NAG Fortran Library

Essential Introduction

Note: *this document is essential reading for any prospective user of the Library.*

Contents

1 The Library and its Documentation

1.1 Structure of the Library

The NAG Fortran Library is a comprehensive collection of Fortran **routines** for the solution of numerical and statistical problems. The word 'routine' is used to denote 'subroutine' or 'function'.

The Library is divided into **chapters**, each devoted to a branch of numerical analysis or statistics. Each chapter has a three-character name and a title, e.g.,

D01 – Quadrature

Exceptionally, two chapters (Chapters H and S) have one-character names. (The chapters and their names are based on the ACM modified SHARE classification index (see ACM (1960–1976)).)

All documented routines in the Library have six-character names, beginning with the characters of the chapter name, e.g.,

D01AJF

Note that the second and third characters are **digits**, not letters; e.g., 0 is the digit zero, not the letter O. The last letter of each routine name almost always appears as 'F' in the documentation, but may be changed to 'E' in some single precision implementations (see Section 1.6). Chapters C05, D03 and E04 have some routines whose last letter is 'A' rather than 'F' ('B' in some single precision implementations). An 'A' version is always paired with an 'F' routine, the 'A' version being safe to use in a multithreaded environment, but otherwise having identical functionality to the 'F' version.

Chapter F06 (Linear Algebra Support Routines) contains all the Basic Linear Algebra Subprograms, BLAS, with NAG-style names as well as with the actual BLAS names, e.g., F06AAF (SROTG/DROTG). The names in brackets are the equivalent single and double precision BLAS names respectively. Chapter F07 (Linear Equations (LAPACK)) and Chapter F08 (Least-squares and Eigenvalue Problems (LAPACK)) contain routines derived from the LAPACK project. Like the BLAS, these routines have NAG-style names as well as LAPACK names, e.g., F07ADF (SGETRF/DGETRF). Details regarding these alternate names can be found in the relevant Chapter Introductions.

In order to take full advantage of machine-specific versions of BLAS and LAPACK routines provided by some computer hardware vendors, you are encouraged to use the BLAS and LAPACK names (e.g., SROTG/DROTG and SGETRF/DGETRF) rather than the corresponding NAG-style names (e.g., F06AAF and F07ADF) wherever possible in your programs.

1.2 Structure of the Documentation

The **NAG Fortran Library Manual** is the principal printed form of documentation for the NAG Fortran Library. It has the same chapter structure as the Library: each chapter of routines in the Library has a corresponding chapter (of the same name) in the Manual. The chapters occur in alphanumeric order. General introductory documents are placed in Volume 1 of the Manual.

Each chapter consists of the following documents:

Chapter Contents, e.g., Contents – D01;

Chapter Introduction, e.g., Introduction – D01;

Routine Documents, one for each documented routine in the chapter.

A routine document has the same name as the routine which it describes. Within each chapter, routine documents occur in alphanumeric order. For those chapters that have both 'A' and 'F' versions of a routine, the routine descriptions are combined into one routine document. Exceptionally, some chapters (Chapters F06, X01 and X02) do not have individual routine documents; instead, all the routines are described together in the Chapter Introduction. Another exception is Chapter A00, which contains neither a Chapter Introduction nor any routine documents. It does however contain a user-callable support routine that identifies which version of the Library is available at your site (see Section 1.7).

In addition to the full printed Manual, NAG also produces a printed **Introductory Guide**, which contains all the introductory material from the Manual, together with all the Chapter Contents and Chapter Introductions.

1.3 Alternative Forms of Documentation

NAG also provides machine-based documentation. Currently NAG provides online documentation in the form of unlinked PDF files accompanied by an HTML index. Introductory material is provided in both PDF and HTML but the Keywords in Context and GAMS Index are provided in HTML only. It is anticipated that future releases will provide HTML files (taking advantage of technology that is currently being developed, e.g., MathML) for the full manual and fully linked PDF files for all but the Keywords in Context and GAMS Index. The most uptodate version of the online documentation is accessible via the NAG Web site (see Section 4).

1.4 Marks of the Library

Periodically a new **Mark** of the NAG Fortran Library is released: new routines are added, corrections and/or improvements are made to existing routines; occasionally routines are withdrawn if they have been superseded by improved routines.

