

GAMS Index for the NAG Fortran 90 Library

This index classifies NAG *f90* procedures according to Version 2 of the GAMS classification scheme described in [1]. Note that only those GAMS classes which contain Library procedures, either directly or in a subclass, are included below.

C	Elementary and special functions (<i>search also class L5</i>)	
C3a2	Chebyshev, Legendre	
	nag_cheb_1d_eval (8.5)	Evaluation of fitted polynomial in one variable, from Chebyshev series form
C4	Elementary transcendental functions	
C4c	Hyperbolic, inverse hyperbolic	
	nag_arctanh (3.1)	Inverse hyperbolic tangent, $\operatorname{arctanh} x$
	nag_arcsinh (3.1)	Inverse hyperbolic sine, $\operatorname{arcsinh} x$
	nag_arccosh (3.1)	Inverse hyperbolic cosine, $\operatorname{arccosh} x$
C7	Gamma	
C7a	Gamma, log gamma, reciprocal gamma	
	nag_gamma (3.2)	Gamma function
	nag_log_gamma (3.2)	Log gamma function
C7c	Psi function	
	nag_polygamma (3.2)	Polygamma functions
C7e	Incomplete gamma	
	nag_incompl_gamma (3.2)	Incomplete gamma functions
C8	Error functions	
C8a	Error functions, their inverses, integrals, including the Normal distribution function	
	nag_normal_prob (20.1)	Computes probabilities for various parts of a univariate Normal distribution
	nag_erfc (3.3)	Complementary error function $\operatorname{erfc} x$
	nag_erf (3.3)	Error function $\operatorname{erf} x$
C8b	Fresnel integrals	
	nag_fresnel_s (3.5)	Fresnel integral $S(x)$
	nag_fresnel_c (3.5)	Fresnel integral $C(x)$
C8c	Dawson's integral	
	nag_dawson (3.3)	Dawson's integral $F(x)$
C10	Bessel functions	
C10a	J, Y, H_1, H_2	
C10a1	Real argument, integer order	
	nag_bessel_y0 (3.4)	Bessel function $Y_0(x)$
	nag_bessel_y1 (3.4)	Bessel function $Y_1(x)$

		<code>nag_bessel_j0</code> (3.4)	Bessel function $J_0(x)$
		<code>nag_bessel_j1</code> (3.4)	Bessel function $J_1(x)$
C10a4	Complex argument, real order	<code>nag_bessel_j</code> (3.4)	Bessel function $J_\nu(z)$
		<code>nag_bessel_y</code> (3.4)	Bessel function $Y_\nu(z)$
C10b	I, K		
C10b1	Real argument, integer order	<code>nag_bessel_k0</code> (3.4)	Modified Bessel function $K_0(x)$
		<code>nag_bessel_k1</code> (3.4)	Modified Bessel function $K_1(x)$
		<code>nag_bessel_i0</code> (3.4)	Modified Bessel function $I_0(x)$
		<code>nag_bessel_i1</code> (3.4)	Modified Bessel function $I_1(x)$
C10b4	Complex argument, real order	<code>nag_bessel_i</code> (3.4)	Modified Bessel function $I_\nu(z)$
		<code>nag_bessel_k</code> (3.4)	Modified Bessel function $K_\nu(z)$
C10c	Kelvin functions		
		<code>nag_kelvin_ber</code> (3.9)	Kelvin function $\text{ber } x$
		<code>nag_kelvin_bei</code> (3.9)	Kelvin function $\text{bei } x$
		<code>nag_kelvin_ker</code> (3.9)	Kelvin function $\text{ker } x$
		<code>nag_kelvin_kei</code> (3.9)	Kelvin function $\text{kei } x$
C10d	Airy and Scorer functions		
		<code>nag_airy_ai</code> (3.8)	Airy function $\text{Ai}(z)$
		<code>nag_airy_bi</code> (3.8)	Airy function $\text{Bi}(z)$
C13	Jacobian elliptic functions, theta functions		
		<code>nag_ell_jac</code> (3.7)	Jacobian elliptic functions sn , cn and dn
C14	Elliptic integrals		
		<code>nag_ell_rc</code> (3.6)	Degenerate form of elliptic integral of the first kind
		<code>nag_ell_rf</code> (3.6)	Symmetrised elliptic integral of the first kind
		<code>nag_ell_rd</code> (3.6)	Symmetrised elliptic integral of the second kind
		<code>nag_ell_rj</code> (3.6)	Symmetrised elliptic integral of the third kind
D	Linear Algebra		
D1	Elementary vector and matrix operations		
D1a	Elementary vector operations		
D1a10	Convolutions		
		<code>nag_fft_conv</code> (7.3)	Computes the convolution or correlation of two real or complex vectors
D1b	Elementary matrix operations		
D1b2	Norm		

	<code>nag_gen_mat_norm</code> (4.1)	Computes a norm, or the element of largest absolute value, of a general real or complex matrix
	<code>nag_gen_bnd_mat_norm</code> (4.1)	Computes a norm, or the element of largest absolute value, of a real or complex square banded matrix
	<code>nag_sym_mat_norm</code> (4.1)	Computes a norm, or the element of largest absolute value, of a real or complex, symmetric or Hermitian matrix, stored in conventional or packed storage
	<code>nag_sym_bnd_mat_norm</code> (4.1)	Computes a norm, or the element of largest absolute value, of a real or complex, symmetric or Hermitian band matrix
	<code>nag_trap_mat_norm</code> (4.1)	Computes a norm, or the element of largest absolute value, of a real or complex trapezoidal matrix
	<code>nag_tri_mat_norm</code> (4.1)	Computes a norm, or the element of largest absolute value, of a real or complex triangular matrix, stored in conventional or packed storage
	<code>nag_tri_bnd_mat_norm</code> (4.1)	Computes a norm, or the element of largest absolute value, of a real or complex triangular band matrix
	<code>nag_hessen_mat_norm</code> (4.1)	Computes a norm, or the element of largest absolute value, of a real or complex upper Hessenberg matrix
D2	Solution of systems of linear equations (including inversion, <i>LU</i> and related decompositions)	
D2a	Real nonsymmetric matrices	
D2a1	General	
	<code>nag_gen_mat_inv</code> (4.2)	Computes the inverse of a general real or complex matrix
	<code>nag_gen_mat_inv_fac</code> (4.2)	Computes the inverse of a general real or complex matrix, with the matrix previously factorized using <code>nag_gen_lin_fac</code>
	<code>nag_gen_lin_fac</code> (5.1)	Performs an <i>LU</i> factorization of a general real or complex matrix
	<code>nag_gen_lin_sol</code> (5.1)	Solves a general real or complex system of linear equations with one or many right-hand sides
	<code>nag_gen_lin_sol_fac</code> (5.1)	Solves a general real or complex system of linear equations, with coefficient matrix previously factorized by <code>nag_gen_lin_fac</code>
D2a2	Banded	
	<code>nag_gen_bnd_lin_fac</code> (5.4)	Performs an <i>LU</i> factorization of a general real or complex band matrix
	<code>nag_gen_bnd_lin_sol</code> (5.4)	Solves a general real or complex banded system of linear equations, with one or many right-hand sides
	<code>nag_gen_bnd_lin_sol_fac</code> (5.4)	Solves a general real or complex banded system of linear equations, with coefficient matrix previously factorized by <code>nag_gen_bnd_lin_fac</code>
D2a3	Triangular	
	<code>nag_tri_mat_inv</code> (4.2)	Computes the inverse of a real or complex triangular matrix
	<code>nag_tri_lin_sol</code> (5.3)	Solves a real or complex triangular system of linear equations
	<code>nag_tri_lin_cond</code> (5.3)	Estimates the condition number of a real or complex triangular matrix
D2a4	Sparse	
	<code>nag_sparse_prec_init_ilu</code> (5.6)	Initializes sparse ILU preconditioner for real non-symmetric or complex non-Hermitian matrices
	<code>nag_sparse_prec_sol</code> (5.6)	Sparse matrix preconditioned system solver
	<code>nag_sparse_gen_lin_sol</code> (5.7)	General sparse linear system solver
D2b	Real symmetric matrices	
D2b1	General	
D2b1a	Indefinite	

