venus_apla_UAI	aplatissement de Vénus dans le système UAI d'après [**DR3**]
réel	pm_terre_r_equa_UAI	rayon équatorial de la Terre dans le système UAI (en m) d'après [**DR3**]

Type	Nom du parameter	Signification mathématique, physique ou informatique
réel	pm_terre_apla_UAI	aplatissement de la Terre dans le système UAI d'après [**DR3**]
réel	pm_mars_r_equa_UAI	rayon équatorial de Mars dans le système UAI (en m) d'après [**DR3**]
réel	pm_mars_apla_UAI	aplatissement de Mars dans le système UAI d'après [**DR3**]
réel	pm_jupiter_r_equa_UAI	rayon équatorial de Jupiter dans le système UAI (en m) d'après [**DR3**]
réel	pm_jupiter_apla_UAI	aplatissement de Jupiter dans le système UAI d'après [**DR3**]
réel	pm_saturne_r_equa_UAI	rayon équatorial de Saturne dans le système UAI (en m) d'après [**DR3**]
réel	pm_saturne_apla_UAI	aplatissement de Saturne dans le système UAI d'après [**DR3**]
réel	pm_uranus_r_equa_UAI	rayon équatorial d'Uranus dans le système UAI (en m) d'après [**DR3**]
réel	pm_uranus_apla_UAI	aplatissement d'Uranus dans le système UAI d'après [**DR3**]
réel	pm_neptune_r_equa_UAI	rayon équatorial de Neptune dans le système UAI (en m) d'après [**DR3**]
réel	pm_neptune_apla_UAI	aplatissement de Neptune dans le système UAI d'après [**DR3**]
réel	pm_pluton_r_equa_UAI	rayon équatorial de Pluton dans le système UAI (en m) d'après [**DR3**]
réel	pm_pluton_apla_UAI	aplatissement de Pluton dans le système UAI d'après [**DR3**]
entier	pm_reel	parameter intervenant dans la définition du "sous-type réel MSLIB". Pour plus d'information, voir paragraphe 2.2.1
entier	pm_entier	parameter intervenant dans la définition du "sous-type entier MSLIB". Pour plus d'information, voir paragraphe 2.2.1

Nota : dans le tableau ci-dessus, tous les parameters de type réel utilise le "sous-type réel MSLIB" et tous les parameters de type entier utilise le sous-type entier par défaut. Le "sous-type réel MSLIB" est présenté au paragraphe 2.2.1.

Les valeurs des paramètres définissant les planètes correspondent aux numéros retenus par la bibliothèque NAIF / SPICE de la NASA:

Planète	*Valeur NAIF / SPICE*	*nom du parameter associé*
Mercure	199	`pm_pla_mercure`
Venus	299	`pm_pla_venus`
Terre	399	`pm_pla_terre`
Mars	499	`pm_pla_mars`
Jupiter	599	`pm_pla_jupiter`
Saturne	699	`pm_pla_saturne`
Uranus	799	`pm_pla_uranus`
Neptune	899	`pm_pla_neptune`
Pluton	999	`pm_pla_pluton`

Rappel : Vous pouvez utiliser les parameters MSLIB de la même manière que vous utilisez vos propres parameters, à la condition qu'en début de votre sous-programme, vous ayez mis l'instruction "`use mslib`" (cf. paragraphe 2.5).

Exemple de programmation pour une routine utilisant les parameters MSLIB "`pm_reel`" et "`pm_pi`":

```
subroutine SURFACE_SPHERE (RAYON,SURFACE)
use mslib
real(pm_reel),intent(in)::RAYON
real(pm_reel),intent(out)::SURFACE
SURFACE = 4._pm_reel*pm_pi*RAYON**2
end subroutine SURFACE_SPHERE
```

Remarque : la routine ci-dessus peut être programmée plus simplement, mais dans ce cas on ne peut plus garantir la portabilité.

2.2.3 Les parameters utilisés en tant que clé de sélection au niveau de la séquence d'appel de certaines routines

Pour expliquer le rôle de ces parameters, nous allons traiter un exemple à l'aide de la routine "`mr_rep_fon`".

Cette routine est utilisée pour calculer la matrice de passage d'un repère fondamental R_i à un repère

fondamental R_j sachant qu'il existe 4 types de repères fondamentaux :

- les repères équatoriaux moyens,
- les repères équatoriaux vrais,
- les repères écliptiques moyens,
- les repères écliptiques vrais.

Pour indiquer la transformation que l'on veut faire, autrement dit le type du repère de départ et le type du repère d'arrivée, on utilise dans la séquence d'appel un paramètre d'entrée de type entier : ce paramètre sert de clé de sélection.

Supposons que pour passer d'un repère équatorial moyen à un repère écliptique vrai, la clé vaille 14.

Si dans le code source de l'utilisateur, il était écrit :

```
...
cle = 14
call mr_rep_fon(cle,...)
...
```

alors il y aurait des problèmes de maintenance et de lisibilité. Aussi, pour remédier à ce problème, nous proposons pour cette routine un ensemble cohérent de parameters MSLIB, ce qui permet à l'utilisateur d'écrire son code source de la manière suivante :

```
...
cle = pm_equa_moy_ecli_vrai
call mr_rep_fon(cle,...)
...
```

A la lecture de ce code source, il est possible pour quelqu'un qui n'aurait pas écrit lui-même ce code source, de deviner que l'on cherche à passer d'un repère équatorial moyen à un repère écliptique vrai, ce qui n'était pas le cas lorsqu'on écrivait l'instruction "`cle = 14`".

Cet exemple illustre le rôle et l'intérêt de ces parameters.

Par ailleurs, pour une routine donnée et une clé donnée, au niveau de la notice d'utilisation, nous avons décidé de fournir uniquement la liste des noms des parameters qui peuvent être utilisés : nous ne donnons pas leurs valeurs.

Ainsi, pour reprendre l'exemple de la routine `mr_rep_fon`, dans sa notice d'utilisation, nous indiquons que la clé peut valoir "`pm_equa_moy_ecli_vrai`" mais nous n'indiquons pas que cette clé peut valoir "`14`" ou que l'on a "`pm_equa_moy_ecli_vrai=14`".

Ce choix a été retenu pour nous permettre de modifier les valeurs des parameters MSLIB sans que cela ait un impact pour les utilisateurs, et ceci compte-tenu que les utilisateurs ne sont pas sensés utiliser explicitement les valeurs des parameters MSLIB.

2.2.4 Le parameter utilisé pour connaître le niveau de version d’un binaire MSLIB

Tout utilisateur d’un binaire MSLIB Fortran 90 peut connaître le niveau de version de son binaire.

Cette information est contenue dans le parameter MSLIB "`pm_version_MSLIB90`".

Pour accéder à cette information, il suffit d’écrire :

```
write (*, *) pm_version_MSLIB90
```

Nota : le parameter "`pm_version_MSLIB90`" est de type chaîne de caractères.

2.3 Structuration des données pour la MSLIB Fortran 90

2.3.1 Généralités

La structuration des données a été l'une des principales priorités au cours de la phase de spécification de la MSLIB Fortran 90.

Un ensemble de jeux de données utiles à la mécanique spatiale a été défini, et pour chaque jeu de données, nous avons défini quel était la structuration la plus adaptée.

Pour les vecteurs position et les vecteurs vitesse en coordonnées cartésiennes, pour les matrices de changement de repères et pour les jacobiennes de transformation, nous utilisons des vecteurs ou des tableaux au sens Fortran 90. Par contre, pour pratiquement tous les autres jeux de données, nous utilisons des types dérivés.

Sur l'ensemble de la bibliothèque MSLIB, nous utilisons les mêmes structures de données, et pour chaque jeu de données, il n'existe qu'une seule structure (vecteur, tableau ou type dérivé).

Les types dérivés de la MSLIB Fortran 90 peuvent très bien être utilisés indépendamment de l'utilisation des routines de la MSLIB Fortran 90.

L'utilisateur peut aussi définir ses propres types dérivés à partir des types dérivés de la MSLIB. Par exemple, un utilisateur peut définir un type dérivé "bulletin" constitué des deux types dérivés MSLIB suivant :

- le type dérivé "paramètres orbitaux adaptés aux orbites circulaires équatoriales",
- le type dérivé "date julienne".

Pour utiliser les types dérivés de la MSLIB, vous devez mettre au début de votre sous-programme l'instruction "`use mslib`" (cf. paragraphe 2.5).

Les noms des types dérivés MSLIB commencent tous par "`tm_`" ("`t`" pour type et "`m`" pour MSLIB). Comme pour les préfixes "`m`" et "`pm_`", ce préfixe permet d'éliminer le risque d'homonymie, sous réserve que l'utilisateur n'utilise pas pour les noms de ses routines, de ses modules, de ses parameters, de ses variables, de ses types dérivés, ... , des noms qui commencent par le préfixe "`tm_`".