At each Mark, the documentation of the Library is updated. You must make sure that your documentation has been updated to the same Mark as the Library software that you are using.

Marks are numbered, e.g., 17, 18, 19. The current Mark is 20.

1.5 Implementations of the Library

The NAG Fortran Library is available on many different computer systems. For each distinct system, an **implementation** of the Library is prepared by NAG, e.g., the Sun Solaris implementation. The implementation is distributed to sites as a tested compiled library.

An implementation is usually specific to a range of machines (e.g., the Compaq OpenVMS range); it may also be specific to a particular operating system, Fortran compiler, or compiler option (such as scalar or vector mode or thread safe).

Essentially the same facilities are provided in all implementations of the Library, but, because of differences in arithmetic behaviour and in the compilation system, routines cannot be expected to give identical results on different systems, especially for sensitive numerical problems.

The documentation supports all implementations of the Library, with the help of a few simple conventions, and a small amount of implementation-dependent information, which is published in a separate **Users' Note** for each implementation (see Section 3.4).

1.6 Precision of the Library

The NAG Fortran Library is developed in both **single precision** and **double precision** versions. REAL variables and arrays in the single precision version are replaced by DOUBLE PRECISION variables and arrays in the double precision version.

On most systems only one precision of the Library is available; the precision chosen is that which is considered most suitable in general for numerical computation (double precision on most systems).

On some systems both precisions are provided: in this case, the double precision routines have names ending in 'F' (or 'A') (as in the documentation), and the single precision routines have names ending in 'E' (or 'B'). Thus in Compaq OpenVMS implementations:

D01AJF is a routine in the double precision implementation;

D01AJE is the corresponding routine in the single precision implementation.

Whatever the precision, INTEGER variables (and elements of arrays) always occupy one numeric storage unit, that is the Library is **not** implemented using non-standard integer storage (see ANSI (1978)), e.g., INTEGER*2.

1.7 Library Identification

You must know **which implementation, which precision** and **which Mark** of the Library you are using or intend to use. To find out which implementation, precision and Mark of the Library is available at your site, you can run a program which calls the NAG Library routine A00AAF (or A00AAE in most single

precision implementations). This routine has no parameters; it simply outputs text to the NAG Library advisory message unit (see Section 2.4). An example of the output is:

```
*** Start of NAG Library implementation details ***
Implementation title: Sun(SPARC) Solaris
           Precision: double
        Product Code: FLSOL20DA
                Mark: 20
*** End of NAG Library implementation details ***
```

(The product code can be ignored, except possibly when communicating with NAG; see Section 4.)

1.8 Fortran Language Standards

All routines in the Library conform to the ISO Fortran 95 Standard (ISO (1997)), except for the use of a double precision complex data type (usually COMPLEX*16) in some routines in Fortran 77 compiled double precision implementations of the Library – there is no provision for this data type in the old ANSI Standard Fortran 77 (ANSI (1978)).

Many of the routines in the Library were originally written to conform to the earlier Fortran 66 standard (ANSI (1966)), and their calling sequences may contain a few parameters which are not strictly necessary in Fortran 77.

2 Using the Library

2.1 General Advice

A NAG Fortran Library routine **cannot** be guaranteed to return meaningful results irrespective of the data supplied to it. Care and thought **must** be exercised in:

(a) formulating the problem;

(b) programming the use of library routines;

(c) assessing the significance of the results.

The Foreword to the Manual provides some further discussion of points (a) and (c); the remainder of Section 2 is concerned with (b).

2.2 Programming Advice

The NAG Fortran Library and its documentation are designed on the assumption that you know how to write a calling program in Fortran.

When programming a call to a routine, read the routine document carefully, especially the description of the **Parameters**. This states clearly which parameters must have values assigned to them on entry to the routine, and which return useful values on exit. See Section 3.3 for further guidance.

The most common types of programming error in using the Library are:

incorrect parameters in a call to a Library routine;

calling a double precision implementation of the Library from a single precision program, or vice-versa.

Therefore if a call to a Library routine results in an unexpected error message from the system (or possibly from within the Library), **check** the following:

Has the NAG routine been called with the correct number of parameters?

Do the parameters all have the correct type?

