Conversion from the NAG Fortran 77 Library

This document offers guidance to users who are familiar with the NAG Fortran 77 Library and who now wish to use the NAG Fortran 90 Library, especially users who have Fortran 77 programs which call NAG Fortran 77 Library routines and who wish to convert such programs to Fortran 90 and to call NAG Fortran 90 library procedures.

1 The NAG Fortran 77 and Fortran 90 Libraries

The NAG Fortran 90 Library is a distinct library from the NAG Fortran 77 Library, although it uses essentially the same algorithms. It takes advantage of new features in the Fortran 90 language to offer a simpler, safer and more flexible user interface than the Fortran 77 Library.

This Release of the Fortran 90 Library covers roughly 39% of the algorithms available in Mark 18 of the Fortran 77 Library. You can call both the Fortran 77 and Fortran 90 libraries from a Fortran 90 program (provided that implementations of both libraries are available on your machine); no name-clashes will occur. Therefore, if the algorithm you need is in the Fortran 77 Library but not in the Fortran 90 Library, you can call the Fortran 77 version.

Section 2 summarises the most important general differences between the two libraries, and Section 3 gives specific advice for each chapter of the Fortran 77 Library, principally the names of equivalent Fortran 90 procedures wherever they exist.

Before starting to use the Fortran 90 Library, you should read the **Essential Introduction**, and you may find it helpful to read the **Tutorial**, especially if you are not familiar with the Fortran 90 language.

2 General Differences

This section summarises the most important general differences between the Fortran 77 and Fortran 90 libraries.

Terminology: Some terms used in the documentation for the Fortran 90 Library differ from those used in the documentation for the Fortran 77 Library:

Fortran 77 Library	Fortran 90 Library
parameter	argument
routine	procedure

The documentation for the Fortran 90 Library uses the same terms as the Fortran 90 standard.

Organization into chapters: The Fortran 77 Library is divided into chapters based originally on the ACM Modified SHARE classification scheme (with several subsequent extensions). The Fortran 90 Library is also divided into chapters, but they are not based on the same classification scheme and they are not in one-to-one correspondence with the chapters of the Fortran 77 Library; numerical chapters come before the statistical chapters, and chapters containing simpler algorithms come before those containing more complex algorithms. The chapters are numbered (but the numbers are not all consecutive, because new chapters may be inserted at future Releases).

Modules: In the Fortran 77 Library, each chapter simply contains a number of routines (subroutines or functions). In the Fortran 90 Library, each chapter contains one or more *modules*; each module contains a group of related procedures (subroutines or functions), and some modules also contain type definitions and named constants. Each program unit that refers to a procedure, type definition or named constant from the Fortran 90 Library must also contain a `USE` statement to access the required module; see Section 3 in the **Essential Introduction**.

Structure of the documentation: For the Fortran 77 Library, the documentation for each chapter consists of a **Chapter Introduction** document, a **Chapter Contents** document, and a *routine*

document for each routine. For the Fortran 90 Library, the documentation for each chapter consists of a **Chapter Introduction** document, and a *module document* for each module, including separate sections for each procedure and type definition. The Chapter Introductions are shorter than for the Fortran 77 Library, because much of the information and advice is given in the Introductions to the module documents.

Naming scheme: In the Fortran 77 Library, the names of all routines consist of 6 characters, starting with a 1-character or 3-character chapter code, originally derived from the ACM Modified SHARE classification index, and ending in `F` for double precision or `E` for single precision. In the Fortran 90 Library, the names of all modules, procedures or derived types begin with the prefix `nag_`, but follow no rigid scheme; they are of different lengths, and are intended to be much more intelligible and easier to remember than the Fortran 77 names. The procedure names are not precision dependent: the same names are used for all available precisions (the names are in fact the names of generic interfaces).

Indexing scheme: In the documentation for the Fortran 77 Library, routine documents occur in alphanumeric order of the routine names. For the Fortran 90 Library, each module document has a two-level index number of the form $(c.m)$, where c is the chapter number and m is the module number within the chapter. Module documents are ordered according to this index number. References to modules and procedures include this index number wherever necessary, so that the relevant document can be located.

Generic interfaces: In the Fortran 90 Library, each procedure is accessed through a generic interface block. For most procedures this covers versions in different precisions (if implemented); for many procedures it also covers versions which differ in the type and rank of their arguments. See Section 4.4 in the **Essential Introduction**. Each generic interface is documented as if it were a single procedure; the provision of generic interfaces helps to reduce the amount of documentation needed, and one such generic interface may correspond to several different Fortran 77 routines.

Optional arguments: Procedures in the Fortran 90 Library make extensive use of optional arguments. Therefore there is no need to provide separate 'easy-to-use' and 'comprehensive' interfaces, as in some chapters of the Fortran 77 Library. See Section 4.1 in the **Essential Introduction**. Again, this means that one Fortran 90 procedure may correspond to two or more Fortran 77 routines.

Assumed-shape arrays: All array arguments of Fortran 90 Library procedures are *assumed-shape* arrays (except for a few which are array pointers). See Section 4.2 in the **Essential Introduction**. As a result, there is hardly ever any need for separate arguments to specify the dimensions of the problem, or the leading dimensions of 2- or 3-dimensional arrays.

