Utilisation de la bibliothèque NAG

O. Louisnard

Juin 1995

1 Introduction

NAG (Numerical Algorithm Group) est une bibliothèque de subroutines FORTRAN susceptible de résoudre la plupart des problèmes d'analyse numérique courants. Elle comporte environ 3000 subroutines.

Les domaines traités sont (liste non exhaustive):

- interpolation
- intégration
- équations différentielles et algébro-différentielles
- équations intégrales
- sommation de séries
- fonctions spéciales
- inversion de systèmes linéaires et de matrices
- calcul de valeurs propres
- optimisation

Il existe d'autres bibliothèques du même type sur le marché (IMSL), mais NAG semble être à ce jour la plus complète. Nous en sommes actuellement à la version 16, ce qui signifie que la plupart des subroutines ont été corrigées et améliorées suite aux remarques des utilisateurs.

La bibliothèque NAG est vendue au tarif universitaire pour environ 5000 F. L'utilisateur dispose des sources des subroutines et d'une documentation complète.

Avertissement : l'utilisation de la bibliothèque NAG suppose que vous maitrisez bien le langage FORTRAN.

2 Nomenclature

Les subroutines portent des noms génériques de 6 caractères. Les trois premiers caractères (une lettre suivie de 2 chiffres) définissent le type de problème traité. Ainsi toutes les subroutines commençant par **D02** traitent de la résolution d'équas diff, toutes celles commençant par **E04** traitent de l'optimisation. La liste complète des sections est présentée en annexe A.

Les trois dernières lettres ne signifient rien et servent simplement à différencier les subroutines d'une même famille.

On peut s'étonner du nombre impressionnant de subroutines disponibles par rapport au nombre somme toute limité de problèmes traités. En fait, pour un problème donné, par exemple la résolution des systèmes linéaires, il y a déjà plusieurs méthodes envisageables: Gauss, Choleski, Householder, ...

De plus, dans un souci d'efficacité, il faut bien comprendre que l'on ne traitera pas de la même manière une matrice pleine, multi-diagonale, creuse, etc... De manière générale, NAG vous guidera vers la subroutine qui est à priori la mieux adaptée à votre problème.

Cela sous-entend qu'il y a un langage commun entre vous et NAG, et que vous avez quelques notions théoriques sur le problème que vous tentez de résoudre. **Il serait faux de croire que l'accès à une bibliothèque de calcul numérique vous dispense d'en connaître les fondements**.

3 Documentation

3.1 Découpage général

La documentation se présente sous la forme de 10 classeurs.

En gros, il s'agit d'un dictionnaire en 10 volumes, où l'on retrouve les sections classées par ordre alphabétique. Chaque section est repérée par un intercalaire comportant les trois caractères d'identification et le type de problème traité. Exemple:

```
D02 Ordinary differential equations
```

Les deux premières section (dans le volume 1) sont un peu particulières:

Introduction donne le contenu général de la librairie (cf. annexe A) et différentes informations sur la librairie et sa documentation. On y trouve également la liste des subroutines qui ont disparu par rapport à la version précédente, et enfin la liste exhaustive de toutes les subroutines avec une brève description pour chacune.

Index contient essentiellement une liste des subroutines classées par rapport à une liste de mots-clés appartenant au jargon (anglais) de l'analyse numérique.

Pour chercher une subroutine, commencez par chercher la section se rapportant à votre problème dans ces deux parties, ou bien plus simplement consultez l'annexe A de ce document. Par exemple, si vous cherchez à résoudre une équa diff, vous allez tomber sur la section D02.

Cherchez ensuite le classeur contenant la section D02 (les sections sont indiquées sur la tranche de chaque classeur, un peu à la manière d'une encyclopédie) et ouvrez-le à la première page de la section.

3.2 Structure des sections

Chaque section commence par quelques considérations théoriques sur le problème traité; on vous indique en général quelles sont les méthodes utilisées, et les différents problèmes qui peuvent se poser. Il est en général intéressant de consulter cette rubrique après coup, lorsque l'utilisation d'une subroutine a échoué.