	<code>nag_sym_mat_inv</code> (4.2)	Computes the inverse of a real or complex, symmetric or Hermitian matrix
	<code>nag_sym_mat_inv_fac</code> (4.2)	Computes the inverse of a real or complex, symmetric or Hermitian matrix, with the matrix previously factorized using <code>nag_sym_lin_fac</code>
	<code>nag_sym_lin_sol</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations with one or many right-hand sides
	<code>nag_sym_lin_fac</code> (5.2)	Performs a Cholesky or Bunch–Kaufman factorization of a real or complex, symmetric or Hermitian matrix
	<code>nag_sym_lin_sol_fac</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations, with coefficient matrix previously factorized by <code>nag_sym_lin_fac</code>
D2b1b	Positive definite	
	<code>nag_sym_mat_inv</code> (4.2)	Computes the inverse of a real or complex, symmetric or Hermitian matrix
	<code>nag_sym_mat_inv_fac</code> (4.2)	Computes the inverse of a real or complex, symmetric or Hermitian matrix, with the matrix previously factorized using <code>nag_sym_lin_fac</code>
	<code>nag_sym_lin_sol</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations with one or many right-hand sides
	<code>nag_sym_lin_fac</code> (5.2)	Performs a Cholesky or Bunch–Kaufman factorization of a real or complex, symmetric or Hermitian matrix
	<code>nag_sym_lin_sol_fac</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations, with coefficient matrix previously factorized by <code>nag_sym_lin_fac</code>
D2b2	Positive definite banded	
	<code>nag_sym_bnd_lin_sol</code> (5.5)	Solves a real symmetric or complex Hermitian positive definite banded system of linear equations, with one or many right-hand sides
	<code>nag_sym_bnd_lin_fac</code> (5.5)	Performs a Cholesky factorization of a real symmetric or complex Hermitian positive definite band matrix
	<code>nag_sym_bnd_lin_sol_fac</code> (5.5)	Solves a real symmetric or complex Hermitian positive definite banded system of linear equations, with coefficient matrix previously factorized by <code>nag_sym_bnd_lin_fac</code>
D2c	Complex non-Hermitian matrices	
D2c1	General	
	<code>nag_gen_mat_inv</code> (4.2)	Computes the inverse of a general real or complex matrix
	<code>nag_gen_mat_inv_fac</code> (4.2)	Computes the inverse of a general real or complex matrix, with the matrix previously factorized using <code>nag_gen_lin_fac</code>
	<code>nag_gen_lin_sol</code> (5.1)	Solves a general real or complex system of linear equations with one or many right-hand sides
	<code>nag_gen_lin_fac</code> (5.1)	Performs an <i>LU</i> factorization of a general real or complex matrix
	<code>nag_gen_lin_sol_fac</code> (5.1)	Solves a general real or complex system of linear equations, with coefficient matrix previously factorized by <code>nag_gen_lin_fac</code>
	<code>nag_sym_lin_sol</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations with one or many right-hand sides
	<code>nag_sym_lin_fac</code> (5.2)	Performs a Cholesky or Bunch–Kaufman factorization of a real or complex, symmetric or Hermitian matrix
	<code>nag_sym_lin_sol_fac</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations, with coefficient matrix previously factorized by <code>nag_sym_lin_fac</code>
D2c2	Banded	

	<code>nag_gen_bnd_lin_fac</code> (5.4)	Performs an <i>LU</i> factorization of a general real or complex band matrix
	<code>nag_gen_bnd_lin_sol</code> (5.4)	Solves a general real or complex banded system of linear equations, with one or many right-hand sides
	<code>nag_gen_bnd_lin_sol_fac</code> (5.4)	Solves a general real or complex banded system of linear equations, with coefficient matrix previously factorized by <code>nag_gen_bnd_lin_fac</code>
D2c3	Triangular	
	<code>nag_tri_mat_inv</code> (4.2)	Computes the inverse of a real or complex triangular matrix
	<code>nag_tri_lin_sol</code> (5.3)	Solves a real or complex triangular system of linear equations
	<code>nag_tri_lin_cond</code> (5.3)	Estimates the condition number of a real or complex triangular matrix
D2c4	Sparse	
	<code>nag_sparse_prec_init_ilu</code> (5.6)	Initializes sparse ILU preconditioner for real non-symmetric or complex non-Hermitian matrices
	<code>nag_sparse_prec_sol</code> (5.6)	Sparse matrix preconditioned system solver
	<code>nag_sparse_gen_lin_sol</code> (5.7)	General sparse linear system solver
D2d	Complex Hermitian matrices	
D2d1	General	
D2d1a	Indefinite	
	<code>nag_sym_mat_inv</code> (4.2)	Computes the inverse of a real or complex, symmetric or Hermitian matrix
	<code>nag_sym_mat_inv_fac</code> (4.2)	Computes the inverse of a real or complex, symmetric or Hermitian matrix, with the matrix previously factorized using <code>nag_sym_lin_fac</code>
	<code>nag_sym_lin_sol</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations with one or many right-hand sides
	<code>nag_sym_lin_fac</code> (5.2)	Performs a Cholesky or Bunch–Kaufman factorization of a real or complex, symmetric or Hermitian matrix
	<code>nag_sym_lin_sol_fac</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations, with coefficient matrix previously factorized by <code>nag_sym_lin_fac</code>
D2d1b	Positive definite	
	<code>nag_sym_mat_inv</code> (4.2)	Computes the inverse of a real or complex, symmetric or Hermitian matrix
	<code>nag_sym_mat_inv_fac</code> (4.2)	Computes the inverse of a real or complex, symmetric or Hermitian matrix, with the matrix previously factorized using <code>nag_sym_lin_fac</code>
	<code>nag_sym_lin_sol</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations with one or many right-hand sides
	<code>nag_sym_lin_fac</code> (5.2)	Performs a Cholesky or Bunch–Kaufman factorization of a real or complex, symmetric or Hermitian matrix
	<code>nag_sym_lin_sol_fac</code> (5.2)	Solves a real or complex, symmetric or Hermitian system of linear equations, with coefficient matrix previously factorized by <code>nag_sym_lin_fac</code>
D2d2	Positive definite banded	
	<code>nag_sym_bnd_lin_sol</code> (5.5)	Solves a real symmetric or complex Hermitian positive definite banded system of linear equations, with one or many right-hand sides
	<code>nag_sym_bnd_lin_fac</code> (5.5)	Performs a Cholesky factorization of a real symmetric or complex Hermitian positive definite band matrix
	<code>nag_sym_bnd_lin_sol_fac</code> (5.5)	Solves a real symmetric or complex Hermitian positive definite banded system of linear equations, with coefficient matrix previously factorized by <code>nag_sym_bnd_lin_fac</code>