Error handling: Most Fortran 77 Library routines have an integer error-handling argument `IFAIL`. Corresponding to this, Fortran 90 Library procedures have an argument `error`, but it is a *structure* and it is *optional*. If the default error-handling behaviour is satisfactory, this argument can safely and conveniently be omitted. Errors detected by library procedures are classified as either *fatal errors*, *failures* or *warnings*, and the way an error is handled depends on its classification. For more details, see Sections 7 and 10 in the **Essential Introduction**.

3 Guidance by Chapter

This section gives specific guidance for each chapter of the Mark 17 or Mark 18 Fortran 77 Library on how to convert calls to Fortran 77 routines into calls to Fortran 90 procedures.

Tables list the Fortran 90 equivalents (if any) of Fortran 77 routines. Note the following points.

1. Each Fortran 90 procedure name is followed by the index number of the module document in which the procedure is described.
2. If the Fortran 90 procedure name is followed by a dagger (†), it means that the procedure provides equivalent functionality, but there are non-trivial differences in the algorithm; however, these differences may not be important for your application.
3. For some Fortran 77 routines the name of an equivalent Fortran 90 intrinsic procedure is given.

4. If the name of a Mark 17 or Mark 18 Fortran 77 routine does not appear in these tables, it means that there is no equivalent procedure in this Release of the Fortran 90 Library.

A00 – Library Identification

A00AAF	nag_lib_ident (1.1)

A02 – Complex Arithmetic

A02AAF	Fortran 90 intrinsic function SQRT
A02ABF	Fortran 90 intrinsic function ABS
A02ACF	Fortran 90 division operator (/)

C02 – Roots of Polynomials

The procedure nag_polynom_roots is a generic procedure that can handle either real or complex polynomials.

C02AFF	nag_polynom_roots (10.1)
C02AGF	nag_polynom_roots (10.1)
C02AHF	nag_polynom_roots (10.1) †
C02AJF	nag_polynom_roots (10.1) †

C05 – Roots of One or More Transcendental Equations

The procedure nag_nlin_sys_sol has options to allow the Jacobian (1st partial derivatives) to be supplied or to be estimated by finite differences; it also has an option to check the Jacobian if it is supplied.

C05ADF	nag_nlin_eqn_sol (10.2)
C05NBF	nag_nlin_sys_sol (10.3)
C05NCF	nag_nlin_sys_sol (10.3)
C05PBF	nag_nlin_sys_sol (10.3)
C05PCF	nag_nlin_sys_sol (10.3)
C05ZAF	incorporated in nag_nlin_sys_sol (10.3)

C06 – Summation of Series

There are two classes of Fortran 90 procedures for discrete Fourier transforms:

basic procedures (with names ending in _basic): they are subroutines which overwrite the transform on the input data, and store Hermitian sequences compactly in real arrays;

general procedures: they are array-valued functions which return the transform as their result, and store Hermitian sequences in complex arrays.

The basic procedures correspond more closely to the Fortran 77 routines and require less memory; the general procedures are, however, easier to use.

The Fortran 90 procedures use complex arrays for storing complex sequences, whereas the Fortran 77 routines use a pair of real arrays.

There is a separate Fortran 90 procedure nag_fft_trig for computing trigonometric coefficients and storing them for subsequent use; in Fortran 77 this facility is provided as an extra option in C06FPF, -FQF, -FRF and -FUF. There is also a Fortran 90 procedure nag_cmplx_to_herm which converts Hermitian sequences from storage in a complex array to compact storage in a real array.

C06EAF	nag_fft_1d_real or nag_fft_1d_basic (7.1)
C06EBF	nag_fft_1d_real or nag_fft_1d_basic (7.1)
C06ECF	nag_fft_1d or nag_fft_1d_basic (7.1)
C06EKF	nag_fft_conv (7.3)
C06FAF	nag_fft_1d_real or nag_fft_1d_basic (7.1)
C06FBF	nag_fft_1d_real or nag_fft_1d_basic (7.1)
C06FCF	nag_fft_1d or nag_fft_1d_basic (7.1)

C06FKF	nag_fft_conv (7.3)
C06FPF	nag_fft_1d_real or nag_fft_1d_basic (7.1)
C06FQF	nag_fft_1d_real or nag_fft_1d_basic (7.1)
C06FRF	nag_fft_1d or nag_fft_1d_basic (7.1)
C06FUF	nag_fft_2d or nag_fft_2d_basic (7.1)
C06FXF	nag_fft_3d or nag_fft_3d_basic (7.1)
C06GBF	nag_conj_herm (7.1)
C06GCF	Fortran 90 intrinsic function CONJG
C06GQF	nag_conj_herm (7.1)
C06GSF	nag_herm_to_cmplx (7.1)
C06HAF	nag_fft_sin (7.2)
C06HBF	nag_fft_cos (7.2)
C06HCF	nag_fft_qtr_sin (7.2)
C06HDF	nag_fft_qtr_cos (7.2)

D01 – Quadrature

The Fortran 90 procedures require an *array-valued* function to be supplied to define the integrand, whereas most of the Fortran 77 routines require a scalar-valued function. Using an array-valued function reduces the overhead in computing large numbers of integrand values.