De plus, comme pour les noms des parameters MSLIB, des routines MSLIB et de leurs paramètres d'appel, les noms des types dérivés MSLIB sont construits en utilisant ce qu'on appelle le vocabulaire MSLIB (voir Annexe 1).

Il est à noter que la déclaration de ces types dérivés se fait avec l'attribut "`sequence`", afin qu'il soit possible d'utiliser ces types dérivés dans des "common".

2.3.2 Liste et définition des types dérivés MSLIB

La liste de tous les types dérivés MSLIB est donnée ci-dessous par ordre alphabétique. Pour chacun de ces types, nous donnons :

- le nom du type,
- les cas dans lesquels il est utilisé,
- le nom des champs constituant le type,

- le type des champs (entier, ...),
- la signification de tous les champs et leur unité (rad, m, ...).

Le type dérivé "tm_code_retour"

Comme son nom l'indique, ce type dérivé est utilisé pour le code retour. Toutes les routines de la MSLIB Fortran 90 accessibles aux utilisateurs utilisent dans leur séquence d'appel le type dérivé "tm_code_retour".

Nom du type	Nom des champs	Signification des champs
`tm_code_retour`	`% valeur`	Valeur du code retour. Pour plus d'explication, voir paragraphe 2.4
	`% routine`	Numéro de la routine. Pour plus d'explication, voir paragraphe 2.4
	`% biblio`	Bibliothèque. Pour plus d'explication, voir paragraphe 2.4
	`% message`	Message. Pour plus d'explication, voir paragraphe 2.4

Type des champs :

- les champs `"valeur"`, `"routine"` et `"biblio"` sont de type entier avec le sous-type entier par défaut : `integer`.
- le champ `"message"` est une chaîne de caractères : `character(len=pm_message)`. Avec `pm_message` = 512

Unités utilisées : sans unité.

Le type dérivé "tm_geocentrique"

Comme son nom l'indique, ce type dérivé est utilisé pour les données positions exprimées en coordonnées géocentriques (latitude, longitude, distance centre Terre).

Nom du type	Nom des champs	Signification des champs[1]
`tm_geocentrique`	`% lat`	Latitude géocentrique.
	`% long`	Longitude.
	`% dist`	Distance centre Terre.

1. Les coordonnées géocentriques sont définies dans la présentation du thème T.

Type des champs :

- tous les champs sont de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.

Unités utilisées : rad, m.

Le type dérivé "tm_geodesique"

Comme son nom l'indique, ce type dérivé est utilisé pour les données positions exprimées en coordonnées géodésiques (latitude, longitude, hauteur).

Nom du type	Nom des champs	Signification des champs[1]
`tm_geodesique`	`% lat`	Latitude géodésique.
	`% long`	Longitude.
	`% haut`	Hauteur.

1. Les coordonnées géodésiques sont définies dans la présentation du thème T.

Type des champs :

- tous les champs sont de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.

Unités utilisées : rad, m.

Le type dérivé "tm_jour_sec"

Ce type dérivé est utilisé aussi bien pour des dates (date julienne) que pour des durées.

Nom du type	Nom des champs	Signification des champs
`tm_jour_sec`	`% jour`	Nombre de jours.
	`% sec`	Nombre de secondes dans le jour.

Type des champs :

- le champ `"jour"` est de type entier avec le "sous-type entier MSLIB" : `integer(pm_entier)`.
- le champ `"sec"` est de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.

Unités utilisées : jour, s.

Le type dérivé "tm_nuta"

Ce type de structure est utilisé aussi bien pour la nutation que pour les dérivées première et seconde de la nutation.

Nom du type	Nom des champs	Signification des champs
`tm_nuta`	`% long`	Nutation en longitude, ou dérivé première de la nutation en longitude, ou dérivé seconde de la nutation en longitude.
	`% obli`	Nutation en obliquité, ou dérivé première de la nutation en obliquité, ou dérivé seconde de la nutation en obliquité.

Type des champs :

- tous les champs sont de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.

Unités utilisées : rad, rad/s, rad/s^2.

Le type dérivé "tm_orb_cir"

Ce type dérivé est utilisé aussi bien pour les paramètres osculateurs, que pour les paramètres moyens, ou que pour les écarts admissibles dans les routines d'extrapolation d'orbite.

Nom du type	Nom des champs	Signification des champs[1]
`tm_orb_cir`	`% a`	a
	`% ex`	e_x
	`% ey`	e_y
	`% i`	i
	`% gom`	Ω
	`% pso_M`	$\omega + M$

1. les notations a, e_x, e_y, i, Ω, $\omega + M$ sont expliquées dans la présentation du thème V.

Type des champs :

- tous les champs sont de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.

Unités utilisées : m, rad.

Le type dérivé "tm_orb_cir_equa"

Ce type dérivé est utilisé aussi bien pour les paramètres osculateurs, que pour les paramètres moyens, ou que pour les écarts admissibles dans les routines d'extrapolation d'orbite.

Nom du type	Nom des champs	Signification des champs[1]
`tm_orb_cir_equa`	`% a`	a
	`% ex`	e_x
	`% ey`	e_y
	`% ix`	i_x
	`% iy`	i_y
	`% pso_M`	$\omega + \Omega + M$

1. les notations a, e_x, e_y, i_x, i_y, ω, Ω et M sont expliquées dans la présentation du thème V.

Type des champs :

- tous les champs sont de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.

Unités utilisées : m, rad.

Le type dérivé "tm_orb_equa"

Ce type dérivé est utilisé aussi bien pour les paramètres osculateurs, que pour les paramètres moyens, ou que pour les écarts admissibles dans les routines d'extrapolation d'orbite.

Nom du type	Nom des champs	Signification des champs[1]
`tm_orb_equa`	`% a`	a ou p
	`% e`	e
	`% pgom`	$\omega + \Omega$
	`% ix`	i_x
	`% iy`	i_y
	`% M`	M

1. les notations a, p, e, ω, Ω, i_x, i_y, et M sont expliquées dans la présentation du thème V.

Type des champs :

- tous les champs sont de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.

Unités utilisées : m, rad.

Le type dérivé "tm_orb_kep"

Ce type dérivé est utilisé aussi bien pour les paramètres osculateurs, que pour les paramètres moyens, ou que pour les écarts admissibles dans les routines d'extrapolation d'orbite.

Nom du type	Nom des champs	Signification des champs[1]
`tm_orb_kep`	`% a`	a ou p
	`% e`	e
	`% i`	i
	`% pom`	ω
	`% gom`	Ω
	`% M`	M

1. les notations a, p, e, i, ω, Ω et M sont expliquées dans la présentation du thème V.

Type des champs :

- tous les champs sont de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.

Unités utilisées : m, rad.

Le type dérivé "tm_quat"

Ce type dérivé permet de définir un quaternion $\tilde{q} = q_0 + q_1 \cdot i + q_2 \cdot j + q_3 \cdot k$, avec ($i$, j, k) des nombres imaginaires.

Nom du type	Nom des champs	Signification des champs
`tm_quat`	`% q0`	la partie réelle q_0 du quaternion
	`% q123`	la partie imaginaire (q_1, q_2, q_3) du quaternion (représentée sous forme de vecteur)

Type des champs :

- le champ `q0` est de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.
- le vecteur `q123` est de type réel avec le "sous-type réel MSLIB", et il est de dimension 3: `real(pm_reel), dimension(1:3)`.

Unités utilisées : sans unité.

Le type dérivé "tm_sgd"

Ce type dérivé est utilisé pour les données positions ou les données vitesses exprimées en coordonnées site/gisement/distance.

Nom du type	Nom des champs	Signification des champs
`tm_sgd`	`% s`	Composante site en position ou en vitesse.
	`% g`	Composante gisement en position ou en vitesse.
	`% d`	Composante distance en position ou en vitesse.

Type des champs :

- tous les champs sont de type réel avec le "sous-type réel MSLIB" : `real(pm_reel)`.

Unités utilisées : rad, rad/s, m, m/s.

2.3.3 Liste des jeux de données pour lesquels la MSLIB Fortran 90 utilise des vecteurs ou des tableaux

La liste des jeux de données pour lesquels la MSLIB Fortran 90 utilise des vecteurs ou des tableaux au sens Fortran 90, est donnée ci-dessous.

Dans ce paragraphe, nous ne citons que les jeux de données qui sont couramment utilisés dans la MSLIB Fortran 90.

Pour chacun de ces jeux de données, nous donnons :

- la définition du jeu de données,
- la définition de la structure de données qui contient ce jeu de données.