On trouve ensuite une liste de toutes les subroutines de la section et leurs spécificités; c'est là que se pose le problème crucial: quelle subroutine choisir ? Pour effectuer ce choix difficile, la documentation vous offre parfois un "arbre de décision", qui par un jeu de questions judicieuses, vous dirige tout droit vers la subroutine adaptée à votre problème.

Ce cap difficile étant franchi, il vous reste à vous reporter à la subroutine considérée dans la section (elles sont classées par ordre alphabétique), et à lire attentivement la description de la subroutine.

3.3 Description des subroutines

Pour illustrer ce paragraphe, la description de la subroutine F04ARF (résolution des systèmes linéaires) est fournie en annexe B. La description des subroutines suit toujours le même plan:

1. **Purpose** : indique en quelques lignes ce que fait exactement la subroutine.

2. **Specification** : indique l'en-tête de la subroutine telle que le programmeur l'a écrit. On y trouve donc la liste des arguments de la subroutine et leur déclaration. Cette rubrique définit entièrement le type des variables que vous devez passer à la subroutine. ATTENTION: lorsque vous voyez une variable déclarée de type *real* (en italique) dans la doc NAG, convertissez en **double precision**. Cela vient du fait que le type réel est variable d'une machine à l'autre, et que sur SUN, la bibliothèque est compilée en supposant tous les réels codés en double precision.

3. **Description** indique la méthode employée par la subroutine pour résoudre le problème.

4. **References** vous renvoie à des références bibliographiques sur le sujet.

5. **Parameters** c'est la partie la plus importante : il s'agit d'une description des arguments de la subroutine, dans l'ordre d'apparition.

   On peut considérer une subroutine NAG comme une boite noire avec des points d'entrée et de sortie. L'utilisateur rentre ses données par les points d'entrée, et récupère le résultat du calcul par les points de sortie.

   Rappelons brièvement qu'en FORTRAN, tous les arguments d'une subroutine sont systématiquement des paramètres d'entrée-sortie; c'est au programmeur de la subroutine de faire attention à ne pas modifier un argument si celui-ci doit être considéré comme un paramètre d'entrée.

   La documentation NAG vous indique donc pour chaque argument de subroutine s'il s'agit d'une variable d'entrée, qualifiée de "*Input* ", d'une variable de sortie, "*Output* ", ou bien les deux.

   De plus, il arrive que la subroutine ait besoin de tableaux de travail pour sa cuisine interne; la plupart du temps, elle demandera à l'utilisateur de déclarer ces tableaux et de lui envoyer . Ils sont qualifiés de "*Workspace* ", et ne nécessitent aucune initialisation par l'utilisateur. L'argument WKSPCE de la subroutine F04ARF (cf. annexe B) en est un exemple.

   Enfin, le FORTRAN autorise le passage de noms de fonctions ou de subroutines comme argument d'une subroutine, et ce type d'argument est qualifié par NAG de "*External* ". Lorsq'une subroutine NAG appelle en interne une fonction ou une subroutine dont le nom lui est passé en argument, la documentation vous impose bien sûr le prototype de la fonction ou la subroutine. Un bon exemple est l'argument FCN dans la subroutine D02BAF de résolution d'équas diffs.

   Les arguments d'une subroutine NAG sont donc toujours définis par une carte de la forme:

NOM – TYPE [*Input, Output, Workspace, External*] *On entry* : Ce que vous devez placer dans le paramètre si c'est un paramètre d'entrée. *On exit* : Ce que vous récupérez en sortie s'il s'agit d'un paramètre de sortie. *Constraint* : Contrainte sur la valeur du paramètre d'entrée : doit être supérieur ou inférieur à quelque chose.

Le nom est un nom générique; il va de soi que vous pouvez utiliser le nom de votre choix, du moment que le type est correct.

Dans le cas d'une variable de type vecteur, le nom est suivi par (*) et dans le cas d'une matrice, le nom est suivi par (IA,*), où IA est un autre argument de la subroutine, par lequel il faut passer la première dimension avec laquelle la matrice a été déclarée. Ces notations correspondent aux règles FORTRAN de déclaration des arguments de subroutine de type tableau.