Have all array parameters been dimensioned correctly?

Is your program in the same precision as the NAG Library routines to which your program is being linked?

Have NAG routine names been modified – if necessary – as described in Sections 1.6 and 2.5?

Avoid the use of NAG-type names for your own program units or COMMON blocks: in general, do not use names which contain a three-character NAG chapter name embedded in them; they may clash with the names of an auxiliary routine or COMMON block used by the NAG Library.

2.3 Error Handling and the Parameter IFAIL

NAG Fortran Library routines may detect various kinds of error, failure or warning conditions. Such conditions are handled in a systematic way by the Library. They fall roughly into three classes:

(i) an invalid value of a parameter on entry to a routine;

(ii) a numerical failure during computation (e.g., approximate singularity of a matrix, failure of an iteration to converge);

(iii) a warning that, although the computation has been completed, the results cannot be guaranteed to be completely reliable.

All three classes are handled in the same way by the Library, and are all referred to here simply as 'errors'.

The error-handling mechanism uses the parameter IFAIL, which occurs as the last parameter in the calling sequence of most NAG Library routines. IFAIL serves two purposes:

(i) it allows users to specify what action a Library routine should take if it detects an error;

(ii) it reports the outcome of a call to a Library routine, either 'success' (IFAIL $= 0$) or 'failure' (IFAIL $\neq 0$, with different values indicating different reasons for the failure, as explained in Section 6 of the routine documents).

For the first purpose IFAIL **must** be assigned a value before calling the routine; since IFAIL is reset by the routine, it **must** be passed as a variable, not as an integer constant. Allowed values on entry are:

IFAIL $= 0$: an error message is output, and execution is terminated ('hard failure');

IFAIL $= +1$: execution continues without any error message;

IFAIL $= -1$: an error message is output, and execution continues.

The settings IFAIL $= \pm 1$ are referred to as 'soft failure'.

The safest choice is to set IFAIL to 0, but this is not always convenient: some routines return useful results even though a failure (in some cases merely a warning) is indicated. However, if IFAIL is set to ± 1 on entry, it is **essential** for the program to test its value on exit from the routine, and to take appropriate action.

The specification of IFAIL in Section 5 of a routine document suggests a suitable setting of IFAIL for that routine.

For a full description of the error-handling mechanism, see the P01 Chapter Introduction.

Routines in Chapters F07 and F08 do **not** use the usual error handling mechanism; in order to preserve complete compatibility with LAPACK software, they have a diagnostic output parameter INFO which need not be set before entry. See the relevant Chapter Introduction for further details.

Some routines in Chapter F06 output an error message if an illegal input parameter is detected, then terminate program execution immediately. See the F06 Chapter Introduction for further details.

2.4 Input/output in the Library

Most NAG Library routines perform no output to an external file, except possibly to output an error message. All error messages are written to a logical **error message** unit. This unit number (which is set by default to 6 in most implementations) can be changed by calling the Library routine X04AAF.

Some NAG Library routines may optionally output their final results, or intermediate results to monitor the course of computation. In general, output other than error messages is written to a logical **advisory message** unit. This unit number (which is also set by default to 6 in most implementations) can be changed by calling the Library routine X04ABF. Although it is logically distinct from the error message unit, in practice the two unit numbers may be the same. A few routines in Chapter E04 allow this unit number to be specified directly as an option.

All output from the Library is formatted.

There are only a few Library routines which perform input from an external file. These examples occur in Chapters E04 and H. The unit number of the external file is a parameter to the routine, and all input is formatted.

You must ensure that the relevant Fortran unit numbers are associated with the desired external files, either by an OPEN statement in your calling program, or by operating system commands.

2.5 Auxiliary Routines

In addition to those Library routines which are documented and are intended to be called by users, the Library also contains many auxiliary routines. Details of all the auxiliary routines which are called directly or indirectly by any documented NAG Library routine are supplied to sites in machine-readable form with the Library software.

In general, you need not be concerned with them at all, although you may be made aware of their existence if, for example, you examine a memory map of an executable program which calls NAG routines. The only exception is that when calling some NAG Library routines you may be required or allowed to supply the name of an auxiliary routine from the NAG Library as an external procedure parameter. The routine documents give the necessary details. In such cases, you only need to supply the name of the routine; you **never** need to know details of its parameter list.