Définition du jeu de données	Définition de la structure de données utilisées
Position en coordonnées cartésiennes (x, y, z)	Vecteur de réels de dimension 3
Vitesse en coordonnées cartésiennes ($\dot{x}$, $\dot{y}$, $\dot{z}$)	Vecteur de réels de dimension 3
Coefficients harmoniques zonaux (CN0)	Vecteur de réels de dimension variable suivant les modèles utilisés
Jacobienne des transformations	Tableau de réels Les dimensions couramment utilisées sont (3, 3) ou (6, 6)
Matrice de passage d'un repère à un autre	Tableau de réels Les dimensions couramment utilisées sont (3, 3) ou (6, 6)

Nota : pour les positions et les vitesses en coordonnées cartésiennes, nous utilisons des vecteurs de réels, plutôt que des types dérivés, de manière à pouvoir effectuer simplement des produits scalaire, des produits vectoriels ou des produits par des matrices. Nous utilisons ainsi l'un des grands apports du langage Fortran 90.

2.4 Gestion des anomalies

Toutes les routines de la MSLIB Fortran 90 disposent d'un code retour dans leur séquence d'appel.

Ces codes retours sont de type dérivé "`tm_code_retour`", et ils contiennent les champs suivants :

- "`valeur`",
- "`routine`",
- "`biblio`",
- "`message`".

Le champ "`valeur`" est prévu pour permettre d'effectuer un diagnostic sur les sorties de la routine.

Quant aux champs "`routine`" et "`biblio`", leur rôle est de fournir la valeur du numéro d'identification de la routine MSLIB, ainsi que le numéro d'identification de la bibliothèque

Attention: le numéro d'identification de la bibliothèque ne doit pas être confondu avec le numéro de version de la bibliothèque.

Le champ "`message`" permet de son côté de fournir des éléments supplémentaires sur la cause de l'anomalie éventuelle.

Nota : les champs "`valeur`", "`routine`" et "`biblio`" sont de type entier,
le champ "`message`" est une chaîne de caractère (de longueur `pm_message`).

2.4.1 Le champ "valeur" du code retour

Le principal champ au niveau de la structure code retour, c'est le champ "`valeur`" :

- s'il est nul, alors toutes les sorties de la routine sont utilisables,
- s'il est négatif, alors il y a au moins une partie des sorties de la routine qui ne sont pas utilisables,
- s'il est positif, alors pour au moins une partie des sorties de la routine, on est dans une situation où seul l'utilisateur peut juger si celles-ci sont utilisables, compte-tenu du contexte dans lequel il utilise la routine : c'est un avertissement.

Ceci peut se résumer de la manière suivante :

```
code_retour%valeur = 0   :  retour normal
code_retour%valeur < 0   :  erreur grave ou fatale
code_retour%valeur > 0   :  avertissement (warning)
```

Les causes possibles pour un avertissement sont les suivantes :

- il y a une infinité de solutions pour un des paramètres de sortie, et la routine en a retenu une de manière arbitraire,
- un traitement particulier a été effectué compte-tenu de la nature des données d'entrées,
- une donnée d'entrée a été interprétée car sa valeur ne semblait pas correcte,
- la précision numérique des sorties est dégradée compte-tenu de la plage dans laquelle se trouvent les entrées de la routine,
- etc...

Pour la MSLIB Fortran 90, nous avons retenu un système de gestion centralisée pour les anomalies : ainsi si plusieurs routines peuvent avoir la même cause d'anomalie, alors toutes ces routines utiliseront une même valeur pour le champ "`valeur`" du code retour, et cette valeur sera stockée dans un parameter MSLIB (voir paragraphe 2.2) défini de manière unique.

Pour le nommage des parameters MSLIB utilisés pour le champ "`valeur`" du code retour, nous utilisons les conventions suivantes :

- s'il s'agit d'une erreur, alors le nom du parameter commence par "`pm_err_`",
- s'il s'agit d'un avertissement, alors le nom du parameter commence par "`pm_warn_`".

Exemple :
Dans le système de gestion des anomalies, il existe un parameter MSLIB nommé "`pm_err_i_negatif`". A chaque fois qu'il y a un problème dans une routine parce que l'inclinaison est négative, on affecte au champ "`valeur`" du code retour la valeur du parameter "`pm_err_i_negatif`".

Ce système de gestion des anomalies présente les avantages suivants :

- la lisibilité du code source de la MSLIB Fortran 90 est nettement améliorée notamment lorsqu'il y a des appels de routines en cascade : les cause d'anomalies remontent bien jusqu'à la routine principale,
- ce système permet de faire facilement l'inventaire des causes possibles d'anomalies, ce qui va dans le sens d'une meilleure maîtrise du logiciel (voir annexe 2),
- le traitement des codes retours par l'utilisateur est simplifié et ce d'autant plus qu'il utilise plusieurs routines de la MSLIB pouvant avoir les mêmes causes d'anomalies,
- enfin l'affichage de messages d'erreur devient possible à un faible coût.

Le système MSLIB de gestion des anomalies a été défini de telle manière qu'il puisse coexister avec celui de l'utilisateur. Ainsi, la valeur du champ "`valeur`" du code retour MSLIB doit appartenir à une des trois plages de valeurs suivantes :

- [-3000, -1000] (pour les erreurs),
- [0, 0] (pour le fonctionnement nominal),
- [+1000, +3000] (pour les avertissements),

de ce fait, les autres plages sont disponibles pour le système de gestion des anomalies de l'utilisateur.

Les plages de valeurs citées ci-dessus sont découpées en sous-plages qui correspondent à une famille de causes possibles d'anomalies du type :

- problème sur la valeur d'une constante physique,
- problème sur la valeur du demi grand axe d'une conique (a) ou sur le paramètre de la parabole (p),
- problème sur la valeur de l'excentricité (e),
- problème sur la valeur de l'inclinaison (i) ou sur le vecteur inclinaison (i_x, i_y),
- problème sur une durée ou une date,
- problème sur la position et/ou la vitesse,
- etc...

Pour chaque routine de la MSLIB Fortran 90, la documentation utilisateur donne la liste des valeurs possibles pour le champ "`valeur`".

Il est à noter que lors d'une montée de niveau de la MSLIB, cette liste peut évoluer : c'est le cas notamment lorsqu'on ajoute un test sur les données d'entrée. En conséquence, pour pouvoir bénéficier de la compatibilité ascendante, lorsqu'on teste le champ "`valeur`", à l'aide d'une instruction "`select case`" par exemple, il ne faut pas oublier de mettre une clause "`default`".

L'utilisateur qui aurait besoin de plus amples informations pourra consulter la note [**DA1**].

2.4.2 Le champ "`routine`" du code retour

A chaque routine de la MSLIB Fortran 90, nous avons associé un numéro d'identification unique. Celui-ci est contenu dans le champ "`routine`" du code retour. Aussi pour une routine donnée, le champ "`routine`" est constant, et ceci quel que soit le contenu du champ "`valeur`".

Les principaux intérêts de ce champ sont les suivants :

- au niveau de la structure code retour, on peut associer à une cause d'anomalie, le numéro d'identification de la routine MSLIB qui en informe l'utilisateur,
- l'affichage du nom et du rôle de la routine MSLIB ayant informé l'utilisateur d'un problème devient possible à un faible coût.

Nota : De même que pour le champ "`valeur`", les numéros d'identification des routines sont stockés dans des parameters MSLIB.

Pour le nommage de ces parameters, nous utilisons la convention suivante :
pour une routine donnée, le nom du parameter associé est fabriqué en ajoutant "`pm_num_`" devant le nom de la routine.

Exemple : le numéro d'identification de la routine `mu_angle2` est stocké dans le parameter MSLIB dont le nom est `pm_num_mu_angle2`.

2.4.3 Le champ "biblio" du code retour

Le champ "`biblio`" du code retour contient un numéro d'identification de la bibliothèque utilisée.

Cas de la MSLIB Fortran 90:
De même que pour le champ "`valeur`", le numéro d'identification de la bibliothèque est stocké dans un parameter MSLIB.

Exemple pour la MSLIB Fortran 90, le champ "`biblio`" vaut `pm_mslib90`.

2.4.4 Le champ "message" du code retour

Le champ "`message`" du code retour contient des explications complémentaires sur la cause de l'anomalie, sous forme textuelle.

Nota : le champ "`message`" a été créé pour préparer l'avenir. Pour l'instant, le champ "`message`" est initialisé à une chaîne vide. De plus, pour des raisons de performance en temps calcul, son affectation n'est effectuée que dans le cas où le champ "`valeur`" est différent de `pm_OK`.

2.4.5 Les routines de traitement du code retour

Le projet MSLIB met à la disposition des utilisateurs, un ensemble de routines qui servent à traiter la structure code retour.

Ces routines sont de deux types :

- il y a celles qui servent à générer un message, ce message pouvant contenir : soit le libellé de l'anomalie rencontrée, soit le nom et le rôle de la routine MSLIB qui informe l'utilisateur de cette anomalie,
- il y a celles qui servent à écrire un message et à arrêter l'exécution selon la valeur du champ "`valeur`" et selon les choix de l'utilisateur.