D2e	Associated operations (e.g., matrix reorderings)	
	<code>nag_sparse_mat_init_coo</code> (4.3)	Initializes a sparse matrix data structure from COO format
	<code>nag_sparse_mat_init_csc</code> (4.3)	Initializes a sparse matrix data structure from CSC format
	<code>nag_sparse_mat_init_csr</code> (4.3)	Initializes a sparse matrix data structure from CSR format
	<code>nag_sparse_mat_init_dia</code> (4.3)	Initializes a sparse matrix data structure from DIA format
	<code>nag_sparse_mat_extract</code> (4.3)	Extracts details of a sparse matrix from a structure of type <code>nag_sparse_mat_real_wp</code> or <code>nag_sparse_mat_cplx_wp</code>
	<code>nag_sparse_matvec</code> (4.3)	Matrix–vector multiply for a sparse matrix
D3	Determinants	
D3a	Real nonsymmetric matrices	
D3a1	General	
	<code>nag_gen_lin_fac</code> (5.1)	Performs an <i>LU</i> factorization of a general real or complex matrix
D3a2	Banded	
	<code>nag_gen_bnd_lin_fac</code> (5.4)	Performs an <i>LU</i> factorization of a general real or complex band matrix
D3a3	Triangular	
	<code>nag_tri_mat_det</code> (5.3)	Evaluates the determinant of a real or complex triangular matrix
D3b	Real symmetric matrices	
D3b1	General	
D3b1a	Indefinite	
	<code>nag_sym_lin_fac</code> (5.2)	Performs a Cholesky or Bunch–Kaufman factorization of a real or complex, symmetric or Hermitian matrix
D3b1b	Positive definite	
	<code>nag_sym_lin_fac</code> (5.2)	Performs a Cholesky or Bunch–Kaufman factorization of a real or complex, symmetric or Hermitian matrix
D3b2	Positive definite banded	
	<code>nag_sym_bnd_lin_fac</code> (5.5)	Performs a Cholesky factorization of a real symmetric or complex Hermitian positive definite band matrix
D3c	Complex non-Hermitian matrices	
D3c1	General	
	<code>nag_gen_lin_fac</code> (5.1)	Performs an <i>LU</i> factorization of a general real or complex matrix
D3c2	Banded	
	<code>nag_gen_bnd_lin_fac</code> (5.4)	Performs an <i>LU</i> factorization of a general real or complex band matrix
D3c3	Triangular	
	<code>nag_tri_mat_det</code> (5.3)	Evaluates the determinant of a real or complex triangular matrix
D3d	Complex Hermitian matrices	
D3d1	General	
D3d1a	Indefinite	
	<code>nag_sym_lin_fac</code> (5.2)	Performs a Cholesky or Bunch–Kaufman factorization of a real or complex, symmetric or Hermitian matrix
D3d1b	Positive-definite	
	<code>nag_sym_lin_fac</code> (5.2)	Performs a Cholesky or Bunch–Kaufman factorization of a real or complex, symmetric or Hermitian matrix

D3d2	Positive-definite banded <code>nag_sym_bnd_lin_fac</code> (5.5)	Performs a Cholesky factorization of a real symmetric or complex Hermitian positive definite band matrix
D4	Eigenvalues, eigenvectors	
D4a	Ordinary eigenvalue problems ($Ax = \lambda x$)	
D4a1	Real symmetric <code>nag_sym_eig_sel</code> (6.1)	Selected eigenvalues, and optionally the corresponding eigenvectors, of a real symmetric or complex Hermitian matrix
	<code>nag_sym_eig_all</code> (6.1)	All eigenvalues, and optionally eigenvectors, of a real symmetric or complex Hermitian matrix
D4a2	Real nonsymmetric <code>nag_schur_fac</code> (6.2)	Schur factorization of a general real or complex matrix
	<code>nag_nsym_eig_all</code> (6.2)	All eigenvalues, and optionally eigenvectors, of a general real or complex matrix
D4a3	Complex Hermitian <code>nag_sym_eig_all</code> (6.1)	All eigenvalues, and optionally eigenvectors, of a real symmetric or complex Hermitian matrix
D4a4	Complex non-Hermitian <code>nag_schur_fac</code> (6.2)	Schur factorization of a general real or complex matrix
	<code>nag_nsym_eig_all</code> (6.2)	All eigenvalues, and optionally eigenvectors, of a general real or complex matrix
D4a5	Tridiagonal <code>nag_sym_tridiag_eig_all</code> (6.1)	All eigenvalues, and optionally eigenvectors, of a real symmetric tridiagonal matrix
	<code>nag_sym_tridiag_eig_val</code> (6.1)	Selected eigenvalues of a real symmetric tridiagonal matrix
	<code>nag_sym_tridiag_eig_vec</code> (6.1)	Selected eigenvectors of a real symmetric tridiagonal matrix
D4b	Generalized eigenvalue problems (e.g., $Ax = \lambda Bx$)	
D4b1	Real symmetric <code>nag_sym_gen_eig_all</code> (6.5)	All eigenvalues, and optionally eigenvectors, of a real symmetric-definite or complex Hermitian-definite generalized eigenvalue problem
	<code>nag_sym_gen_eig_sel</code> (6.5)	Selected eigenvalues, and optionally the corresponding eigenvectors, of a real symmetric-definite or complex Hermitian-definite generalized eigenvalue problem
D4b2	Real general <code>nag_nsym_gen_eig_all</code> (6.6)	All eigenvalues, and optionally eigenvectors, of a real or complex nonsymmetric generalized eigenvalue problem
	<code>nag_gen_schur_fac</code> (6.6)	Generalized Schur factorization of a real or complex matrix pencil
D4b3	Complex Hermitian <code>nag_sym_gen_eig_all</code> (6.5)	All eigenvalues, and optionally eigenvectors, of a real symmetric-definite or complex Hermitian-definite generalized eigenvalue problem
	<code>nag_sym_gen_eig_sel</code> (6.5)	Selected eigenvalues, and optionally the corresponding eigenvectors, of a real symmetric-definite or complex Hermitian-definite generalized eigenvalue problem
D4b4	Complex general	

	<code>nag_nsym_gen_eig_all</code> (6.6)	All eigenvalues, and optionally eigenvectors, of a real or complex nonsymmetric generalized eigenvalue problem
	<code>nag_gen_schur_fac</code> (6.6)	Generalized Schur factorization of a real or complex matrix pencil
D4c	Associated operations	
D4c1	Transform problem	
D4c1b	Reduce to compact form	
D4c1b1	Tridiagonal	
	<code>nag_sym_tridiag_reduc</code> (6.1)	Reduction of a real symmetric or complex Hermitian matrix to real symmetric tridiagonal form
	<code>nag_sym_tridiag_orth</code> (6.1)	Form or apply the transformation matrix determined by <code>nag_sym_tridiag_reduc</code>
D4c2	Compute eigenvalues of matrix in compact form	
D4c2a	Tridiagonal	
	<code>nag_sym_tridiag_eig_all</code> (6.1)	All eigenvalues, and optionally eigenvectors, of a real symmetric tridiagonal matrix
	<code>nag_sym_tridiag_eig_val</code> (6.1)	Selected eigenvalues of a real symmetric tridiagonal matrix
D4c3	Form eigenvectors from eigenvalues	
	<code>nag_sym_tridiag_eig_vec</code> (6.1)	Selected eigenvectors of a real symmetric tridiagonal matrix
D4c4	Back transform eigenvectors	
	<code>nag_sym_tridiag_orth</code> (6.1)	Form or apply the transformation matrix determined by <code>nag_sym_tridiag_reduc</code>
D5	QR decomposition, Gram–Schmidt orthogonalization	
	<code>nag_qr_fac</code> (6.4)	QR factorization of a general real or complex matrix
	<code>nag_qr_orth</code> (6.4)	Form or apply the matrix Q determined by <code>nag_qr_fac</code>
D6	Singular value decomposition	
	<code>nag_gen_svd</code> (6.3)	Singular value decomposition of a general real or complex matrix
	<code>nag_gen_bidiag_reduc</code> (6.3)	Reduction of a general real or complex matrix to real bidiagonal form
	<code>nag_bidiag_svd</code> (6.3)	Singular value decomposition of a real bidiagonal matrix
D9	Singular, overdetermined or underdetermined systems of linear equations, generalized inverses	
D9a	Unconstrained	
D9a1	Least squares (L_2) solution	
	<code>nag_lin_lsq_sol</code> (6.4)	Solves a real or complex linear least-squares problem
	<code>nag_lin_lsq_sol_svd</code> (6.4)	Solves a real or complex linear least-squares problem, assuming that a singular value decomposition of the coefficient matrix has already been computed
	<code>nag_lin_lsq_sol_qr</code> (6.4)	Solves a real or complex linear least-squares problem, assuming that the QR factorization of the coefficient matrix has already been computed
	<code>nag_lin_lsq_sol_qr_svd</code> (6.4)	Solves a real or complex linear least-squares problem using the SVD, assuming that the QR factorization of the coefficient matrix has already been computed
E	Interpolation	
E1	Univariate data (curve fitting)	
E1a	Polynomial splines (piecewise polynomials)	