D01AHF	nag_quad_1d_gen (11.1) †
D01AJF	nag_quad_1d_gen (11.1)
D01AKF	nag_quad_1d_gen (11.1)
D01ALF	nag_quad_1d_gen (11.1)
D01AMF	nag_quad_1d_inf_gen (11.2)
D01ANF	nag_quad_1d_wt_trig (11.1)
D01APF	nag_quad_1d_wt_end_sing (11.1)
D01AQF	nag_quad_1d_wt_hilb (11.1)
D01ASF	nag_quad_1d_inf_wt_trig (11.2)
D01ATF	nag_quad_1d_gen (11.1)
D01AUF	nag_quad_1d_gen (11.1)
D01BCF	nag_quad_gs_wt_absc (11.4)
D01DAF	nag_quad_2d (11.3)
D01EAF	nag_quad_md_rect_mintg (11.3)
D01FCF	nag_quad_md_rect (11.3)
D01GAF	nag_quad_1d_data (11.1)
D01GBF	nag_quad_md_rect (11.3) †

D02 – Ordinary Differential Equations

D02BAF	nag_rk_setup + nag_rk_interval (12.1) †
D02BBF	nag_rk_setup + nag_rk_interval (12.1) †
D02BDF	nag_rk_setup + nag_rk_interval + nag_rk_global_err (12.1) †
D02PAF	nag_rk_setup + nag_rk_interval + nag_rk_info (12.1) †
D02PCF	nag_rk_interval (12.1)
D02PDF	nag_rk_step (12.1)
D02PVF	nag_rk_setup (12.1)
D02PWF	nag_rk_reset_end (12.1)
D02PXF	nag_rk_interp (12.1)
D02PYF	nag_rk_info (12.1)
D02PZF	nag_rk_global_err (12.1)
D02XAF	nag_rk_interp (12.1) †
D02XBF	nag_rk_interp (12.1) †

D03 – Partial Differential Equations

D03FAF	nag_pde_helm_3d (13.1)

D04 – Differentiation

No equivalent Fortran 90 procedures in this Release.

D05 – Integral Equations

No equivalent Fortran 90 procedures in this Release.

E01 – Interpolation

E01BAF	nag_spline_1d_interp (8.2)
E01BEF	nag_pch_monot_interp (8.1)
E01BFF	nag_pch_eval (8.1)
E01BGF	nag_pch_eval (8.1)
E01BHF	nag_pch_intg (8.1)
E01DAF	nag_spline_2d_interp (8.3)
E01SEF	nag_scat_2d_interp (8.4)
E01SFF	nag_scat_2d_eval (8.4)

E02 – Curve and Surface Fitting

E02BAF	nag_spline_1d_lsq_fit (8.2)
E02BBF	nag_spline_1d_eval (8.2)
E02BCF	nag_spline_1d_eval (8.2)
E02BDF	nag_spline_1d_intg (8.2)
E02BEF	nag_spline_1d_auto_fit (8.2)
E02DAF	nag_spline_2d_lsq_fit (8.3)
E02DCF	nag_spline_2d_auto_fit (8.3)
E02DDF	nag_spline_2d_auto_fit (8.3)
E02DEF	nag_spline_2d_eval (8.3)
E02DFF	nag_spline_2d_eval (8.3)
E02ZAF	incorporated in nag_spline_2d_lsq_fit (8.3)

E04 – Minimizing or Maximizing a Function

The procedure nag_nlin_lsq_sol corresponds to several Fortran 77 routines because it has options to allow the Jacobian (1st partial derivatives) to be supplied or to be estimated by finite differences; it also has an option to check the Jacobian if it is supplied.

Instead of the option-setting mechanism of the Fortran 77 library, the Fortran 90 procedures use a single optional argument which is a structure: to set non-default options, you must first initialize the structure (using a procedure whose name ends _cntrl_init) and then assign values to some of its components.

E04FCF	nag_nlin_lsq_sol (9.2)
E04FDF	nag_nlin_lsq_sol (9.2)
E04GBF	nag_nlin_lsq_sol (9.2)
E04GCF	nag_nlin_lsq_sol (9.2)
E04GDF	nag_nlin_lsq_sol (9.2) †
E04GEF	nag_nlin_lsq_sol (9.2) †
E04HEF	nag_nlin_lsq_sol (9.2) †
E04HFF	nag_nlin_lsq_sol (9.2) †
E04JAF	nag_nlp_sol (9.3) †
E04KAF	nag_nlp_sol (9.3) †
E04KCF	nag_nlp_sol (9.3) †
E04KDF	nag_nlp_sol (9.3) †
E04MBF	nag_qp_sol (9.1) †
E04MFF	nag_qp_sol (9.1)
E04MHF	nag_qp_cntrl_init (9.1) + assignments to structure-components

E04NAF	nag_qp_sol (9.1) †
E04NFF	nag_qp_sol (9.1)
E04NHF	nag_qp_cntrl_init (9.1) + assignments to structure-components
E04UNF	nag_con_nlin_lsq_sol (9.4)
E04UCF	nag_nlp_sol (9.3)
E04UEF	nag_nlp_cntrl_init (9.3) + assignments to structure-components
E04YAF	incorporated in nag_nlin_lsq_sol (9.2)
E04YCF	nag_nlin_lsq_cov (9.2)

F – Linear Algebra

Most of the Fortran 90 procedures are generic procedures which can handle problems with either real or complex data.