Les routines du premier groupe font partie de la MSLIB Fortran 90. Quant aux routines du deuxième groupe, elles ne font pas partie de la MSLIB Fortran 90 ; elles sont livrées aux utilisateurs sous forme de code source de manière à ce que ces derniers puissent les adapter à leurs besoins.

Toutes ces routines sont présentées dans le volume Z [**DA2**].

2.5 Le module MSLIB

Compte-tenu des choix qui ont été retenus au niveau de l'architecture informatique de la bibliothèque, dès que l'on veut utiliser une routine MSLIB, un parameter MSLIB ou un type dérivé MSLIB, il est obligatoire d'utiliser le module MSLIB. Ceci se fait à l'aide de l'instruction "`use mslib`". Celle-ci doit figurer en début de chaque sous-programme utilisant la bibliothèque MSLIB.

Par ailleurs, grâce au module MSLIB, les interfaces explicites de toutes les routines de la MSLIB deviennent connues du compilateur, ce qui permet de vérifier que vous utilisez des séquences d'appel informatiquement correctes.

Nota : Tous les modules internes de la bibliothèque MSLIB ont des noms qui contiennent la séquence de caractères "`mslib`".

2.6 Ce qu'il n'y a pas dans la MSLIB Fortran 90

Afin de faciliter la maintenance et les portages, et pour limiter les risques de conflit avec les applications des utilisateurs, un certain nombre de choses sont interdites dans la MSLIB Fortran 90. Parmi celles-ci, on peut citer :

- les instructions COMMON,
- l'utilisation de fichier de données,
- les instructions READ et WRITE,
- les appels à des bibliothèques externes.

3 Conseils d'utilisation pour la MSLIB Fortran 90

Dans ce chapitre, nous donnons une liste de conseils d'utilisation pour la MSLIB Fortran 90. Cette liste n'a pas la prétention d'être exhaustive, en conséquence, nous recommandons à tous les utilisateurs de lire attentivement le chapitre précédent (chapitre 2 : Qu'est ce que la MSLIB Fortran 90 ?).

3.1 Conseils d'utilisation pour la MSLIB Fortran 90 dans le cadre d'un logiciel à développer

Les conseils suivants ont pour but de vous aider à développer un logiciel basé sur l'utilisation de la MSLIB Fortran 90 et ayant l'architecture informatique la plus simple possible. Ces conseils résultent des caractéristiques de la MSLIB Fortran 90 telles qu'elles ont été définies dans le chapitre précédent (chapitre 2).

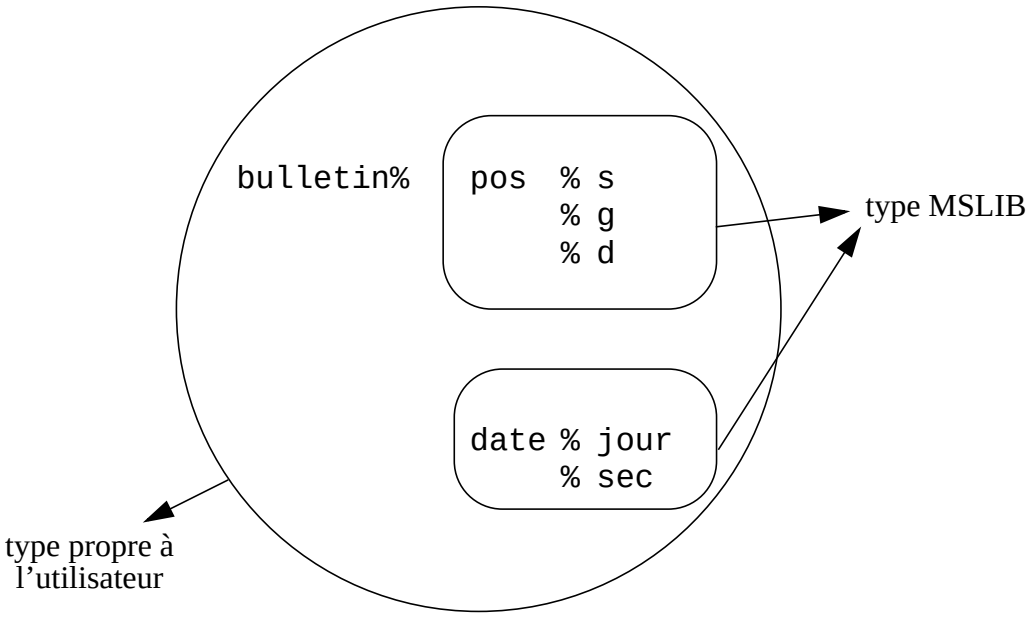
***Conseil util(1)* :**

Utiliser les mêmes structures de données que celles de la MSLIB, cela vous évitera d'avoir à faire des changements de structure de données avant d'appeler une routine MSLIB.
Les listes des structures de données utilisées dans la MSLIB sont fournies au paragraphe 2.3.

Vous pouvez très bien créer pour votre application des structures qui englobent celles de la MSLIB.

Exemple :

Il existe dans la MSLIB, une structure position avec des champs site (`s`), gisement (`g`) et distance (`d`), et une structure date avec des champs jour (`jour`) et seconde (`sec`). Si vous créez une structure bulletin qui est composée des deux structures MSLIB précédentes, alors cette nouvelle structure sera parfaitement compatible avec ce qui est nécessaire au niveau des séquences d'appel des routines de la MSLIB.



Conseil util(2) **:**

Au niveau des séquences d'appel, nommer tous les paramètres optionnels des routines MSLIB.

Justification : si à l'occasion d'une montée de version MSLIB, on ajoute un paramètre optionnel à une routine de la MSLIB, et si celui-ci n'est pas placé en dernière position parmi les paramètres optionnels, il y aura une compatibilité ascendante entre les différentes versions de la MSLIB pour ceux qui nomment les paramètres optionnels.

Conseil util(3) **:**

Déclarer les réels en utilisant systématiquement le "sous-type réel MSLIB" (voir paragraphe 2.2.1).

Déclarer les entiers en utilisant à chaque fois que cela est demandé le "sous-type entier MSLIB" ; dans les autres cas, utiliser le sous-type entier par défaut (voir paragraphe 2.2.1).

Justification : le respect de cette règle permet d'assurer simplement la cohérence entre le logiciel de l'utilisateur et la MSLIB Fortran 90, et elle assure une bonne portabilité au logiciel de l'utilisateur.

Conseil util(4) **:**

Ne pas utiliser explicitement la valeur d'un parameter MSLIB ; utiliser uniquement son nom.

Justification :

- tout en gardant le même sens, la valeur d'un parameter MSLIB peut évoluer ; ceci peut être le cas d'un parameter MSLIB utilisé comme clé de sélection au niveau de la séquence d'appel d'une routine, ou encore d'un parameter MSLIB utilisé pour la gestion des anomalies ;
- de manière exceptionnelle, un parameter MSLIB peut être amené à disparaître suite à une montée de niveau (code retour devenu inutile, option de calcul supprimée, ...), auquel cas si l'utilisateur utilise uniquement le nom du parameter MSLIB (et non sa valeur) alors la suppression du parameter MSLIB sera détectée au niveau de la compilation ;
- au niveau du code source de l'utilisateur, utiliser le nom du parameter MSLIB apporte plus de lisibilité qu'utiliser sa valeur.

Conseil util(5) **:**

Tester systématiquement les codes retour des routines MSLIB.

Justification :

- les codes retour constituent l'unique moyen utilisé par la MSLIB Fortran 90 pour informer l'utilisateur d'un éventuel problème.

Conseil util(6) **:**

Pour la gestion des anomalies, utiliser les routines de traitement du code retour ; celles-ci sont présentées dans le volume Z [**DA2**].

Si elles ne répondent pas complètement à votre besoin, développez votre propre routine de traitement du code retour en utilisant celles qui sont proposées dans le volume Z.

Justification : le respect de cette règle devrait vous permettre d'alléger le code source de votre application.

Conseil util(7) **:**

Lorsque vous testez la valeur du champ "`valeur`" du code retour, à l'aide d'une instruction "`select case`" par exemple, il ne faut pas oublier de mettre une clause "`default`".

Justification : suite à une montée de niveau de la MSLIB, la liste des valeurs possibles pour le champ "`valeur`" peut évoluer : c'est le cas notamment lorsqu'on ajoute un test sur les données d'entrée. Si vous avez prévu une clause "`default`", votre code source sera plus robuste.

Conseil util(8) **:**

Pour les noms de vos routines, de vos modules, de vos parameters, de vos variables, de vos types dérivés, ... , ne pas utiliser des noms qui commencent par les préfixes "`m`", "`pm_`" ou "`tm_`", et ne pas utiliser des noms qui contiennent la séquence de caractères "`mslib`".