		<code>nag_spline_1d_interp</code> (8.2)	Generates a cubic spline interpolant to an arbitrary 1-d data set
		<code>nag_pch_monot_interp</code> (8.1)	Generates a monotonicity-preserving piecewise cubic Hermite interpolant
		<code>nag_spline_1d_lsqr_fit</code> (8.2)	Generates a weighted least-squares cubic spline fit to an arbitrary 1-d data set, with given interior knots
E2	Multivariate data (surface fitting)		
E2a	Gridded		
		<code>nag_spline_2d_interp</code> (8.3)	Generates a bicubic spline interpolating surface through a set of data values, given on a rectangular grid of the <i>xy</i> plane
E2b	Scattered		
		<code>nag_scatter_2d_interp</code> (8.4)	Generates a 2-d interpolating function using a modified Shepard method
		<code>nag_scatter_3d_interp</code> (8.4)	Generates a 3-d interpolating function using a modified Shepard method
E3	Service routines for interpolation		
E3a	Evaluation of fitted functions, including quadrature		
E3a1	Function evaluation		
		<code>nag_pch_eval</code> (8.1)	Computes values and optionally derivatives of a piecewise cubic Hermite interpolant
		<code>nag_spline_1d_eval</code> (8.2)	Computes values of a cubic spline and optionally its first three derivatives
		<code>nag_spline_2d_eval</code> (8.3)	Computes values of a bicubic spline
		<code>nag_scatter_2d_eval</code> (8.4)	Computes values of the interpolant generated by <code>nag_scatter_2d_interp</code> and its partial derivatives
		<code>nag_scatter_3d_eval</code> (8.4)	Computes values of the interpolant generated by <code>nag_scatter_3d_interp</code> and its partial derivatives
		<code>nag_cheb_1d_eval</code> (8.5)	Evaluation of fitted polynomial in one variable, from Chebyshev series form
E3a2	Derivative evaluation		
		<code>nag_pch_eval</code> (8.1)	Computes values and optionally derivatives of a piecewise cubic Hermite interpolant
		<code>nag_spline_1d_eval</code> (8.2)	Computes values of a cubic spline and optionally its first three derivatives
		<code>nag_scatter_2d_eval</code> (8.4)	Computes values of the interpolant generated by <code>nag_scatter_2d_interp</code> and its partial derivatives
		<code>nag_scatter_3d_eval</code> (8.4)	Computes values of the interpolant generated by <code>nag_scatter_3d_interp</code> and its partial derivatives
		<code>nag_cheb_1d_deriv</code> (8.5)	Derivatives of fitted polynomial in Chebyshev series form
E3a3	Quadrature		
		<code>nag_pch_intg</code> (8.1)	Computes the definite integral of a piecewise cubic Hermite interpolant
		<code>nag_spline_1d_intg</code> (8.2)	Computes the definite integral of a cubic spline
		<code>nag_spline_2d_intg</code> (8.3)	Computes the definite integral of a bicubic spline
		<code>nag_cheb_1d_intg</code> (8.5)	Integral of fitted polynomial in Chebyshev series form
E3d	Other		
		<code>nag_pch_extract</code> (8.1)	Extracts details of a piecewise cubic Hermite interpolant from a structure of type <code>nag_pch_comm_wp</code>
		<code>nag_spline_1d_set</code> (8.2)	Initializes a cubic spline with given interior knots and B-spline coefficients
		<code>nag_spline_1d_extract</code> (8.2)	Extracts details of a cubic spline from a structure of type <code>nag_spline_1d_comm_wp</code>
		<code>nag_spline_2d_set</code> (8.3)	Initializes a bicubic spline with given interior knots and B-spline coefficients
		<code>nag_spline_2d_extract</code> (8.3)	Extracts details of a bicubic spline from a structure of type <code>nag_spline_2d_comm_wp</code>

	<code>nag_scatter_2d_set</code> (8.4)	Initializes a structure of type <code>nag_scatter_comm_wp</code> to represent a 2-d scattered data interpolant
	<code>nag_scatter_3d_set</code> (8.4)	Initializes a structure of type <code>nag_scatter_comm_wp</code> to represent a 3-d scattered data interpolant
	<code>nag_scatter_extract</code> (8.4)	Extracts details of a scattered data interpolant from a structure of derived type <code>nag_scatter_comm_wp</code>
F	Solution of nonlinear equations	
F1	Single equation	
F1a	Polynomial	
F1a1	Real coefficients	
	<code>nag_polynom_roots</code> (10.1)	Calculates the roots of a polynomial
F1a2	Complex coefficients	
	<code>nag_polynom_roots</code> (10.1)	Calculates the roots of a polynomial
F1b	Nonpolynomial	
	<code>nag_nlin_eqn_sol</code> (10.2)	Finds a solution of a single nonlinear equation
F2	System of equations	
	<code>nag_nlin_sys_sol</code> (10.3)	Finds a solution of a system of nonlinear equations
G	Optimization (<i>search also classes K, L8</i>)	
G1	Unconstrained	
G1a	Univariate	
G1a1	Smooth function	
G1a1a	User provides no derivatives	
	<code>nag_uv_min_sol</code> (9.5)	Finds the minimum of a continuous function of a single variable in a given finite interval
G1a1b	User provides first derivatives	
	<code>nag_uv_min_sol</code> (9.5)	Finds the minimum of a continuous function of a single variable in a given finite interval
G2	Constrained	
G2a	Linear programming	
G2a1	Dense matrix of constraints	
	<code>nag_qp_sol</code> (9.1)	Solves a linear or quadratic programming problem
G2c	Integer programming	
G2c1	Zero/one	
	<code>nag_ip_sol</code> (19.1)	Solves 'zero-one', 'general', 'mixed' or 'all' integer linear programming problems
G2c6	Pure integer programming	
	<code>nag_ip_sol</code> (19.1)	Solves 'zero-one', 'general', 'mixed' or 'all' integer linear programming problems
G2d	Network (<i>for network reliability search class M</i>)	
G2d1	Shortest path	
	<code>nag_short_path_find</code> (19.2)	Finds the shortest path through a directed or undirected acyclic network
G2e	Quadratic programming	
G2e1	Positive definite Hessian (i.e., convex problem)	