F01 – Matrix Operations, Including Inversion

F01AGF	nag_sym_tridiag_reduc (6.1) †
F01AHF	nag_sym_tridiag_orth (6.1) †
F01AJF	nag_sym_tridiag_reduc (6.1) †
F01AYF	nag_sym_tridiag_reduc (6.1) †
F01AZF	nag_sym_tridiag_orth (6.1) †
F01BCF	nag_sym_tridiag_reduc (6.1) †
F01BNF	nag_sym_lin_fac (5.2)
F01BTF	nag_gen_lin_fac (5.1) †
F01BXF	nag_sym_lin_fac (5.2) †
F01CKF	Fortran 90 intrinsic function MATMUL
F01CTF	Fortran 90 array operations
F01CWF	Fortran 90 array operations
F01LBF	nag_gen_bnd_lin_fac (5.4) †
F01NAF	nag_gen_bnd_lin_fac (5.4) †
F01QCF	nag_qr_fac (6.4) †
F01QDF	nag_qr_orth; also incorporated in nag_qr_fac (6.4) †
F01QEF	nag_qr_orth (6.4) †
F01QFF	nag_qr_fac (6.4)
F01RCF	nag_qr_fac (6.4) †
F01RDF	nag_qr_orth; also incorporated in nag_qr_fac (6.4) †
F01REF	nag_qr_orth (6.4) †
F01RFF	nag_qr_fac (6.4)

F02 – Eigenvalues and Eigenvectors

F02AAF	nag_sym_eig_all (6.1) †
F02ABF	nag_sym_eig_all (6.1) †
F02ADF	nag_sym_gen_eig_all (6.5) †
F02AEF	nag_sym_gen_eig_all (6.5) †
F02AFF	nag_nsym_eig_all (6.2) †
F02AGF	nag_nsym_eig_all (6.2) †
F02AJF	nag_nsym_eig_all (6.2) †
F02AKF	nag_nsym_eig_all (6.2) †
F02AMF	nag_sym_tridiag_eig_all (6.1) †
F02AVF	nag_sym_tridiag_eig_all (6.1) †
F02AWF	nag_sym_eig_all (6.1) †
F02AXF	nag_sym_eig_all (6.1) †
F02AYF	nag_sym_tridiag_eig_all (6.1) †
F02BBF	nag_sym_eig_sel (6.1) †
F02BEF	nag_sym_tridiag_eig_val + nag_sym_tridiag_eig_vec (6.1) †
F02BFF	nag_sym_tridiag_eig_val (6.1) †
F02BJF	nag_nsym_gen_eig_all (6.6) †

F02EAF	nag_schur_fac (6.2)
F02EBF	nag_nsym_eig_all (6.2)
F02FAF	nag_sym_eig_all (6.1)
F02FCF	nag_sym_eig_sel (6.1)
F02FDF	nag_sym_gen_eig_all (6.5)
F02GAF	nag_schur_fac (6.2)
F02GBF	nag_nsym_eig_all (6.2)
F02GJF	nag_nsym_gen_eig_all (6.6) †
F02HAF	nag_sym_eig_all (6.1)
F02HCF	nag_sym_eig_sel (6.1)
F02HDF	nag_sym_gen_eig_all (6.5)
F02SYF	nag_bidiag_svd (6.3) †
F02UYF	nag_bidiag_svd (6.3) †
F02WEF	nag_gen_svd (6.3) †
F02XEF	nag_gen_svd (6.3) †

F03 – Determinants

F03AAF	incorporated in nag_gen_lin_fac (5.1)
F03ABF	incorporated in nag_sym_lin_fac (5.2)
F03ACF	incorporated in nag_sym_bnd_lin_fac (5.5)
F03ADF	incorporated in nag_gen_lin_fac (5.1)
F03AEF	nag_sym_lin_fac (5.2)
F03AFF	nag_gen_lin_fac (5.1) †
F03AGF	nag_sym_bnd_lin_fac (5.5)
F03AHF	nag_gen_lin_fac (5.1) †

F04 – Simultaneous Linear Equations

Some of the Fortran 77 procedures use *iterative refinement* in *additional precision*. There is no equivalent facility in the Fortran 90 procedures, though there is an option for iterative refinement in working precision. See the **Chapter Introduction** for Chapter 5 for more details.