Justification : le respect de cette règle élimine tout risque d'homonymie entre les routines, les modules, les parameters et les types dérivés de la MSLIB Fortran 90 et les "objets" de l'utilisateur (routines, modules, parameters, variables, types dérivés, ...).

3.2 Conseils d'utilisation pour la MSLIB Fortran 90 dans le cadre d'un logiciel existant

Si vous souhaitez utiliser la MSLIB Fortran 90 dans un logiciel qui existe déjà, il est impératif de vérifier qu'il n'y a pas de risque d'homonymie entre les routines, les modules, les parameters et les types dérivés de la MSLIB Fortran 90 et ceux de votre logiciel.

Concrètement, vous devez vérifier que les noms de vos routines, de vos modules, de vos parameters, de vos variables, de vos types dérivés, ... , ne commencent pas par les préfixes "`m`", "`pm_`", ou "`tm_`", et qu'ils ne contiennent pas la séquence de caractères "`mslib`".

S'il s'avère qu'il y a des homonymies qui sont clairement établies, par exemple, vous connaissez une routine MSLIB qui porte le même nom que l'une de vos routines, alors votre logiciel n'est pas dans l'état actuel compatible avec la MSLIB Fortran 90.
Une évolution de votre logiciel est nécessaire si vous souhaitez utiliser la MSLIB Fortran 90.

S'il y a uniquement des risques d'homonymies, alors il y a potentiellement un risque d'incompatibilité entre votre logiciel et la MSLIB Fortran 90. De plus à chaque montée de niveau de la MSLIB Fortran 90, ce risque réapparaîtra (voir Conseil util (8) du paragraphe 3.1).

Dès que ces problèmes de compatibilité sont traités, vous pouvez utiliser la MSLIB Fortran 90.
Nous vous recommandons cependant de suivre les conseils cités dans les paragraphes suivants.

Ceux-ci sont regroupés en deux listes :

- ceux qui sont faciles à suivre sans remettre en cause de manière importante votre logiciel,
- ceux qui peuvent remettre en cause de manière importante votre logiciel.

3.2.1 Les conseils faciles à suivre

Les conseils listés ci-dessous sont les conseils que vous pouvez toujours suivre et ceci quel que soit votre application. Ils ne remettent jamais en cause de manière importante votre logiciel.

L'énoncé de ces conseils est donné dans le chapitre 3.1 (Conseils d'utilisation pour la MSLIB Fortran 90 dans le cadre d'un logiciel à développer).

Ces conseils sont :

- Util(2),
- Util(4),
- Util(5),
- Util(6),
- Util(7).

3.2.2 Les autres conseils

Dans ce paragraphe, nous présentons les deux conseils qui peuvent remettre en cause de manière importante votre logiciel.

3.2.2.1 Cas du conseil Util(1)

L'énoncé du conseil Util(1) est donné dans le chapitre 3.1.

Si vous ne suivez pas ce conseil, au niveau de l'appel des routines de la MSLIB, le code source de votre application risque d'être un peu lourd et de ne pas avoir une lisibilité optimale. Pour atténuer ce problème, vous pouvez écrire des routines qui encapsulent les routines de la MSLIB et qui effectuent les changements de structures de données nécessaires.

3.2.2.2 Cas du conseil Util(3)

L'énoncé du conseil Util(3) est donné dans le chapitre 3.1.

Si vous ne suivez pas ce conseil, vous devez vérifier qu'il y a cohérence au niveau de la signification entre les sous-types de votre application et ceux de la MSLIB.
Si ce n'est pas le cas, vous devrez effectuer des conversions de sous-type avant d'appeler les routines de la MSLIB.

En outre, si vous effectuez un portage, vous serez à nouveau obligés de vérifier la cohérence entre les différents sous-types.

Nota 1 : il est envisageable de fournir aux utilisateurs qui en ferait la demande, une version de la bibliothèque MSLIB Fortran 90 avec des sous-types réel et entier différents de ce qu'il y a habituellement de manière standard dans la MSLIB Fortran 90.

Nota 2 : même si d'un point de vue théorique, le problème ci-dessus existe réellement, dans la pratique, vu les choix faits au niveau de la MSLIB Fortran 90, cela devrait généralement correspondre aux besoins des utilisateurs.

4 Documentation utilisateur : Comment utiliser les notices d'utilisation des routines ?

Chaque routine de la MSLIB Fortran 90 est décrite par les neuf rubriques suivantes :

- identification,
- rôle,
- séquence d'appel,
- description des arguments,
- conditions sur les arguments,
- notes d'utilisation,
- références documentaires,
- code retour,
- exemple en Fortran 90 portable.

4.1 Rubrique "Identification"

Cette rubrique donne de manière succincte le lien entre le nom de la routine et sa fonctionnalité.

Exemple :

Pour la routine "`mt_car_geoc`", la rubrique identification contient une phrase du type : "Calcul des coordonnées **géoc**entriques à partir des coordonnées **car**tésiennes".

4.2 Rubrique "Rôle"

Cette rubrique présente de manière exhaustive le rôle de la routine.

4.3 Rubrique "Séquence d'appel"

Cette rubrique donne une pseudo-séquence d'appel. En effet, les paramètres optionnels sont mis entre crochets et ils ne sont pas nommés contrairement à ce qu'on conseille de faire aux utilisateurs.

Par contre, l'ordre de présentation des paramètres au sein de la liste d'appel est respecté pour les paramètres obligatoires.

Une séquence d'appel juste et complète est fournie à titre d'exemple dans la rubrique "Exemple en Fortran 90 portable".

Cependant, un utilisateur initié à l'utilisation de la MSLIB Fortran 90 pourra utiliser uniquement cette pseudo-séquence d'appel, cette dernière ayant pour principal intérêt de fournir de manière synthétique toutes les informations utiles.

Exemple :

Pour la routine "`mu_norme`", la pseudo-séquence d'appel est :

```
call mu_norme(vect,norme,code_retour[,vect_norme])
```

4.4 Rubrique "Description des arguments"

Les arguments sont décrits dans l'ordre de présentation au sein de la liste d'appel. Compte-tenu des règles de programmation utilisées pour la MSLIB Fortran 90, ils sont rangés dans l'une des quatre sous-rubriques suivantes :

- "entrées obligatoires",
- "sorties obligatoires",
- "entrées facultatives",
- "sorties facultatives".

Chaque argument est décrit à l'aide de trois colonnes :

- la première colonne contient le type informatique de l'argument,
- la deuxième colonne donne en caractère gras le nom informatique de l'argument,
- et la troisième colonne donne le rôle de l'argument, son nom mathématique ou physique (a, e, i, ω, Ω, ..., π, ...) et son unité (unité SI en général).

Voici quelques exemples de type informatique avec leur signification :

`pm_reel`	:	réel de sous-type MSLIB `pm_reel`.
`pm_reel(3)`	:	vecteur de réels (de sous-type MSLIB `pm_reel`) de dimension 3 (avec des indices variant de 1 à 3).
`pm_reel(2:6)`	:	vecteur de réels (de sous-type MSLIB `pm_reel`) de dimension 5 avec des indices variant de 2 à 6.
`pm_reel(6,6)`	:	tableau de réels (de sous-type MSLIB `pm_reel`) de dimension (6, 6).
`entier`	:	entier avec le sous-type par défaut.
`entier(2)`	:	vecteur d'entiers (avec le sous-type par défaut) de dimension 2.
`pm_entier`	:	entier de sous-type MSLIB `pm_entier`.
`tm_code_retour`	:	structure de données de type `tm_code_retour` (voir paragraphe 2.3.2 pour la définition).
`tm_geodesique`	:	structure de données de type `tm_geodesique` (voir paragraphe 2.3.2 pour la définition).
`tm_sgd`	:	structure de données de type `tm_sgd` (voir paragraphe 2.3.2 pour la définition).
`character(len=80)`	:	chaîne de caractères de 80 éléments.

etc...

Rappel :

- Toutes les structures de données sont présentées et définies dans le paragraphe 2.3.2.
- Le sous-type MSLIB `pm_reel` et le sous-type MSLIB `pm_entier` sont présentés et définies dans le paragraphe 2.2.1.

Les unités sont données à titre indicatif et elles constituent un ensemble d'unités cohérentes pour la routine. L'utilisateur peut en déduire un autre ensemble d'unités cohérentes.

Exemple :

Pour la routine "`mt_car_geoc`", le contenu de la rubrique "Description des arguments" est le suivant :

- Entrées obligatoires

`pm_reel(3)`	**`pos_car`**	Position en coordonnées cartésiennes x, y, z (m)

- Sorties obligatoires

`tm_geocentrique`	**`pos_geoc`**	Position en coordonnées géocentriques latitude/longitude/distance centre Terre (rad, m)
`tm_code_retour`	**`code_retour`**	

Nota : on remarquera que pour alléger l'écriture, on n'a pas mis la troisième colonne pour l'argument code_retour. Si elle y était, son contenu serait du type : "Code retour pour la gestion des éventuelles anomalies".