	nag_qp_sol (9.1)	Solves a linear or quadratic programming problem
G2e2	Indefinite Hessian	
	nag_qp_sol (9.1)	Solves a linear or quadratic programming problem
G2h	General nonlinear programming	
G2h1	Simple bounds	
G2h1a	Smooth function	
G2h1a1	User provides no derivatives	
	nag_nlp_sol (9.3)	Solves a dense nonlinear programming problem
	nag_con_nlin_lsqr_sol (9.4)	Finds a constrained minimum of a sum of squares
	nag_con_nlin_lsqr_sol_1 (9.4)	Finds a constrained minimum of a sum of squares
	nag_nlp_sparse_sol (9.6)	Solves a sparse nonlinear programming problem
G2h1a2	User provides first derivatives	
	nag_nlp_sol (9.3)	Solves a dense nonlinear programming problem
	nag_con_nlin_lsqr_sol (9.4)	Finds a constrained minimum of a sum of squares
	nag_con_nlin_lsqr_sol_1 (9.4)	Finds a constrained minimum of a sum of squares
	nag_nlp_sparse_sol (9.6)	Solves a sparse nonlinear programming problem
G2h2	Linear equality or inequality constraints	
G2h2a	Smooth function	
G2h2a1	User provides no derivatives	
	nag_nlp_sol (9.3)	Solves a dense nonlinear programming problem
	nag_con_nlin_lsqr_sol (9.4)	Finds a constrained minimum of a sum of squares
	nag_con_nlin_lsqr_sol_1 (9.4)	Finds a constrained minimum of a sum of squares
	nag_nlp_sparse_sol (9.6)	Solves a sparse nonlinear programming problem
G2h2a2	User provides first derivatives	
	nag_nlp_sol (9.3)	Solves a dense nonlinear programming problem
	nag_con_nlin_lsqr_sol (9.4)	Finds a constrained minimum of a sum of squares
	nag_con_nlin_lsqr_sol_1 (9.4)	Finds a constrained minimum of a sum of squares
	nag_nlp_sparse_sol (9.6)	Solves a sparse nonlinear programming problem
G2h3	Nonlinear constraints	
G2h3a	Equality constraints only	
G2h3a1	Smooth function and constraints	
	nag_nlp_sol (9.3)	Solves a dense nonlinear programming problem
	nag_con_nlin_lsqr_sol (9.4)	Finds a constrained minimum of a sum of squares
	nag_con_nlin_lsqr_sol_1 (9.4)	Finds a constrained minimum of a sum of squares
	nag_nlp_sparse_sol (9.6)	Solves a sparse nonlinear programming problem
G2h3b	Equality and inequality constraints	

G2h3b1	Smooth function and constraints	
G2h3b1a	User provides no derivatives	
	<code>nag_con_nlin_lsq_sol</code> (9.4)	Finds a constrained minimum of a sum of squares
G2h3b1b	User provides first derivatives of function and constraints	
	<code>nag_con_nlin_lsq_sol</code> (9.4)	Finds a constrained minimum of a sum of squares
G4	Service routines	
G4c	Check user-supplied derivatives	
	<code>nag_nlin_lsq_sol</code> (9.2)	Finds an unconstrained minimum of a sum of squares
G4d	Find feasible point	
	<code>nag_qp_sol</code> (9.1)	Solves a linear or quadratic programming problem
	<code>nag_con_nlin_lsq_sol</code> (9.4)	Finds a constrained minimum of a sum of squares
G4f	Other	
	<code>nag_qp_cntrl_init</code> (9.1)	Initialization procedure for <code>nag_qp_cntrl_wp</code>
	<code>nag_nlin_lsq_cntrl_init</code> (9.2)	Initialization procedure for <code>nag_nlin_lsq_cntrl_wp</code>
	<code>nag_nlp_cntrl_init</code> (9.3)	Initialization procedure for <code>nag_nlp_cntrl_wp</code>
	<code>nag_con_nlin_lsq_cntrl_init</code> (9.4)	Initialization procedure for <code>nag_con_nlin_lsq_cntrl_wp</code>
	<code>nag_nlp_sparse_cntrl_init</code> (9.6)	Initialization procedure for <code>nag_nlp_sparse_cntrl_wp</code>
H	Differentiation, integration	
H2	Quadrature (numerical evaluation of definite integrals)	
H2a	One-dimensional integrals	
H2a1	Finite interval (general integrand)	
H2a1a	Integrand available via user-defined procedure	
H2a1a1	Automatic (user need only specify required accuracy)	
	<code>nag_quad_1d_gen</code> (11.1)	1-d quadrature, adaptive, finite interval, allowing for badly behaved integrand, allowing for singularities at user-specified break-points, suitable for oscillatory integrands
H2a1b	Integrand available only on grid	
H2a1b2	Nonautomatic	
	<code>nag_quad_1d_data</code> (11.1)	1-d quadrature, integration of function defined by data values, Gill–Miller method
H2a2	Finite interval (specific or special type integrand including weight functions, oscillating and singular integrands, principal value integrals, splines, etc.)	
H2a2a	Integrand available via user-defined procedure	
H2a2a1	Automatic (user need only specify required accuracy)	
	<code>nag_quad_1d_gen</code> (11.1)	1-d quadrature, adaptive, finite interval, allowing for badly behaved integrand, allowing for singularities at user-specified break-points, suitable for oscillatory integrands
	<code>nag_quad_1d_wt_trig</code> (11.1)	1-d quadrature, adaptive, finite interval, weight function $\cos(\omega x)$ or $\sin(\omega x)$
	<code>nag_quad_1d_wt_end_sing</code> (11.1)	1-d quadrature, adaptive, finite interval, weight function with end-point singularities of algebraico-logarithmic type

	<code>nag_quad_1d_wt_hilb</code> (11.1)	1-d quadrature, adaptive, finite interval, weight function $1/(x - c)$, Cauchy principal value (Hilbert transform)
H2a2b	Integrand available only on grid	
H2a2b1	Automatic (user need only specify required accuracy)	
	<code>nag_spline_1d_intg</code> (8.2)	Computes the definite integral of a cubic spline
H2a3	Semi-infinite interval (including e^{-x} weight function)	
H2a3a	Integrand available via user-defined procedure	
H2a3a1	Automatic (user need only specify required accuracy)	
	<code>nag_quad_1d_inf_gen</code> (11.2)	1-d quadrature, adaptive, semi-infinite or infinite interval
	<code>nag_quad_1d_inf_wt_trig</code> (11.2)	1-d quadrature, adaptive, semi-infinite interval, weight function $\cos(\omega x)$ or $\sin(\omega x)$
H2a4	Infinite interval (including $\exp(-x^2)$ weight function)	
H2a4a	Integrand available via user-defined procedure	
H2a4a1	Automatic (user need only specify required accuracy)	
	<code>nag_quad_1d_inf_gen</code> (11.2)	1-d quadrature, adaptive, semi-infinite or infinite interval
H2b	Multidimensional integrals	
H2b1	One or more hyper-rectangular regions (includes iterated integrals)	
H2b1a	Integrand available via user-defined procedure	
H2b1a1	Automatic (user need only specify required accuracy)	
	<code>nag_quad_md_rect_mintg</code> (11.3)	Multi-dimensional adaptive quadrature over a hyper-rectangle, multiple integrands
	<code>nag_quad_md_rect</code> (11.3)	Multi-dimensional adaptive quadrature over a hyper-rectangle
	<code>nag_quad_2d</code> (11.3)	2-d quadrature, finite region
	<code>nag_quad_monte_carlo</code> (11.3)	Multi-dimensional quadrature over hyper-rectangle, Monte-Carlo method
H2c	Service routines (e.g., compute weights and nodes for quadrature formulas)	
	<code>nag_quad_gs_wt_absc</code> (11.4)	Calculation of weights and abscissae for Gaussian quadrature rules, general choice of rule
I	Differential and integral equations	
I1	Ordinary differential equations (ODE's)	
I1a	Initial value problems	
I1a1	General, nonstiff or mildly stiff	
I1a1a	One-step methods (e.g., Runge-Kutta)	
	<code>nag_rk_interval</code> (12.1)	Integrates across an interval and provides the solution at user-specified points
	<code>nag_rk_step</code> (12.1)	Integrates one step at a time
I1a2	Stiff and mixed algebraic- differential equations	
	<code>nag_pde_parab_1d_fd</code> (13.3)	Integrates a system of parabolic PDEs in one space variable, coupled with ODEs; using finite differences for the spatial discretisation with optional automatic adaptive spatial remeshing
I1c	Service routines (e.g., interpolation of solutions, error handling, test programs)	
	<code>nag_rk_setup</code> (12.1)	Sets up the integration
	<code>nag_rk_reset_end</code> (12.1)	Resets the end point of integration