F04AAF	nag_gen_lin_sol (5.1)
F04ABF	nag_sym_lin_sol (5.2) †
F04ACF	nag_sym_bnd_lin_sol (5.5) †
F04ADF	nag_gen_lin_sol (5.1)
F04AEF	nag_gen_lin_sol (5.1) †
F04AFF	nag_sym_lin_sol_fac (5.2) †
F04AGF	nag_sym_lin_sol_fac (5.2)
F04AHF	nag_gen_lin_sol_fac (5.1) †
F04AJF	nag_gen_lin_sol_fac (5.1)
F04AKF	nag_gen_lin_sol_fac (5.1)
F04ALF	nag_sym_bnd_lin_sol_fac (5.5)
F04AMF	nag_lin_lsq_sol (6.4) †
F04ANF	nag_lin_lsq_sol_qr (6.4) †
F04ARF	nag_gen_lin_sol (5.1)
F04ASF	nag_sym_lin_sol (5.2) †
F04ATF	nag_gen_lin_sol (5.1) †
F04AWF	nag_sym_lin_sol_fac (5.2)
F04AYF	nag_gen_lin_sol_fac (5.1) †
F04AZF	nag_sym_lin_sol_fac (5.2) †
F04JAF	nag_lin_lsq_sol (6.4) †
F04JDF	nag_lin_lsq_sol (6.4) †
F04JGF	nag_lin_lsq_sol (6.4) †
F04LDF	nag_gen_bnd_lin_sol_fac (5.4) †
F04NAF	nag_gen_bnd_lin_sol_fac (5.4) †

F05 – Orthogonalisation

F05AAF	nag_qr_fac (6.4) †

F06 – Linear Algebra Support Routines

F06RAF	nag_gen_mat_norm (4.1)
F06RBF	nag_gen_bnd_mat_norm (4.1)
F06RCF	nag_sym_mat_norm (4.1)
F06RDF	nag_sym_mat_norm (4.1)
F06REF	nag_sym_bnd_mat_norm (4.1)
F06RJF	nag_trap_mat_norm (4.1)
F06RKF	nag_tri_mat_norm (4.1)
F06RLF	nag_tri_bnd_mat_norm (4.1)
F06RMF	nag_hessen_mat_norm (4.1)
F06UAF	nag_gen_mat_norm (4.1)
F06UBF	nag_gen_bnd_mat_norm (4.1)
F06UCF	nag_sym_mat_norm (4.1)
F06UDF	nag_sym_mat_norm (4.1)
F06UEF	nag_sym_bnd_mat_norm (4.1)
F06UFF	nag_sym_mat_norm (4.1)
F06UGF	nag_sym_mat_norm (4.1)
F06UHF	nag_sym_bnd_mat_norm (4.1)
F06UJF	nag_trap_mat_norm (4.1)
F06UKF	nag_tri_mat_norm (4.1)
F06ULF	nag_tri_bnd_mat_norm (4.1)
F06UMF	nag_hessen_mat_norm (4.1)

F07 – Linear Equations (LAPACK)

F07ADF	nag_gen_lin_fac (5.1)
F07AEF	nag_gen_lin_sol_fac (5.1)
F07AGF	incorporated in nag_gen_lin_fac (5.1)
F07AHF	incorporated in nag_gen_lin_sol_fac (5.1)
F07ARF	nag_gen_lin_fac (5.1)
F07ASF	nag_gen_lin_sol_fac (5.1)
F07AUF	incorporated in nag_gen_lin_fac (5.1)
F07AVF	incorporated in nag_gen_lin_sol_fac (5.1)
F07BDF	nag_gen_bnd_lin_fac (5.4)
F07BEF	nag_gen_bnd_lin_sol_fac (5.4)
F07BGF	incorporated in nag_gen_bnd_lin_fac (5.4)
F07BHF	incorporated in nag_gen_bnd_lin_sol_fac (5.4)
F07BRF	nag_gen_bnd_lin_fac (5.4)
F07BSF	nag_gen_bnd_lin_sol_fac (5.4)
F07BUF	incorporated in nag_gen_bnd_lin_fac (5.4)
F07BVF	incorporated in nag_gen_bnd_lin_sol_fac (5.4)
F07FDF	nag_sym_lin_fac (5.2)
F07FEF	nag_sym_lin_sol_fac (5.2)
F07FGF	incorporated in nag_sym_lin_fac (5.2)
F07FHF	incorporated in nag_sym_lin_sol_fac (5.2)
F07FRF	nag_sym_lin_fac (5.2)
F07FSF	nag_sym_lin_sol_fac (5.2)
F07FUF	incorporated in nag_sym_lin_fac (5.2)
F07FVF	incorporated in nag_sym_lin_sol_fac (5.2)
F07GDF	nag_sym_lin_fac (5.2)
F07GEF	nag_sym_lin_sol_fac (5.2)
F07GGF	incorporated in nag_sym_lin_fac (5.2)
F07GHF	incorporated in nag_sym_lin_sol_fac (5.2)