4.5 Rubrique "Conditions sur les arguments"

Les conditions nécessaires sur les arguments d'entrée sont précisées (fourchette de valeurs, valeurs spécifiques, codes particuliers et leurs significations, ...). De la même façon, sont fournies les informations utiles relatives aux paramètres de sortie sachant que les valeurs du code retour sont exprimées à une autre rubrique.

Enfin, dans cette rubrique, on indique pour chaque paramètre de sortie facultatif, la liste des paramètres d'entrées facultatifs qui sont indispensables pour pouvoir effectuer le calcul.

4.6 Rubrique "Notes d'utilisation"

Toute information nécessaire à une bonne utilisation de la routine est fournie, comme par exemple les noms des autres routines qui doivent (ou peuvent) être appelées avant l'appel de la routine que l'on décrit (pour définir les valeurs de certains paramètres).

Enfin, s'il existe une ou plusieurs restrictions d'utilisation, elles sont exprimées de façon claire et non ambiguë.

D'autre part, cette rubrique permet de signaler certaines caractéristiques de la routine quand celles-ci divergent par rapport aux caractéristiques globales de la bibliothèque et/ou quand il est important que l'utilisateur en soit averti. (Il peut s'agir par exemple de temps d'exécution, de la précision des calculs, ...).

4.7 Rubrique "Références documentaires"

Cette rubrique donne les références documentaires qui concernent la routine et son algorithme.

Quelle que soit l'origine de ces références, elles sont archivées au niveau de la gestion de configuration de la documentation du projet MSLIB.

Leur nomenclature MSLIB figure dans cette rubrique afin qu'un utilisateur intéressé puisse facilement accéder à ces références (voir le chapitre 5).

4.8 Rubrique "Code retour"

Cette rubrique donne la liste des différentes valeurs possibles du champ "`valeur`" du code retour.

Pour chaque valeur du champ "`valeur`" du code retour, on donne le nom du "parameter MSLIB" qui contient cette valeur, ainsi que la signification mathématique ou physique.

Rappel :

- La gestion des anomalies est présentée au chapitre 2.4.
- Dans le chapitre 3, nous donnons quelques conseils concernant notamment la façon de traiter les codes retours (voir conseils Util(4), Util(5), Util(6), Util(7)).

4.9 Rubrique "Exemple en Fortran 90 portable"

Cette rubrique donne un exemple complet en Fortran 90. Elle contient une partie "code source" et une partie "résultats attendus". La partie "code source" et la partie "résultats attendus" sont différenciées par des fontes de caractères différentes.

Pour faciliter la compréhension de l'exemple, on fait comme si l'exemple avait été écrit par un programmeur qui utilisait des lettres minuscules pour tous les mots réservés (ceux du langage Fortran 90 et ceux de la MSLIB Fortran 90) et des lettres majuscules pour tous les mots propres à ce programmeur. Cette méthode permet de faire la différence entre les mots que vous êtes pratiquement obligés d'utiliser ("`call`", "`use`", "`mslib`", ...) et ceux que vous devez définir vous-même (vos variables, vos noms de routines, ...).

On remarquera que le numéro d'identification de la routine figure toujours à la dernière ligne de l'exemple (la définition du numéro d'identification est donnée au chapitre 2.4 "Gestion des anomalies").

Exemple :

```
program DEMONSTRATION
  use mslib
  real(pm_reel)        ::POS_CAR(3)
  type(tm_geocentrique)::POS_GEOC
  type(tm_code_retour) ::CODE_RETOUR
```

```
    POS_CAR(1) = 42 356 421.57_pm_reel
    POS_CAR(2) = 0.
    POS_CAR(3) = 0.
    call mt_car_geoc(POS_CAR,POS_GEOC,CODE_RETOUR)
    ! appel a la routine utilisateur d'ecriture des resultats
    call WRITE_RESULTATS(POS_GEOC,CODE_RETOUR)
  end program DEMONSTRATION
```

Résultats attendus :

POS_GEOC % lat	= 0.
POS_GEOC % long	= 0.
POS_GEOC % dist	= 4.24 10^{+7} (valeur arrondie au 3ème chiffre significatif)
CODE_RETOUR%valeur	= 0
CODE_RETOUR%routine	= 1050

Nota : dans l'exemple ci-dessus, on a écrit "`42356421.57_pm_reel`". Le choix de ce type d'écriture a été fait pour garantir la portabilité. Si on avait écrit quelque chose du type "`42356421.57`" ou "`42356421.57d0`", il serait impossible de garantir cette portabilité.

5 La documentation technique du projet MSLIB disponible pour les utilisateurs

La documentation du projet MSLIB est gérée par le Service de Gestion de la Documentation de la sous-direction MPI du CNES.

Une partie de cette documentation est accessible à tous les utilisateurs de la MSLIB Fortran 90.

Il s’agit des documents suivants :

- publication, article de recherche, notes techniques concernant les algorithmes utilisés par les routines de la MSLIB. Ces documents sont cités dans la rubrique "références documentaires" des notices d’utilisation des routines ;
- "Spécification de la gestion des anomalies pour la MSLIB Fortran 90" (M-ST-0-79-CN) ;
- la documentation utilisateur de la MSLIB Fortran 90 (M-MU-0-100-CN).

Tous ces documents portent une nomenclature MSLIB et ceci quel que soit leur origine.

Si vous êtes intéressés par l’un de ces documents, vous devez contacter la personne chargée de la diffusion de la documentation MSLIB en lui précisant la nomenclature du document désiré (le nom et les coordonnées de cette personne sont donnés dans l’annexe 4 du volume 2 de la documentation utilisateur M-MU-0-102-CN).

Annexe 1
Le vocabulaire MSLIB

Dans cette annexe, nous donnons l'équivalence entre les appellations utilisées dans le langage courant et les noms qui sont utilisés dans la MSLIB Fortran 90, que ce soit pour les noms des routines, les noms des paramètres d'appels, les noms des parameters ou encore les noms des types dérivés. Pour faciliter la recherche d'information, nous avons fait un classement par thème.

Constantes mathématiques ou physiques

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`mu`	μ
`r_equa`	Rayon équatorial
`apla`	Aplatissement
`inv_apla`	Inverse de l'aplatissement
`vit_rot`	Vitesse de rotation
`vit_rot_terre`	Vitesse de rotation de la Terre
`C20`	Coefficient C_{20}
`cn0`	Coefficients C_{20} à C_{50} par exemple
`pi`	π

Date

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`an`	Année
`mois`	Mois
`jour`	Jour
`heure`	Heure
`min`	Minute
`sec`	Seconde
`jul1950`	Jours Juliens 1950 et secondes dans le jour

Les repères

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`rep`	Repère
`qsw`	Repère orbital local ($\vec{q}$, $\vec{s}$, $\vec{w}$)
`tnw`	Repère orbital local ($\vec{t}$, $\vec{n}$, $\vec{w}$)
`iner`	inertiel (repère - ; exemple : "H0-n")
`ref`	terrestre de référence (repère -)
`topo`	topocentrique (repère -)
`moy`	moyen (repère -)
`vrai`	vrai (repère -)
`equa`	équatorial (repère -)
`equaUAI`	équatorial planétaire UAI (repère -)
`ecli`	écliptique (repère -)
`ter`	terrestre (repère -)
`pla, planet`	planétocentrique (repère -)
`BBR`	Body Body Rotating (repère -)
`J2000`	J2000 ou EME2000 (repère -)

Position, vitesse, paramètres orbitaux

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`cir_equa`	a, $\tilde{e}_x$, $\tilde{e}_y$, i_x, i_y, $\omega + \Omega + M$
`cir`	a, e_x, e_y, i, Ω, $\omega + M$
`equa`	a, e, $\omega + \Omega$, i_x, i_y, M *(ou p, e,* $\omega + \Omega$, i_x, i_y, M pour la parabole)
`kep`	a, e, i, ω, Ω, M *(ou p, e, i,* ω, Ω, M pour la parabole)
`a`	a (demi grand axe)

Position, vitesse, paramètres orbitaux

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`p`	*p* (paramètre de la parabole)
`e`	*e* (excentricité)
`i`	*i* (inclinaison)
`pom`	ω
`gom`	Ω
`M, anom_M`	*M* (anomalie moyenne)
`anom_E`	*E* (anomalie excentrique)
`anom_v, v`	*v* (anomalie vraie)
`anom`	anomalie
`ex`	e_x, $\tilde{e}_x$
`ey`	e_y, $\tilde{e}_y$
`ix`	i_x
`iy`	i_y
`ixiy`	Vecteur inclinaison (i_x, i_y)
`pgom`	ω + Ω
`pso_v`	ω + v
`pso_M`	ω + M
`pso_M`	ω + Ω + M
`pso_E`	ω + E
`pso_E`	ω + Ω + E
`lat`	Latitude
`long`	Longitude
`haut`	Hauteur
`alt`	Altitude
`pos`	Position
`vit`	Vitesse
`pos_car`	Position en coordonnées cartésiennes
`vit_car`	Vitesse en coordonnées cartésiennes