	<code>nag_rk_interp</code> (12.1)	Interpolates the solution
	<code>nag_rk_info</code> (12.1)	Provides statistics about the integration
	<code>nag_rk_global_err</code> (12.1)	Provides information about global error assessment
I2	Partial differential equations	
I2a	Initial boundary value problems	
I2a1	Parabolic	
I2a1a	One spatial dimension	
	<code>nag_pde_parab_1d_coll</code> (13.3)	Integrates a system of parabolic PDEs in one space variable, coupled with ODEs; using a Chebyshev C^0 collocation method for the spatial discretisation
	<code>nag_pde_interp_1d_coll</code> (13.3)	Interpolates the solution and first derivative of a system of partial differential equations solved using a Chebyshev C^0 collocation method, at a set of user-specified points
	<code>nag_pde_interp_1d_fd</code> (13.3)	Interpolates the solution and first derivative of a system of partial differential equations solved using finite differences, at a set of user-specified points
I2b	Elliptic boundary value problems	
I2b1	Linear	
I2b1a	Second order	
I2b1a1	Poisson (Laplace) or Helmholtz equation	
I2b1a1a	Rectangular domain (or topologically rectangular in the coordinate system)	
	<code>nag_pde_helm_3d</code> (13.1)	Solves the 3-d Helmholtz equation using a standard seven-point finite difference scheme and a fast Fourier transform method
I2b1a3	Nonseparable problems	
	<code>nag_pde_ell_rect</code> (13.2)	Generates a seven-diagonal system of linear equations which arises from the discretization of a two-dimensional elliptic partial differential equation on a rectangle
I2b4	Service routines	
	<code>nag_pde_ell_rect</code> (13.2)	Generates a seven-diagonal system of linear equations which arises from the discretization of a two-dimensional elliptic partial differential equation on a rectangle
	<code>nag_pde_interp_1d_coll</code> (13.3)	Interpolates the solution and first derivative of a system of partial differential equations solved using a Chebyshev C^0 collocation method, at a set of user-specified points
	<code>nag_pde_interp_1d_fd</code> (13.3)	Interpolates the solution and first derivative of a system of partial differential equations solved using finite differences, at a set of user-specified points
I2b4a	Domain triangulation (<i>search also class P</i>)	
	<code>nag_pde_ell_mg_sol</code> (13.2)	Solves a seven-diagonal system of linear equations using a multigrid iteration
J	Integral transforms	
J1	Trigonometric transforms including fast Fourier transforms	
J1a	One-dimensional	
J1a1	Real	
	<code>nag_fft_1d_basic</code> (7.1)	Single or multiple 1-d real, Hermitian or complex discrete Fourier transform, which is overwritten on the input data

		<code>nag_fft_1d_real</code> (7.1)	Single or multiple 1-d real or Hermitian discrete Fourier transform, or its inverse
J1a2	Complex		
		<code>nag_fft_1d</code> (7.1)	Single or multiple 1-d complex discrete Fourier transform, or its inverse
		<code>nag_fft_1d_basic</code> (7.1)	Single or multiple 1-d real, Hermitian or complex discrete Fourier transform, which is overwritten on the input data
		<code>nag_fft_1d_real</code> (7.1)	Single or multiple 1-d real or Hermitian discrete Fourier transform, or its inverse
		<code>nag_fft_cos</code> (7.2)	Single or multiple 1-d discrete Fourier cosine transform
		<code>nag_fft_sin</code> (7.2)	Single or multiple 1-d discrete Fourier sine transform
		<code>nag_fft_qtr_cos</code> (7.2)	Single or multiple 1-d discrete quarter-wave Fourier cosine transform, or its inverse
		<code>nag_fft_qtr_sin</code> (7.2)	Single or multiple 1-d discrete quarter-wave Fourier sine transform, or its inverse
		<code>nag_conj_herm</code> (7.1)	Complex conjugates of Hermitian sequences
		<code>nag_herm_to_cmplx</code> (7.1)	Convert Hermitian sequences to general complex sequences
		<code>nag_cmplx_to_herm</code> (7.1)	Convert Hermitian complex sequences to their compact real form
J1b	Multidimensional		
		<code>nag_fft_2d</code> (7.1)	2-d complex discrete Fourier transform, or its inverse
		<code>nag_fft_2d_basic</code> (7.1)	2-d complex discrete Fourier transform, which is overwritten on the input data
		<code>nag_fft_3d</code> (7.1)	3-d complex discrete Fourier transform, or its inverse
		<code>nag_fft_3d_basic</code> (7.1)	3-d complex discrete Fourier transform, which is overwritten on the input data
J2	Convolutions		
		<code>nag_fft_conv</code> (7.3)	Computes the convolution or correlation of two real or complex vectors
J4	Hilbert transforms		
		<code>nag_quad_1d_wt_hilb</code> (11.1)	1-d quadrature, adaptive, finite interval, weight function $1/(x - c)$, Cauchy principal value (Hilbert transform)
K	Approximation (<i>search also class L8</i>)		
K1	Least squares (L_2) approximation		
K1a	Linear least squares (<i>search also classes D5, D6, D9</i>)		
K1a1	Unconstrained		
K1a1a	Univariate data (curve fitting)		
K1a1a1	Polynomial splines (piecewise polynomials)		
		<code>nag_spline_1d_lsq_fit</code> (8.2)	Generates a weighted least-squares cubic spline fit to an arbitrary 1-d data set, with given interior knots
		<code>nag_spline_1d_auto_fit</code> (8.2)	Generates a cubic spline approximation to an arbitrary 1-d data set, with automatic knot selection
K1a1b	Multivariate data (surface fitting)		
		<code>nag_spline_2d_lsq_fit</code> (8.3)	Generates a minimal, weighted least-squares bicubic spline surface fit to a given set of data points, with given interior knots
		<code>nag_spline_2d_auto_fit</code> (8.3)	Generates a bicubic spline approximation to a 2-d data set, with automatic knot selection
K1b	Nonlinear least squares		
K1b1	Unconstrained		

K1b1a	Smooth functions	
K1b1a1	User provides no derivatives	
	<code>nag_nlin_lsq_sol</code> (9.2)	Finds an unconstrained minimum of a sum of squares
K1b1a2	User provides first derivatives	
	<code>nag_nlin_lsq_sol</code> (9.2)	Finds an unconstrained minimum of a sum of squares
K1b1a3	User provides first and second derivatives	
	<code>nag_nlin_lsq_sol</code> (9.2)	Finds an unconstrained minimum of a sum of squares
K1b2	Constrained	
K1b2b	Nonlinear constraints	
	<code>nag_con_nlin_lsq_sol</code> (9.4)	Finds a constrained minimum of a sum of squares
K6	Service routines for approximation	
K6a	Evaluation of fitted functions, including quadrature	
K6a1	Function evaluation	
	<code>nag_spline_1d_eval</code> (8.2)	Computes values of a cubic spline and optionally its first three derivatives
	<code>nag_spline_2d_eval</code> (8.3)	Computes values of a bicubic spline
	<code>nag_cheb_1d_eval</code> (8.5)	Evaluation of fitted polynomial in one variable, from Chebyshev series form
K6a2	Derivative evaluation	
	<code>nag_spline_1d_eval</code> (8.2)	Computes values of a cubic spline and optionally its first three derivatives
	<code>nag_cheb_1d_deriv</code> (8.5)	Derivatives of fitted polynomial in Chebyshev series form
K6a3	Quadrature	
	<code>nag_spline_1d_intg</code> (8.2)	Computes the definite integral of a cubic spline
	<code>nag_spline_2d_intg</code> (8.3)	Computes the definite integral of a bicubic spline
	<code>nag_cheb_1d_intg</code> (8.5)	Integral of fitted polynomial in Chebyshev series form
K6d	Other	
	<code>nag_spline_1d_set</code> (8.2)	Initializes a cubic spline with given interior knots and B-spline coefficients
	<code>nag_spline_1d_extract</code> (8.2)	Extracts details of a cubic spline from a structure of type <code>nag_spline_1d_comm_wp</code>
	<code>nag_spline_2d_set</code> (8.3)	Initializes a bicubic spline with given interior knots and B-spline coefficients
	<code>nag_spline_2d_extract</code> (8.3)	Extracts details of a bicubic spline from a structure of type <code>nag_spline_2d_comm_wp</code>
	<code>nag_nlin_lsq_cov</code> (9.2)	Computes the variance-covariance matrix for a nonlinear least-squares problem
	<code>nag_nlin_lsq_cntrl_init</code> (9.2)	Initialization procedure for <code>nag_nlin_lsq_cntrl_wp</code>
L	Statistics, probability	
L1	Data summarization	
L1a	One-dimensional data	
L1a1	Raw data	
	<code>nag_summary_stats_1v</code> (22.1)	Computes basic descriptive statistics for univariate data
L1a3	Grouped data	