F07GRF	nag_sym_lin_fac (5.2)
F07GSF	nag_sym_lin_sol_fac (5.2)
F07GUF	incorporated in nag_sym_lin_fac (5.2)
F07GVF	incorporated in nag_sym_lin_sol_fac (5.2)
F07HDF	nag_sym_bnd_lin_fac (5.5)
F07HEF	nag_sym_bnd_lin_sol_fac (5.5)
F07HGF	incorporated in nag_sym_bnd_lin_fac (5.5)
F07HHF	incorporated in nag_sym_bnd_lin_sol_fac (5.5)
F07HRF	nag_sym_bnd_lin_fac (5.5)
F07HSF	nag_sym_bnd_lin_sol_fac (5.5)
F07HUF	incorporated in nag_sym_bnd_lin_fac (5.5)
F07HVF	incorporated in nag_sym_bnd_lin_sol_fac (5.5)
F07MDF	nag_sym_lin_fac (5.2)
F07MEF	nag_sym_lin_sol_fac (5.2)
F07MGF	incorporated in nag_sym_lin_fac (5.2)
F07MHF	incorporated in nag_sym_lin_sol_fac (5.2)
F07MRF	nag_sym_lin_fac (5.2)
F07MSF	nag_sym_lin_sol_fac (5.2)
F07MUF	incorporated in nag_sym_lin_fac (5.2)
F07MVF	incorporated in nag_sym_lin_sol_fac (5.2)
F07NRF	nag_sym_lin_fac (5.2)
F07NSF	nag_sym_lin_sol_fac (5.2)
F07NUF	incorporated in nag_sym_lin_fac (5.2)
F07NVF	incorporated in nag_sym_lin_sol_fac (5.2)
F07PDF	nag_sym_lin_fac (5.2)
F07PEF	nag_sym_lin_sol_fac (5.2)
F07PGF	incorporated in nag_sym_lin_fac (5.2)
F07PHF	incorporated in nag_sym_lin_sol_fac (5.2)
F07PRF	nag_sym_lin_fac (5.2)
F07PSF	nag_sym_lin_sol_fac (5.2)
F07PUF	incorporated in nag_sym_lin_fac (5.2)
F07PVF	incorporated in nag_sym_lin_sol_fac (5.2)
F07QRF	nag_sym_lin_fac (5.2)
F07QSF	nag_sym_lin_sol_fac (5.2)
F07QUF	incorporated in nag_sym_lin_fac (5.2)
F07QVF	incorporated in nag_sym_lin_sol_fac (5.2)
F07TEF	nag_tri_lin_sol (5.3)
F07TGF	nag_tri_lin_cond (5.3)
F07THF	incorporated in nag_tri_lin_sol (5.3)
F07TSF	nag_tri_lin_sol (5.3)
F07TUF	nag_tri_lin_cond (5.3)
F07TVF	incorporated in nag_tri_lin_sol (5.3)
F07UEF	nag_tri_lin_sol (5.3)
F07UGF	nag_tri_lin_cond (5.3)
F07UHF	incorporated in nag_tri_lin_sol (5.3)
F07USF	nag_tri_lin_sol (5.3)
F07UUF	nag_tri_lin_cond (5.3)
F07UVF	incorporated in nag_tri_lin_sol (5.3)

F08 – Least-squares and Eigenvalue Problems (LAPACK)

F08AEF	nag_qr_fac (6.4)
F08AFF	nag_qr_orth; also incorporated in nag_qr_fac (6.4)
F08AGF	nag_qr_orth (6.4)
F08ASF	nag_qr_fac (6.4)
F08ATF	nag_qr_orth; also incorporated in nag_qr_fac (6.4)
F08AUF	nag_qr_orth (6.4)

F08BEF	nag_qr_fac (6.4)
F08BSF	nag_qr_fac (6.4)
F08FEF	nag_sym_tridiag_reduc (6.1)
F08FFF	nag_sym_tridiag_orth; also incorporated in nag_sym_tridiag_reduc (6.1)
F08FGF	nag_sym_tridiag_orth (6.1)
F08FSF	nag_sym_tridiag_reduc (6.1)
F08FTF	nag_sym_tridiag_orth; also incorporated in nag_sym_tridiag_reduc (6.1)
F08FUF	nag_sym_tridiag_orth (6.1)
F08GEF	nag_sym_tridiag_reduc (6.1)
F08GFF	nag_sym_tridiag_orth; also incorporated in nag_sym_tridiag_reduc (6.1)
F08GGF	nag_sym_tridiag_orth (6.1)
F08GSF	nag_sym_tridiag_reduc (6.1)
F08GTF	nag_sym_tridiag_orth; also incorporated in nag_sym_tridiag_reduc (6.1)
F08GUF	nag_sym_tridiag_orth (6.1)
F08JEF	nag_sym_tridiag_eig_all (6.1)
F08JFF	nag_sym_tridiag_eig_all (6.1)
F08JGF	nag_sym_tridiag_eig_all (6.1)
F08JJF	nag_sym_tridiag_eig_val (6.1)
F08JKF	nag_sym_tridiag_eig_vec (6.1)
F08JSF	nag_sym_tridiag_eig_all (6.1)
F08JUF	nag_sym_tridiag_eig_all (6.1)
F08JXF	nag_sym_tridiag_eig_vec (6.1)
F08KEF	nag_gen_bidiag_reduc (6.3)
F08KFF	incorporated in nag_gen_bidiag_reduc (6.3)
F08KSF	nag_gen_bidiag_reduc (6.3)
F08KTF	incorporated in nag_gen_bidiag_reduc (6.3)
F08MEF	nag_bidiag_svd (6.3)
F08MSF	nag_bidiag_svd (6.3)