NAG auxiliary routines have names which are similar to the name of the documented routine(s) to which they are related, but with last letter 'Z', 'Y', and so on, e.g.,

D01BAZ is an auxiliary routine called by D01BAF.

In a single precision implementation in which the names of documented routines end in 'E', the names of auxiliary routines have their first three and last three characters interchanged, e.g.,

BAZD01 is an auxiliary routine (corresponding to D01BAZ) called by D01BAE.

A few chapters contain auxiliary routines whose names are obtained by adding 50 to the second and third characters of the chapter name. For instance, Chapter E04 has an auxiliary routine with the name E54UCZ.

2.6 Thread Safety

Some implementations of the Library facilitate the use of threads; that is, you can call routines from the Library from within a multithreaded application. Fully thread safe libraries are provided for several platforms — for more information please contact your local Response Centre (see Section 4). See the Thread Safety document for more detailed guidance on using the Library in a multithreaded context. You may also need to refer to the Users' Note for details of whether your implementation of the Library has been compiled in a manner that facilitates the use of threads.

2.7 Calling the Library from Other Languages

In general the NAG Fortran Library can be called from other computer languages (such as C and Visual Basic) provided that appropriate mappings exist between their data types.

NAG has produced C Header Files which comprise of a set of header files indicating the match between C and Fortran data types for various compilers, documentation and examples. The documentation and examples are available from the NAG Web sites (see Section 4).

The Dynamic Link Library (DLL) version can be called in a straightforward manner from Visual Basic, Visual Basic for Applications (Excel), Delphi, C and C++. Guidance on this is provided as part of the NAG Fortran Library DLLs. Further details can be found on the NAG Web sites.

3 Using the Documentation

3.1 Using the Manual

The Manual is designed to serve the following functions:

to give background information about different areas of numerical and statistical computation;

to advise on the choice of the most suitable NAG Library routine or routines to solve a particular problem;

to give all the information needed to call a NAG Library routine correctly from a Fortran program, and to assess the results.

At the beginning of the Manual are some general introductory documents which provide some background and additional information. The Library Contents (a structured list of routines in the Library, by chapter) may help you to find the chapter, and possibly the routine, which you need to solve your problem. Online documentation provides you with a fully linked HTML Keywords in Context (a keyword index to chapters and routines) and GAMS Index (a list of NAG routines classified according to the GAMS scheme).

Having found a likely chapter or routine, you should read the corresponding **Chapter Introduction,** which gives background information about that area of numerical computation, and recommendations on the choice of a routine, including indexes, tables or decision trees.

When you have chosen a routine, you must consult the **routine document**. Each routine document is essentially self-contained (it may, however, contain references to related documents). It includes a description of the method, detailed specifications of each parameter, explanations of each error exit, remarks on accuracy, and (in most cases) an example program to illustrate the use of the routine.

3.2 Structure of Routine Documents

Note that at Mark 17 a new typesetting scheme was used to generate documentation. If you have a Manual which contains pre-Mark 17 routine documents, you will find that it contains older documents which differ in appearance, although the structure is the same.

Note also that at Mark 14 some changes were made to the style and appearance of routine documents. If you have a Manual which contains pre-Mark 14 routine documents, you will find that it contains older documents which differ in style, although they contain essentially the same information. Sections 3.2, 3.3 and 3.5 describe the **new-style** routine documents. Section 3.7 gives some details about the old-style documents.

All routine documents have the same structure, consisting of nine numbered sections:

1. Purpose

2. Specification

3. Description

4. References

5. Parameters (see Section 3.3 below)

6. Error Indicators and Warnings

7. Accuracy

8. Further Comments

9. Example (see Section 3.5 below)

In a few documents (notably Chapter E04) there are a further three sections:

10. Algorithmic Details

11. Optional Parameters

12. Description of Monitoring Information

3.3 Specification of Parameters

Section 5 of each routine document contains the specification of the parameters, in the order of their appearance in the parameter list.

3.3.1 Classification of parameters

Parameters are classified as follows.

Input: you must assign values to these parameters on or before entry to the routine, and these values are unchanged on exit from the routine.

Output: you need not assign values to these parameters on or before entry to the routine; the routine may assign values to them.