Position, vitesse, paramètres orbitaux

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`pos_sgd`	Position en coordonnées site/gisement/distance
`vit_sgd`	Vitesse en coordonnées site/gisement/distance
`pos_geod`	Position en coordonnées géodésiques
`pos_geoc`	Position en coordonnées géocentriques
`asc_droite`	Ascension droite
`declinaison`	Déclinaison
`osc`	Paramètres osculateurs
`moy`	Paramètres moyens

Le vocabulaire mathématique

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`sup`	Supérieur
`inf`	Inférieur
`interval`	Intervalle
`oz`	Axe O_z
`orig`	Origine
`deriv`	Dérive, dérivée
`conv`	Convergence
`mat`	Matrice
`vect`	Vecteur
`dir`	Direction
`dist`	Distance
`quat`	Quaternion
`conjug`	Conjugué (pour un quaternion)
`rot`	Rotation
`prod`	Produit (multiplication)

Gestion des anomalies

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`code_retour`	Code retour
`err`	Erreur
`warn`	Warning, avertissement

Divers : le vocabulaire mécanique spatiale

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`UAI`	Union Astronomique Internationale
`sol`	Soleil
`pla`	Planète
`nuta`	Nutation
`prec`	Précession
`obli, obliquite`	Obliquité
`tsid`	Temps sidéral
`circul`	Circulaire
`ellip`	Ellipse, ellipsoïde
`hyperb`	Hyperbole
`parab`	Parabole
`moy`	(Paramètre) moyen
`osc`	(Paramètre) osculateur
`equa`	Equatorial

Divers : le vocabulaire général

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`mat`	Matrice

Divers : le vocabulaire général

Nom utilisé dans la MSLIB Fortran 90	Appellation dans le langage courant
`jacob`	Jacobienne
`model`	Modèle
`para`	Paramètre
`ind`	Indicateur
`trsf`	Transformation
`val`	Valeur
`valid`	Validation
`gene`	Général
`fon`	Fondamental
`frac`	Fractionnaire
`comp, compar`	Comparaison
`conv`	Conversion
`admi`	Admissible
`def`	Définition
`indef`	Indéfini
`impul`	Impulsion
`eps`	Epsilon

Annexe 2
Inventaire des causes possibles d'anomalies

Dans cette annexe, nous donnons l'inventaire des causes possibles d'anomalies liées à l'utilisation de la MSLIB Fortran 90. Toutes les causes sont classées par thème ;

Au sein de chaque thème, pour chaque cause possible d'anomalie, nous donnons :

- la valeur du champ "valeur" du code retour,
- le nom du "parameter MSLIB' qui contient cette valeur,
- la signification mathématique, physique ou fonctionnelle.

Cas nominal:	
pm_OK	(0) : Retour normal.

Problème sur la valeur d'une constante physique **[+/- 1001, +/- 1099]:**	
pm_err_mu_negatif	(-1001) : La constante gravitationnelle est négative.
pm_err_mu_nul	(-1002) : La constante gravitationnelle est proche de 0.
pm_err_apla_sup1	(-1003) : L'aplatissement (f) est supérieur ou égal à 1.
pm_err_apla_negatif	(-1004) : L'aplatissement (f) est négatif.
pm_err_cn0_nul	(-1005) : Un des coefficients zonaux (CN0) est proche de 0.

Problème sur la valeur du demi-grand axe de la conique **[+/- 1101, +/- 1199]:**	
pm_err_a_negatif	(-1101) : Le demi-grand axe (a) ou le paramètre (p) de la parabole est négatif.
pm_err_a_nul	(-1102) : Le demi-grand axe (a) ou le paramètre (p) de la parabole est proche de 0 .
pm_err_a_infini	(-1103) : L'inverse du demi-grand axe (1/a) est proche de 0 (le demi-grand axe est donc infini).

Problème sur la valeur de l'excentricité **[+/- 1201, +/- 1299]:**	
pm_warn_e_circul_i_equa	(+1201) : L'excentricité (e) est proche de 0 et sin(i) est proche de 0; l'orbite est circulaire et équatoriale (i = 0 ou i = pi).
pm_warn_e_grand_eck_hech	(+1202) : L'excentricité (e) est supérieure à 0,005; la précision est dégradée dans les routines de calcul liées au modèle d'extrapolation de ECKSTEIN-HECHLER.
pm_warn_e_faible_brouwer	(+1203) : L'excentricité (e) est inférieure à 0,01 ; la précision est dégradée dans les routines de calcul liées au modèle d'extrapolation de BROUWER.
pm_warn_e_circul	(+1204) : L'excentricité (e) est proche de 0; l'orbite est circulaire.
pm_warn_e_parab	(+1206) : L'excentricité (e) est proche de 1 : les calculs ont été faits en considérant que l'orbite était parabolique.
pm_err_e_negatif	(-1201) : L'excentricité (e) est négative.
pm_err_e_circul	(-1202) : L'excentricité (e) est proche de 0 ; l'orbite est circulaire.
pm_err_e_grand_brouwer	(-1204) : L'excentricité (e) est supérieure à 0,9 ; les routines de calcul liées au modèle d'extrapolation de BROUWER n'autorisent pas ces valeurs de l'excentricité.
pm_err_e_parab	(-1205) : L'excentricité (e) est proche de 1 ; l'orbite est parabolique.
pm_err_e_non_ellip	(-1208) : L'excentricité (e) n'appartient pas à l'intervalle [0, 1 [; l'orbite n'est pas elliptique.
pm_err_e_hyperb	(-1209) : L'excentricité (e) est supérieure à 1 ; l'orbite est hyperbolique.
pm_err_e_grand_eck_hech	(-1210) : L'excentricité (e) est supérieure ou égale à 0,1; les routines de calcul liées au modèle d'extrapolation de ECKSTEIN-HECHLER n'autorisent pas ces valeurs de l'excentricité.
pm_err_e_faible_brouwer	(-1211) : L'excentricité (e) est inférieure à 0,0001; les routines de calcul liées au modèle d'extrapolation de BROUWER n'autorisent pas ces valeurs de l'excentricité.

Problème sur la valeur de l'excentricité **[+/- 1201, +/- 1299]:**	
pm_err_e_faible	(-1212) : L'excentricité (e) est inférieure à 1e-7; les routines de changement de variables car -> kep et car -> equa n'autorisent pas ces valeurs de l'excentricité.
pm_err_jac_non_calc_e_circul	(-1213) : L'orbite est circulaire (e=0). Il en résulte que la jacobienne n'est pas calculable.
pm_err_anom_v_incompatible_e	(-1214) : L'anomalie vraie v est incompatible avec l'excentricité de l'hyperbole: cos(v) < -1/e.

Problème sur la valeur de l'inclinaison i ou sur le vecteur inclinaison (i_x, i_y) **[+/- 1301, +/- 1399]:**	
pm_warn_i_equa	(+1302) : sin(i) est proche de 0; l'orbite est équatoriale (i=0 ou i=pi).
pm_warn_i_faible_brouwer	(+1303) : L'inclinaison (i) est inférieure à 0,018 radian ; la précision est dégradée dans les routines de calcul liées au modèle d'extrapolation de BROUWER.
pm_warn_e_circul_i_equa	se reporter à la plage [+/- 1201 , +/- 1299]
pm_err_i_negatif	(-1301) : L'inclinaison (i) est négative.
pm_err_i_equa	(-1302) : sin(i) est proche de 0 ; l'orbite est équatoriale (i=0 ou i=pi).
pm_err_i_critique	(-1304) : L'inclinaison (i) a une valeur proche d'une des deux valeurs de l'inclinaison critique (pour plus d'informations voir la routine mc_phys du thème "Constantes").
pm_err_i_sup_pi	(-1305) : L'inclinaison (i) est supérieure à pi.
pm_err_ix_iy_sup2	(-1306) : La norme du vecteur inclinaison est trop grande (supérieure à 2).
pm_err_i_equa_retro	(-1307) : L'inclinaison (i) est égale à pi.
pm_err_jac_non_calc_i_equa	(-1308) : L'orbite est équatoriale (i=0 ou pi). Il en résulte que la jacobienne n'est pas calculable.

Problème sur une date, une durée, un temps **[+/- 1401, +/- 1499]:**	
pm_err_an_inf1950	(-1401) : L'année est antérieure à 1950.
pm_err_mois_interval	(-1402) : Le mois n'appartient pas à l'intervalle [1, 12].