G01 – Simple Calculations on Statistical Data

G01AAF	nag_summary_stats_1v (22.1)
G01ADF	nag_summary_stats_1v (22.1)
G01BJF	nag_binom_prob (20.7)
G01BKF	nag_poisson_prob (20.7)
G01BLF	nag_hypergeo_prob (20.7)
G01CEF	nag_normal_deviate (20.1) †
G01EAF	nag_normal_prob (20.1)
G01EBF	nag_t_prob (20.2)
G01ECF	nag_chisq_prob (20.3)
G01EDF	nag_f_prob (20.4)
G01EEF	nag_beta_prob (20.5)
G01EFF	nag_gamma_prob (20.6)
G01FAF	nag_normal_deviate (20.1)
G01FBF	nag_t_deviate (20.2)
G01FCF	nag_chisq_deviate (20.3)
G01FDF	nag_f_deviate (20.4)
G01FEF	nag_beta_deviate (20.5)
G01FFF	nag_gamma_deviate (20.6)
G01HAF	nag_bivar_normal_prob (20.1)
G01HBF	nag_mv_normal_prob (20.1)

G02 – Correlation and Regression Analysis

G02BXF	nag_prod_mom_correl (25.2)
G02BYF	nag_part_correl (25.2)
G02CAF	nag_simple_lin_reg (25.1)
G02CBF	nag_simple_lin_reg (25.1)

G02CCF	nag_simple_lin_reg (25.1)
G02CDF	nag_simple_lin_reg (25.1)
G02DAF	nag_mult_lin_reg (25.1)

G03 – Multivariate Methods

G03AAF	nag_prin_comp (28.1)
G03ACF	nag_canon_var (28.2)
G03BAF	nag_orthomax (28.3)

G04 – Analysis of Variance

No equivalent Fortran 90 procedures in this Release.

G05 – Random Number Generators

The Fortran 90 procedures use an argument called `seed` (a structure) to hold information about the stream of random numbers being generated. In the Fortran 77 Library, this information is 'hidden' in a `COMMON` block, and routines `G05CFF` and `G05CGF` are provided to save and restore this information if two or more independent streams are being generated; no such procedures are needed in the Fortran 90 Library. See the **Chapter Introduction** for Chapter 21 for further guidance.

G05CAF	nag_rand_uniform (21.2)
G05CBF	nag_rand_seed_set (21.1)
G05CCF	nag_rand_seed_set (21.1)
G05DAF	nag_rand_uniform (21.2)
G05DBF	nag_rand_neg_exp (21.2)
G05DDF	nag_rand_normal (21.2)
G05EAF	nag_rand_mv_normal (21.2)
G05EDF	nag_rand_binom (21.3)
G05EEF	nag_rand_neg_binom (21.3)
G05EFF	nag_rand_hypergeo (21.3)
G05EXF	nag_rand_user_dist (21.3)
G05EYF	nag_rand_ref_vec (21.3)
G05EZF	nag_rand_mv_normal (21.2)
G05FAF	nag_rand_uniform (21.2)
G05FBF	nag_rand_neg_exp (21.2)
G05FDF	nag_rand_normal (21.2)
G05FEF	nag_rand_beta (21.2)
G05FFF	nag_rand_gamma (21.2)

G07 – Univariate Estimation

No equivalent Fortran 90 procedures in this Release.

G08 – Nonparametric Statistics

No equivalent Fortran 90 procedures in this Release.

G10 – Smoothing in Statistics

No equivalent Fortran 90 procedures in this Release.

G11 – Contingency Table Analysis

No equivalent Fortran 90 procedures in this Release.

G12 – Survival Analysis

No equivalent Fortran 90 procedures in this Release.

G13 – Time Series Analysis

G13ABF	nag_tsa_acf (29.1)
G13ACF	nag_tsa_pacf (29.1)
G13EAF	nag_kalman_sqrt_cov_var (29.2)
G13EBF	nag_kalman_sqrt_cov_invar (29.2)

H – Operations Research

No equivalent Fortran 90 procedures in this Release.

M01 – Sorting

Most of the Fortran 90 procedures are generic procedures which can handle integer, real or character data.

M01CAF	nag_sort_vec (1.4)
M01CBF	nag_sort_vec (1.4)
M01CCF	nag_sort_vec (1.4)
M01DAF	nag_rank_vec (1.4)
M01DBF	nag_rank_vec (1.4)
M01DCF	nag_rank_vec (1.4)
M01DEF	nag_rank_mat (1.4)
M01DFF	nag_rank_mat (1.4)
M01DJF	nag_rank_mat (1.4)
M01DKF	nag_rank_mat (1.4)
M01DZF	nag_rank_arb_data (1.4)
M01EAF	nag_reorder_vec (1.4)
M01EBF	nag_reorder_vec (1.4)
M01ECF	nag_reorder_vec (1.4)
M01ZAF	nag_invert_perm (1.4)
M01ZBF	nag_check_perm (1.4)
M01ZCF	nag_decomp_perm (1.4)

P01 – Error Trapping

There is no documented Fortran 90 equivalent for the Fortran 77 routine P01ABF, but this is not normally called by users. For a description of error-handling in the Fortran 90 Library, see Sections 7 and 10 in the **Essential Introduction** or the module document nag_error_handling (1.2).