Input/Output: you must assign values to these parameters on or before entry to the routine, and the routine may then change these values.

Workspace: array parameters which are used as workspace by the routine. You must supply arrays of the correct type and dimension. In general, you need not be concerned with their contents.

External Procedure: a subroutine or function which must be supplied (e.g., to evaluate an integrand or to print intermediate output). Usually it must be supplied as part of your calling program, in which case its specification includes full details of its parameter list and specifications of its parameters (all enclosed in a box). Its parameters are classified in the same way as those of the Library routine, but because you must write the procedure rather than call it, the significance of the classification is different.

> *Input*: values may be supplied on entry, which your procedure **must not** change.
>
> *Output*: you may or must assign values to these parameters before exit from your procedure.
>
> *Input/Output*: values may be supplied on entry, and you may or must assign values to them before exit from your procedure.

Occasionally, as mentioned in Section 2.5, the procedure can be supplied from the NAG Library, and then you only need to know its name.

User Workspace: array parameters which are passed by the Library routine to an external procedure parameter. They are not used by the routine, but you may use them to pass information between your calling program and the external procedure.

Dummy: a simple variable which is not used by the routine. A variable or constant of the correct type must be supplied, but its value need not be set. (A dummy parameter is usually a parameter which was required by an earlier version of the routine and is retained in the parameter list for compatibility.)

3.3.2 Constraints and suggested values

The word '*Constraint*:' or '*Constraints*:' in the specification of an *Input* parameter introduces a statement of the range of valid values for that parameter, e.g.,

> *Constraint*: $\mathrm{N} > 0$.

If the routine is called with an invalid value for the parameter (e.g., $\mathrm{N} = 0$), the routine will usually take an error exit, returning a non-zero value of IFAIL (see Section 2.3).

In newer routine documents, constraints on parameters of type CHARACTER only list upper case alphabetic characters, e.g.,

> *Constraint*: STRING = 'A' or 'B'.

In practice, all routines with CHARACTER parameters will permit the use of lower case characters.

The phrase '*Suggested Value*:' introduces a suggestion for a reasonable initial setting for an *Input* parameter (e.g., accuracy or maximum number of iterations) in case you are unsure what value to use; you should be prepared to use a different setting if the suggested value turns out to be unsuitable for your problem.

3.3.3 Array parameters

Most array parameters have dimensions which depend on the size of the problem. In Fortran terminology they have 'adjustable dimensions': the dimensions occurring in their declarations are integer variables which are also parameters of the Library routine.

For example, a Library routine might have the specification:

```
SUBROUTINE <name> (M, N, A, B, LDB)
INTEGER            M, N, A(N), B(LDB,N), LDB
```

For a **one-dimensional** array parameter, such as A in this example, the specification would begin

A(N) – INTEGER array

You must ensure that the dimension of the array, as declared in your calling (sub)program, is at least as large as the value you supply for N. It may be larger, but the routine uses only the first N elements.

For a **two-dimensional** array parameter, such as B in the example, the specification might be

B(LDB,N) – INTEGER array

On entry: the m by n matrix B.

and the parameter LDB might be described as follows:

LDB – INTEGER

On entry: the first dimension of the array B as declared in the (sub)program from which <name> is called.

Constraint: LDB $\geq$ M.

You **must** supply the **first** dimension of the array B, as declared in your calling (sub)program, through the parameter LDB, even though the number of rows actually used by the routine is determined by the parameter M. You must ensure that the first dimension of the array is at least as large as the value you supply for M. The extra parameter LDB is needed because Fortran does not allow information about the dimensions of array parameters to be passed automatically to a routine.

You must also ensure that the **second** dimension of the array, as declared in your calling (sub)program, is at least as large as the value you supply for N. It may be larger, but the routine uses only the first N columns.

A program to call the hypothetical routine used as an example in this section might include the statements:

```
INTEGER AA(100), BB(100,50)
LDB = 100
.
.
.
M = 80
N = 20
CALL <name>(M,N,AA,BB,LDB)
```

Fortran 77 requires that the dimensions which occur in array declarations must be greater than zero. Many NAG routines are designed so that they can be called with a parameter like N in the above example set to 0 (in which case they would usually exit immediately without doing anything). If so, the declarations in the Library routine would use the 'assumed size' array dimension, and would be given as

```
INTEGER        M, N, A(*), B(LDB,*), LDB
```

However, the original declaration of an array in your calling program must always have constant dimensions, greater than or equal to 1.