	<code>nag_summary_stats_1v</code> (22.1)	Computes basic descriptive statistics for univariate data
L1c	Multi-dimensional data	
L1c1	Raw data	
L1c1b	Covariance, correlation	
	<code>nag_prod_mom_correl</code> (25.2)	Calculates the variance-covariance matrix and the Pearson product-moment correlation coefficients for a set of data
	<code>nag_part_correl</code> (25.2)	Calculates the partial variance-covariance matrix and the partial correlation matrix from a correlation or variance covariance matrix
L5	Function evaluation (<i>search also class C</i>)	
L5a	Univariate	
L5a1	Cumulative distribution functions, probability density functions	
L5a1b	Beta, binomial	
	<code>nag_binom_prob</code> (20.7)	Computes lower tail, upper tail or point probability for a binomial distribution with parameters n and p
	<code>nag_beta_prob</code> (20.5)	Computes lower or upper tail probability for a beta distribution with parameters a and b
L5a1c	Cauchy, χ^2	
	<code>nag_chisq_prob</code> (20.3)	Computes lower or upper tail probability for a χ^2 -distribution with ν degrees of freedom
L5a1e	Error function, exponential, extreme value	
	<code>nag_erfc</code> (3.3)	Complementary error function $\text{erfc } x$
	<code>nag_erf</code> (3.3)	Error function $\text{erf } x$
L5a1f	F -distribution	
	<code>nag_f_prob</code> (20.4)	Computes lower or upper tail probability for an F -distribution with ν_1 and ν_2 degrees of freedom
L5a1g	Gamma, general, geometric	
	<code>nag_gamma_prob</code> (20.6)	Computes lower or upper tail probability for a gamma distribution with shape parameter a and scale parameter b
L5a1h	Halfnormal, hypergeometric	
	<code>nag_hypergeo_prob</code> (20.7)	Computes lower tail, upper tail or point probability for a hypergeometric distribution with parameters n , l , and m
L5a1n	Negative binomial, normal	
	<code>nag_normal_prob</code> (20.1)	Computes probabilities for various parts of a univariate Normal distribution
L5a1p	Pareto, Poisson	
	<code>nag_poisson_prob</code> (20.7)	Computes lower tail, upper tail or point probability for a Poisson distribution with parameter λ
L5a1t	t -distribution	
	<code>nag_t_prob</code> (20.2)	Computes probabilities for various parts of a Student's t -distribution with ν degrees of freedom
L5a2	Inverse distribution functions, sparsity functions	
L5a2b	Beta, binomial	
	<code>nag_beta_deviate</code> (20.5)	Computes the deviate associated with a given lower tail probability of a beta distribution with parameters a and b

L5a2c	Cauchy, χ^2 <code>nag_chisq_deviate</code> (20.3)	Computes the deviate associated with a given lower tail probability of a χ^2 -distribution with ν degrees of freedom
L5a2f	F -distribution <code>nag_f_deviate</code> (20.4)	Computes the deviate associated with a given lower tail probability of an F -distribution with ν_1 and ν_2 degrees of freedom
L5a2g	Gamma, general, geometric <code>nag_gamma_deviate</code> (20.6)	Computes the deviate associated with a given lower tail probability of a gamma distribution with shape parameter a and scale parameter b
L5a2n	Negative binomial, normal, normal order statistics <code>nag_normal_deviate</code> (20.1)	Computes the deviate associated with a given probability of a standard Normal distribution
L5a2t	t -distribution <code>nag_t_deviate</code> (20.2)	Computes the deviate associated with a given probability of a Student's t -distribution
L5b	Multivariate	
L5b1	Cumulative multivariate distribution functions, probability density functions	
L5b1n	Normal <code>nag_bivar_normal_prob</code> (20.1) <code>nag_mv_normal_prob</code> (20.1)	Computes the lower tail probability for a bivariate Normal distribution Computes probabilities for various parts of a multivariate Normal distribution
L6	Random number generation	
L6a	Univariate <code>nag_rand_ref_vec</code> (21.3)	Generates random integers from a discrete distribution, using a reference vector
L6a2	Beta, binomial, Boolean <code>nag_rand_binom</code> (21.3) <code>nag_rand_beta</code> (21.2)	Generates random integers from a binomial distribution and/or returns a reference vector for the distribution Generates random numbers from a beta distribution with parameters a and b
L6a5	Exponential, extreme value <code>nag_rand_neg_exp</code> (21.2)	Generates random numbers from a (negative) exponential distribution with mean a
L6a7	Gamma, general (continuous, discrete), geometric <code>nag_rand_user_dist</code> (21.3) <code>nag_rand_gamma</code> (21.2)	Generates random integers and/or returns a reference vector from a discrete distribution defined in terms of its PDF or CDF Generates random numbers from a gamma distribution with parameters a and b
L6a8	Halfnormal, hypergeometric <code>nag_rand_hypergeo</code> (21.3)	Generates random integers from an hypergeometric distribution and/or returns a reference vector for the distribution
L6a14	Negative binomial, normal, normal order statistics <code>nag_rand_normal</code> (21.2) <code>nag_rand_neg_binom</code> (21.3)	Generates random numbers from a Normal distribution with mean a and standard deviation b Generates random integers from a negative binomial distribution and/or returns a reference vector for the distribution

L6a21	Uniform (continuous, discrete), uniform order statistics <code>nag_rand_uniform</code> (21.2)	Generates random numbers from a uniform distribution over (a, b)
L6b	Multivariate	
L6b14	Normal <code>nag_rand_mv_normal</code> (21.2)	Generates a vector of n random numbers from a multivariate Normal distribution with mean vector a and covariance matrix C
L6c	Service routines (e.g., seed) <code>nag_rand_seed_set</code> (21.1)	Sets the seed used by random number generating procedures to give a repeatable or non-repeatable sequence of random numbers
L7	Analysis of variance (including analysis of covariance)	
L7f	Generate experimental designs <code>nag_mult_lin_reg</code> (25.1)	Performs a general multiple linear regression analysis for any given predictor variables and a response variable
L8	Regression (<i>search also classes D5, D6, D9, G, K</i>)	
L8a	Simple linear (i.e., $y = b_0 + b_1x$) (<i>search also class L8h</i>)	
L8a1	Ordinary least squares	
L8a1a	Parameter estimation	
L8a1a1	Unweighted data <code>nag_simple_lin_reg</code> (25.1)	Performs a simple linear regression analysis for a pair of related variables
L8c	Multiple linear (i.e., $y = b_0 + b_1x_1 + \dots + b_px_p$)	
L8c1	Ordinary least squares	
L8c1b	Parameter estimation (<i>search also class L8c1a</i>)	
L8c1b1	Using raw data <code>nag_mult_lin_reg</code> (25.1)	Performs a general multiple linear regression analysis for any given predictor variables and a response variable
L8e	Nonlinear (i.e., $y = F(X, b)$) (<i>search also class L8h</i>)	
L8e1	Ordinary least squares	
L8e1b	Parameter estimation (<i>search also class L8e1a</i>)	
L8e1b1	Unweighted data, user provides no derivatives <code>nag_nlin_lsq_cov</code> (9.2)	Computes the variance-covariance matrix for a nonlinear least-squares problem
	<code>nag_nlin_lsq_sol</code> (9.2)	Finds an unconstrained minimum of a sum of squares
	<code>nag_con_nlin_lsq_sol</code> (9.4)	Finds a constrained minimum of a sum of squares
	<code>nag_con_nlin_lsq_sol.1</code> (9.4)	Finds a constrained minimum of a sum of squares
L8e1b2	Unweighted data, user provides derivatives <code>nag_nlin_lsq_sol</code> (9.2)	Finds an unconstrained minimum of a sum of squares
	<code>nag_con_nlin_lsq_sol</code> (9.4)	Finds a constrained minimum of a sum of squares
	<code>nag_con_nlin_lsq_sol.1</code> (9.4)	Finds a constrained minimum of a sum of squares