S – Approximations of Special Functions

S07AAF	Fortran 90 intrinsic function TAN
S09AAF	Fortran 90 intrinsic function ASIN
S09ABF	Fortran 90 intrinsic function ACOS
S10AAF	Fortran 90 intrinsic function TANH
S10ABF	Fortran 90 intrinsic function SINH
S10ACF	Fortran 90 intrinsic function COSH
S11AAF	nag_arctanh (3.1)
S11ABF	nag_arcsinh (3.1)
S11ACF	nag_arccosh (3.1)
S14AAF	nag_gamma (3.2)
S14ABF	nag_log_gamma (3.2)
S14ACF	nag_polygamma (3.2)
S14ADF	nag_polygamma (3.2)
S14BAF	nag_incompl_gamma (3.2)
S15ABF	nag_normal_prob (20.1)
S15ACF	nag_normal_prob (20.1)
S15ADF	nag_erfc (3.3)

S15AEF	nag_erf (3.3)
S15AFF	nag_dawson (3.3)
S17ACF	nag_bessel_y0 (3.4)
S17ADF	nag_bessel_y1 (3.4)
S17AEF	nag_bessel_j0 (3.4)
S17AFF	nag_bessel_j1 (3.4)
S17AGF	nag_airy_ai (3.8)
S17AHF	nag_airy_bi (3.8)
S17AJF	nag_airy_ai (3.8)
S17AKF	nag_airy_bi (3.8)
S17DCF	nag_bessel_y (3.4)
S17DEF	nag_bessel_j (3.4)
S17DGF	nag_airy_ai (3.8)
S17DHF	nag_airy_bi (3.8)
S18ACF	nag_bessel_k0 (3.4)
S18ADF	nag_bessel_k1 (3.4)
S18AEF	nag_bessel_i0 (3.4)
S18AFF	nag_bessel_i1 (3.4)
S18CCF	nag_bessel_k0 (3.4)
S18CDF	nag_bessel_k1 (3.4)
S18CEF	nag_bessel_i0 (3.4)
S18CFF	nag_bessel_i1 (3.4)
S18DCF	nag_bessel_k (3.4)
S18DEF	nag_bessel_i (3.4)
S19AAF	nag_kelvin_ber (3.9)
S19ABF	nag_kelvin_bei (3.9)
S19ACF	nag_kelvin_ker (3.9)
S19ADF	nag_kelvin_kei (3.9)
S20ACF	nag_fresnel_s (3.5)
S20ADF	nag_fresnel_c (3.5)
S21BAF	nag_ell_rc (3.6)
S21BBF	nag_ell_rf (3.6)
S21BCF	nag_ell_rd (3.6)
S21BDF	nag_ell_rj (3.6)
S21CAF	nag_ell_jac (3.7)

X01 – Mathematical Constants

X01AAF	nag_pi (1.5)
X01ABF	nag_euler_constant (1.5)

X02 – Machine Constants

Most of the X02 functions have counterparts as Fortran 90 intrinsic functions. However, the intrinsic functions may not give exactly the same values. For example, on some machines the value of EPSILON may differ by a factor of 2 from the value of X02AJF. For many purposes the differences are not likely to be significant.

X02AJF	Fortran 90 intrinsic function EPSILON
X02AKF	Fortran 90 intrinsic function TINY
X02ALF	Fortran 90 intrinsic function HUGE
X02BBF	Fortran 90 intrinsic function HUGE
X02BHF	Fortran 90 intrinsic function RADIX
X02BJF	Fortran 90 intrinsic function DIGITS
X02BKF	Fortran 90 intrinsic function MINEXPONENT
X02BLF	Fortran 90 intrinsic function MAXEXPONENT

X03 – Innerproducts

No equivalent Fortran 90 procedures in this Release if the use of *additional precision* is important. If working precision is used, the Fortran 90 intrinsic function `DOTPRODUCT` may be a suitable replacement.

X04 – Input/Output Utilities

The Fortran 90 procedures are generic procedures which can handle real or complex data; `nag_write_gen_mat` can also handle integer data.

X04CAF	`nag_write_gen_mat` (1.3)
X04CBF	`nag_write_gen_mat` (1.3)
X04CCF	`nag_write_gen_mat` or `nag_write_tri_mat` (1.3)
X04CDF	`nag_write_gen_mat` or `nag_write_tri_mat` (1.3)
X04CEF	`nag_write_bnd_mat` (1.3)
X04CFF	`nag_write_bnd_mat` (1.3)
X04DAF	`nag_write_gen_mat` (1.3)
X04DBF	`nag_write_gen_mat` (1.3)
X04DCF	`nag_write_gen_mat` or `nag_write_tri_mat` (1.3)
X04DDF	`nag_write_gen_mat` or `nag_write_tri_mat` (1.3)
X04DEF	`nag_write_bnd_mat` (1.3)
X04DFF	`nag_write_bnd_mat` (1.3)
X04EAF	`nag_write_gen_mat` (1.3)
X04EBF	`nag_write_gen_mat` (1.3)

X05 – Date and Time Utilities

X05AAF	Fortran 90 intrinsic subroutine `DATE_AND_TIME`