Consult an expert or a textbook on Fortran if you have difficulty in calling NAG routines with array parameters.

3.4 Implementation-dependent Information

In order to support all implementations of the Library, the Manual has adopted a convention of using ***bold italics*** to distinguish terms which have different interpretations in different implementations.

The most important bold italicised terms are the following; their interpretation depends on whether the implementation is in single precision or double precision.

real	means REAL or DOUBLE PRECISION
complex	means COMPLEX or COMPLEX*16 (or equivalent)
basic precision	means single precision or double precision
additional precision	means double precision or quadruple precision

Another important bold italicised term is ***machine precision***, which denotes the relative precision to which ***real*** floating-point numbers are stored in the computer, e.g., in an implementation with approximately 16 decimal digits of precision, ***machine precision*** has a value of approximately 10^{-16}.

The precise value of ***machine precision*** is given by the function X02AJF. Other functions in Chapter X02 return the values of other implementation-dependent constants, such as the overflow threshold, or the largest representable integer. Refer to the X02 Chapter Introduction for more details.

The bold italicised term ***block size*** is used only in Chapters F07 and F08. It denotes the block size used by block algorithms in these chapters. You only need to be aware of its value when it affects the amount of workspace to be supplied – see the parameters WORK and LWORK of the relevant routine documents and the Chapter Introduction.

For each implementation of the Library, a separate **Users' Note** is published. This is a short document, revised at each Mark. At most installations it is available in machine-readable form. It gives any necessary additional information which applies specifically to that implementation, in particular:

the interpretation of bold italicised terms;

the values returned by X02 routines;

the default unit numbers for output (see Section 2.4);

details of name changes for Library routines (see Sections 1.6 and 2.5).

In Chapters F06, F07 and F08, where alternate routine names are available for BLAS and LAPACK derived routines, the alternate name appears in ***bold italics***, for example, ***sgetrf***, which should be interpreted as either SGETRF (in single precision) or DGETRF (in double precision) in the case of F07ADF, which handles real matrices. Similarly, F07ARF for complex matrices uses ***cgetrf***, which should be interpreted as either CGETRF (in single precision) or ZGETRF (in double precision).

3.5 Example Programs and Results

The **example program** in Section 9 of each routine document illustrates a simple call of the routine. The programs are designed so that they can fairly easily be modified, and so serve as the basis for a simple program to solve your problem.

Bold italicised terms are used in the printed text of the example program to denote precision-dependent features in the code. The correct Fortran code must therefore be substituted before the program can be run. In addition to the terms ***real*** and ***complex***, which were explained in Section 3.4, the following terms are used in the example programs:

Intrinsic Functions:	***real***	means REAL or DBLE (see Note below)
	imag	means AIMAG or DIMAG
	cmplx	means CMPLX or DCMPLX
	conjg	means CONJG or DCONJG
Edit Descriptor:	***e***	means E or D (in FORMAT statements)
Exponent Letter:	***e***	means E or D (in constants)

Note that in some implementations the intrinsic function ***real*** with a ***complex*** argument must be interpreted as DREAL rather than DBLE.

The examples in Chapters F07 and F08 use the precision-dependent LAPACK routine names, as mentioned in Section 3.4.

For each implementation of the Library, NAG distributes the example programs in machine-readable form, with all necessary modifications already applied. Many sites make the programs accessible to you in this form. They may also be obtained directly from the NAG Web site.

Note that the results from running the example programs may not be identical in all implementations, and may not agree exactly with the results which are printed in the Manual and which were obtained from a double precision implementation (with approximately 16 digits of precision).

The Users' Note for your implementation will mention any special changes which need to be made to the example programs, and any significant differences in the results.