Problème sur une date, une durée, un temps **[+/- 1401, +/- 1499]:**	
pm_err_jour_interval	(-1403) : Le jour n'appartient pas à l'intervalle [1, n] avec n = 28, 29, 30 ou 31 selon le mois et l'année.
pm_err_jul1950_negatif	(-1404) : Le nombre de jours juliens 1950 est négatif.
pm_err_duree_negatif	(-1405) : La durée est négative.
pm_err_sec_interval_jour	(-1406) : Le nombre de secondes dans le jour n'appartient pas à l'intervalle [0, 86400[.
pm_err_heure_interval	(-1407) : Le nombre d'heures dans le jour n'appartient pas à l'intervalle [0, 24[.
pm_err_min_interval	(-1408) : Le nombre de minutes dans l'heure n'appartient pas à l'intervalle [0, 60[.
pm_err_sec_interval_min	(-1409) : Le nombre de secondes dans la minute n'appartient pas à l'intervalle [0, 60[.
pm_err_an_sup2099	(-1412) : L'année est postérieure à l'année 2099.
pm_err_jul1950_sup2099	(-1413) : Le nombre de jours juliens 1950 est trop grand (date postérieure au 31 décembre 2099).

Problème sur la position et/ou la vitesse **[+/- 1501, +/- 1599]:**	
pm_warn_pos_Oz_topo	(+1504) : Position sur l'axe *Oz* du repère topocentrique. Il existe donc une infinité de solutions pour la valeur du gisement, et on lui a donné arbitrairement la valeur 0.
pm_warn_alt_negatif	(+1506) : L'altitude est négative. Les calculs ont été effectués en prenant une altitude nulle.
pm_warn_pos_Oz_ref	(+1507) : Position sur l'axe Oz du repère de référence. Il existe donc une infinité de solutions pour la valeur de la longitude, et on lui a donné arbitrairement une valeur.
pm_err_pos_nul	(-1501) : La norme du vecteur position est proche de 0 .
pm_err_vit_nul	(-1502) : La norme du vecteur vitesse est proche de 0 .
pm_err_pos_vit_colineaire	(-1503) : Le produit vectoriel position-vitesse est pratiquement nul (ce qui signifie: position nulle ou vitesse nulle ou vecteurs position et vitesse colinéaires).

Problème sur la position et/ou la vitesse **[+/- 1501, +/- 1599]:**	
pm_err_pos_Oz_topo	(-1504) : Position sur l'axe Oz du repère topocentrique. Il existe donc une infinité de solutions pour la valeur du gisement. Les composantes de la vitesse ainsi que la jacobienne ne sont pas définies.
pm_err_pos_orig_topo	(-1505) : La position est confondue avec l'origine du repère topocentrique.
pm_err_alt_sup1000km	(-1507) : L'altitude est supérieure à 1000 km ; or le modèle d'atmosphère US76 n'est pas applicable à des altitudes supérieures à 1000 km.
pm_err_orig_topo_centre_terre	(-1508) : L'origine du repère topocentrique est au centre de la Terre.
pm_err_pos_Oz_ref	(-1509) : Position sur l'axe Oz du repère de référence. Il existe donc une infinité de solutions pour la valeur de la longitude. Les composantes de la vitesse ainsi que la jacobienne ne sont pas définies.
pm_err_jac_non_calc_poles	(-1510) : La position de référence se situe aux pôles, il en résulte que la jacobienne n'est pas calculable.
pm_err_jac_non_calc_alt_neg	(-1511) : L'altitude est négative, il en résulte que la jacobienne n'est pas calculable.
pm_err_meme_planete	(-1512) : Les planètes sont confondues, le repère BBR est donc indéfini.

Problème sur la valeur d'une variable physique **[+/- 1601, +/- 1699]:**	
sans objet	sans objet

Problème de codage dans le code de l'utilisateur **[+/- 1801, +/- 1898]:**	
pm_warn_para_option	(+1801): Manque de cohérence entre les entrées optionnelles fournies et les sorties optionnelles demandées: - soit il manque une ou plusieurs sorties optionnelles compte tenu des entrées optionnelles fournies, - soit une entrée optionnelle a été fournie inutilement. Vérifier votre séquence d'appel.
pm_warn_conv_identite	(+1802): Conversion identité: les types d'anomalie en entrée et en sortie sont les mêmes.

Problème de codage dans le code de l'utilisateur **[+/- 1801, +/- 1898]:**	
pm_err_para_option	(-1801) : Compte tenu des sorties optionnelles demandées, il manque des entrées optionnelles.
pm_err_ind_nuta	(-1802) : La valeur donnée pour l'indicateur du modèle de nutation est incorrecte.
pm_err_ind_prec	(-1803) : La valeur donnée pour l'indicateur du modèle de précession est incorrecte.
pm_err_ind_model	(-1804) : La valeur donnée pour l'indicateur du modèle est incorrecte.
pm_err_ind_trsf	(-1805) : La valeur donnée pour l'indicateur de la transformation est incorrecte.
pm_err_val_code_retour_inconnu	(-1806) : Cette valeur de code retour est inconnue dans la bibliothèque.
pm_err_numero_routine_inconnu	(-1807) : Cette valeur de numéro de routine est inconnue dans la bibliothèque.
pm_err_ind_rep	(-1808) : La valeur donnée pour l'indicateur du repère est incorrecte.
pm_err_planete	(-1809) : La valeur donnée pour l'indicateur de l'astre/planète est incorrecte.
pm_err_clef_rot	(-1810) : La clef de rotation ne correspond pas à une rotation de Cardan ou d'Euler.
pm_err_type_anom	(-1811) : La valeur donnée pour le type d'anomalie est incorrecte.

Problème de convergence dans des routines de mécanique spatiale **[+/- 1901, +/- 1998]:**	
pm_err_conv_car_geod	(-1901) : L'algorithme itératif utilisé pour le calcul des coordonnées géodésiques à partir des coordonnées cartésiennes n'a pas réussi à converger vers la bonne solution. Contacter l'assistance utilisateur MSLIB.
pm_err_conv_kepler_ellip	(-1902) : L'algorithme itératif utilisé pour la résolution de l'équation de Kepler (orbite elliptique) n'a pas réussi à converger vers la bonne solution. Contacter l'assistance utilisateur MSLIB.

Problème de convergence dans des routines de mécanique spatiale **[+/- 1901, +/- 1998]:**	
pm_err_conv_kepler_hyperb	(-1903) : L'algorithme itératif utilisé pour la résolution de l'équation de Kepler (orbite hyperbolique) n'a pas réussi à converger vers la bonne solution. Contacter l'assistance utilisateur MSLIB.
pm_err_conv_kepler_gene	(-1904) : L'algorithme itératif utilisé pour la résolution de l'équation de Kepler généralisée (équation avec le vecteur excentricité) n'a pas réussi à converger vers la bonne solution. Contacter l'assistance utilisateur MSLIB.
pm_err_conv_brouwer	(-1905) : L'algorithme itératif utilisé pour le calcul des paramètres moyens pour le modèle de Brouwer n'a pas réussi à converger vers la bonne solution. Vous pouvez essayer d'autres valeurs pour les écarts admissibles, ou bien contacter l'assistance utilisateur MSLIB.
pm_err_conv_eck_hech	(-1906) : L'algorithme itératif utilisé pour le calcul des paramètres moyens pour le modèle de Eckstein-Hechler n'a pas réussi à converger vers la bonne solution. Vous pouvez essayer d'autres valeurs pour les écarts admissibles, ou bien contacter l'assistance utilisateur MSLIB.
pm_err_conv_lyddane	(-1907) : L'algorithme itératif utilisé pour le calcul des paramètres moyens pour le modèle de LYDDANE n'a pas réussi à converger vers la bonne solution. Contactez l'assistance utilisateur MSLIB.

Problème ayant une cause non identifiée à ce jour dans une routine de mécanique spatiale **[-1999]:**	
pm_err_cni	(-1999) : Problème numérique. Contacter l'assistance utilisateur MSLIB.

Problème mathématique **[+/- 2001, +/- 2999]:**	
pm_warn_angle1_ou_3_indef	(+2001) :Infinité de solutions pour le premier et le troisième angle. Arbitrairement nous donnons la valeur 0 au premier angle pour le cas de Cardan ou au troisième angle pour le cas d'Euler.

Problème mathématique **[+/- 2001, +/- 2999]:**	
pm_err_vect_nul	(-2001) : Vecteur nul .
pm_err_eps_negatif	(-2002) : La valeur de l'epsilon en entrée est négative.
pm_err_eps_nul	(-2003) : La valeur de l'epsilon en entrée est nulle.
pm_err_axe_rot_nul	(-2004) : La norme de l'axe de rotation est nulle.
pm_err_quat_nul	(-2005) : La norme du quaternion est nulle.
pm_err_axe_rot_indef	(-2006) : La première composante du quaternion normé vaut 1 ou -1: l'axe de rotation est indéfini.
pm_err_mat_non_rot	(-2007) : La matrice n'est pas une matrice de rotation.