Il faut prendre garde aux arguments de type *Input / Output*, car ils correspondent à des données qui sont modifiés par la subroutine. Un exemple flagrant est l'argument A de la subroutine F04ARF qui résoud les systèmes linéaires. En entrée, ce paramètre doit contenir la matrice du système, et en sortie la subroutine renvoie les matrices L et U de la décomposition dans ce même argument. Si l'utilisateur n'a pas pris soin de sauvegarder sa matrice, elle est perdue !

6. **Error Indicators and Warnings** : On peut constater que presque toutes les subroutines NAG présentent un argument appelé IFAIL, de type *Input / Output*. Ce paramètre permet à NAG de vous indiquer si l'appel de la subroutine s'est bien passé ou non.

   Lorsqu'en sortie IFAIL contient 0, c'est bon. Si IFAIL contient autre chose, cette rubrique vous permet de savoir pourquoi la subroutine a échoué dans le travail qui lui était demandé. Il arrive que l'interprétation d'une erreur nécessite de nombreuses explications théoriques, auquel cas cette rubrique peut être assez fournie.

   IFAIL est aussi un paramètre d'entrée: si vous l'initialisez à 0, alors en cas d'erreur, votre programme s'arrêtera brutalement à l'inteérieur de la subroutine NAG, avec un message d'erreur en anglais, incluant notamment "IFAIL=une valeur". Cela signifie que vous ne pouvez pas reprendre la main dans votre programme. Cette solution est idéale pour le débutant.

   Si vous l'initialisez à 1, alors en cas d'erreur, il n'y aura pas d'arrêt du programme; on ressort normalement de la subroutine, IFAIL contenant une valeur différente de 0, mais le travail n'a pas été fait. C'est au programmeur de tester la valeur de IFAIL à la sortie de la subroutine, et d'agir en conséquence. Cette solution permet d'écrire des programmes qui traitent les erreurs NAG de manière intelligente, dans un but de convivialité.

7. Accuracy : vous donne des informations sur la précision obtenue.

8. Further Comments : tout le bla-bla qui ne rentre pas dans les autres rubriques. C'est souvent intéressant !!

9. Example : un exemple d'utilisation de la subroutine écrit de A à Z. Bien pratique si vous n'avez rien compris à ce qu'il fallait mettre dans les variables d'entrée ...

4 Compilation

Les subroutines NAG sont déjà compilées et archivées dans la librairie statique `/usr/local/nag/nagfl15df/libnag.a` . Pour un compte installé de manière standard à l'EMAC, ce chemin est déjà enregistré dans la variable d'environnement `LD_LIBRARY_PATH`, et il vous suffit d'ajouter "`-lnag`" à la fin de votre commande de link.

Ainsi, si votre programme tient dans un seul fichier appelé `monprog.f`, il vous suffira d'entrer la ligne:

```
f77 monprog.f -o monprog -lnag
```

Si vous obtenez le message:

```
ld: -lnag: No such file or directory
```

demandez à l'ingénieur système qu'il inclue le chemin `/usr/local/nag/nagfl15df` dans votre variable d'environnement `LD_LIBRARY_PATH`, ou bien faites le vous même si vous maitrisez ce genre de chose.

Si vous souhaitez jeter un coup d'oeil aux sources, allez dans le directory :

```
/usr/local/nag/nagfl15df/source
```

Si vous cherchez une subroutine commençant par "d", allez dans le sous-directory `d`. Chaque fichier de ce directory contient une subroutine NAG, le nom du fichier étant celui de la subroutine avec en plus un "t" à la fin. Exemple: `D02BBF` est dans le fichier `d02bbft.f`.

Enfin, vous pouvez retrouver les exemples donnés dans la doc dans le directory :

```
/usr/local/nag/nagfl15df/examples/source
```

Les fichiers contenant les programmes-exemples portent le nom de la subroutine avec un "e" à la fin. Ainsi, le fichier contenant l'exemple illustrant `D02BBF` est dans le fichier `d02bbfe.f`.

Annexe A :

Liste des sections

Annexe B :

Description de la subroutine F04ARF