3.6 Summary for New Users

If you are unfamiliar with the this Library and are thinking of using a (sub)program from it, please follow these instructions: (a) read the whole of the **Essential Introduction**;

(b) consult the **Library Contents** list to select an appropriate chapter or (sub)program;

(c) or search through the **Keywords in Context Index**, **GAMS Index** or via an online search facility;

(d) read the relevant **Chapter Introduction**;

(e) choose a (sub)program, and read the **(sub)program document**. If the (sub)program does not after all meet your needs, return to step (b), (c) or (d);

(f) read the **Users' Note** for your implementation;

(g) consult local documentation, which should be provided by your local support staff, about access to the this Library on your computing system;

(h) obtain an online copy of the example program for the particular (sub)program of interest and experiment with it.

You should now be in a position to include a call to the (sub)program in a program, and to attempt to compile and run it. You may of course need to refer back to the relevant documentation in the case of difficulties, for advice on assessment of results, and so on.

As you become familiar with the Library, some of steps (a) to (h) can be omitted, but it is always essential to: be familiar with the Chapter Introduction;

read the (sub)program document;

be aware of the **Users' Note** for your implementation.

3.7 Pre-Mark 14 Routine Documents

You need only read this section if you have an updated Manual which contains pre-Mark 14 documents.

You will find that older routine documents appear in a somewhat different style, or even several styles if your Manual dates back to Mark 7, say. The following are the most important differences between the earlier styles and the new style introduced at Mark 14.

Before Mark 12, routine documents had 13 sections: the extra sections have either been dropped or merged with the present Section 8 (Further Comments).

In Section 5, parameters were not classified as *Input*, *Output* and so on; the phrase 'Unchanged on exit' was used to indicate an input parameter.

The example programs were revised at Mark 12 and again at Mark 14, to take advantage of features of Fortran 77: the programs printed in older documents do not correspond exactly with those which are now distributed to sites in machine-readable form or available on the NAG Web site.

Before Mark 12, the printed example programs did not use bold italicised terms; they were written in standard single precision Fortran.

Before Mark 9, the printed example results were generated on an ICL 1906A (with approximately 11 digits of precision), and between Marks 9 and 12 they were generated on an ICL 2900 (with approximately 16 digits of precision).

Before Mark 13, documents referred to 'the appropriate implementation document'; this means the same as 'the Users' Note for your implementation'.

4 Support from NAG

(a) Contact with NAG

Queries concerning this library should be directed initially to your local Advisory Service. If you have difficulty in making contact locally, you can contact NAG directly.

(b) NAG Response Centres

The NAG Response Centres are available for general enquiries from all users and also for technical queries from users with Support.

The Response Centres are open during office hours, but contact is possible by fax, email and telephone (answering machine) at all times. Please see the User's Note or the NAG web sites for contact details.

When contacting one of our Response Centres it helps us to deal with you query quickly if you can quote your NAG user reference and NAG product code.

(c) NAG Web Site

The NAG web site is an information service providing items of interest to users and prospective users of NAG products and services. The information is regularly updated and reviewed, and includes implementation availability, descriptions of products, down-loadable software and technical reports. NAG web sites can be accessed at `http://www.nag.co.uk` or

`http://www.nag.com` or

`http://www.naggmbh.de` or

`http://www.nag-j.co.jp`

5 Background to NAG

Various aspects of the design and development of the NAG Library, and NAG's technical policies and organisation are given in Ford (1982), Ford *et al.* (1979), Ford and Pool (1984), and Hague *et al.* (1982).

6 References

Ford B (1982) Transportable numerical software *Lecture Notes in Computer Science* **142** 128–140 Springer-Verlag

Ford B, Bentley J, Du Croz J J and Hague S J (1979) The NAG Library 'machine' *Softw. Pract. Exper.* **9 (1)** 65–72

Ford B and Pool J C T (1984) The evolving NAG Library service *Sources and Development of Mathematical Software* (ed W Cowell) 375–397 Prentice-Hall

Hague S J, Nugent S M and Ford B (1982) Computer-based documentation for the NAG Library *Lecture Notes in Computer Science* **142** 91–127 Springer-Verlag

ACM (1960–1976) Collected algorithms from ACM index by subject to algorithms

ANSI (1966) USA standard Fortran *Publication X3.9* American National Standards Institute

ANSI (1978) American National Standard Fortran *Publication X3.9* American National Standards Institute

ISO (1997) ISO Fortran 95 programming language (ISO/IEC 1539-1:1997)