L8g	Spline (i.e., piecewise polynomial)	
	<code>nag_spline_1d_lsq_fit</code> (8.2)	Generates a weighted least-squares cubic spline fit to an arbitrary 1-d data set, with given interior knots
	<code>nag_spline_1d_auto_fit</code> (8.2)	Generates a cubic spline approximation to an arbitrary 1-d data set, with automatic knot selection
L10	Time series analysis (<i>search also class J</i>)	
L10a	Univariate (<i>search also classes L3a6 and L3a7</i>)	
L10a2	Time domain analysis	
L10a2a	Summary statistics	
L10a2a1	Autocorrelations and autocovariances	
	<code>nag_tsa_acf</code> (29.1)	Calculates the sample autocorrelation function of a univariate time series
L10a2a2	Partial autocorrelations	
	<code>nag_tsa_pacf</code> (29.1)	Calculates the sample partial autocorrelation function of a univariate time series
L10a2c1	Model identification	
	<code>nag_tsa_pacf</code> (29.1)	Calculates the sample partial autocorrelation function of a univariate time series
L10a3	Frequency domain analysis (<i>search also class J1</i>)	
L10a3a	Spectral analysis	
L10a3a3	Spectrum estimation using the periodogram	
	<code>nag_spectral_cov</code> (29.3)	Calculates the smoothed sample spectrum of a univariate time series using autocovariances data
L10a3a4	Spectrum estimation using the Fourier transform of the autocorrelation function	
	<code>nag_spectral_data</code> (29.3)	Calculates the smoothed sample spectrum of a univariate time series
L10b	Two time series (<i>search also classes L3b3c, L10c, and L10d</i>)	
L10b3	Frequency domain analysis (<i>search also class J1</i>)	
L10b3a	Cross-spectral analysis	
L10b3a3	Cross-spectrum estimation using the cross-periodogram	
	<code>nag_bivar_spectral_cov</code> (29.3)	Calculates the smoothed sample cross spectrum of a bivariate time series using autocovariances data
L10b3a4	Cross-spectrum estimation using the Fourier transform of the cross-correlation or cross-covariance function	
	<code>nag_bivar_spectral_data</code> (29.3)	Calculates the smoothed sample cross spectrum of a bivariate time series
L10b3a6	Spectral functions	
	<code>nag_bivar_spectral_coh</code> (29.3)	Calculates the squared coherency, the cross amplitude, the gain and the phase spectra
	<code>nag_bivar_spectral_lin_sys</code> (29.3)	Calculates the noise spectrum and the impulse response function from a linear system
L10c	Multivariate time series (<i>search also classes J1, L3e3 and L10b</i>)	
	<code>nag_kalman_init</code> (29.2)	Provides an initial estimate of the Kalman filter state covariance matrix
	<code>nag_kalman_predict</code> (29.2)	Calculates a one step prediction for the square root covariance Kalman filter
	<code>nag_kalman_sqrt_cov_var</code> (29.2)	Calculates a time-varying square root covariance Kalman filter
	<code>nag_kalman_sqrt_cov_invar</code> (29.2)	Calculates a time-invariant square root covariance Kalman filter
L12	Discriminant analysis	
	<code>nag_canon_var</code> (28.2)	Performs canonical variate analysis

L13	Covariance structure models	
L13a	Factor analysis	
	<code>nag_orthomax</code> (28.3)	Computes orthogonal rotation, using a generalized orthomax rotations
L13b	Principal components analysis	
	<code>nag_prin_comp</code> (28.1)	Performs principal component analysis
L13c	Canonical correlation	
	<code>nag_canon_var</code> (28.2)	Performs canonical variate analysis
N	Data handling (<i>search also class L2</i>)	
N1	Input, output	
	<code>nag_write_gen_mat</code> (1.3)	Writes a real, complex or integer general matrix
	<code>nag_write_tri_mat</code> (1.3)	Writes a real or complex triangular matrix
	<code>nag_write_bnd_mat</code> (1.3)	Writes a real or complex band matrix
N6	Sorting	
N6a	Internal	
N6a1	Passive (i.e., construct pointer array, rank)	
	<code>nag_rank_arb_data</code> (1.4)	Ranks arbitrary data according to a user-supplied comparison procedure
N6a1a	Integer	
	<code>nag_rank_vec</code> (1.4)	Ranks a vector of numeric or character data in ascending or descending order
	<code>nag_rank_mat</code> (1.4)	Ranks the rows or columns of a matrix of integer or real numbers in ascending or descending order
N6a1b	Real	
	<code>nag_rank_vec</code> (1.4)	Ranks a vector of numeric or character data in ascending or descending order
	<code>nag_rank_mat</code> (1.4)	Ranks the rows or columns of a matrix of integer or real numbers in ascending or descending order
N6a1c	Character	
	<code>nag_rank_vec</code> (1.4)	Ranks a vector of numeric or character data in ascending or descending order
N6a2	Active	
N6a2a	Integer	
	<code>nag_sort_vec</code> (1.4)	Sorts a vector of numeric or character data into ascending or descending order
N6a2b	Real	
	<code>nag_sort_vec</code> (1.4)	Sorts a vector of numeric or character data into ascending or descending order
N6a2c	Character	
	<code>nag_sort_vec</code> (1.4)	Sorts a vector of numeric or character data into ascending or descending order
N8	Permuting	
	<code>nag_reorder_vec</code> (1.4)	Reorders a vector of numeric or character data into the order specified by a vector of ranks
	<code>nag_invert_perm</code> (1.4)	Inverts a permutation, thus converts a rank vector to an index vector, or vice-versa
	<code>nag_check_perm</code> (1.4)	Checks the validity of a permutation
	<code>nag_decomp_perm</code> (1.4)	Decomposes a permutation into cycles, as an aid to reordering ranked data

R	Service routines	
	nag_lib_ident (1.1)	Prints details of the Library implementation
	nag_deallocate (1.1)	Deallocates storage from structures with types defined by the Library
R1	Machine-dependent constants	
	nag_pi (1.5)	Returns an approximation to π
	nag_euler_constant (1.5)	Returns an approximation to γ (Euler's constant)
R3	Error handling	
R3c	Other utilities	
	nag_set_error (1.2)	Controls how errors are to be handled by the Library

References

- [1] Boisvert R F, Howe S E and Kahaner D K (1990) The guide to available mathematical software problem classification scheme *Report NISTIR 4475* Applied and Computational Mathematics Division, National Institute of Standards and Technology
- [2] Boisvert R F, Howe S E and Kahaner D K (1985) GAMS — a framework for the management of scientific software *ACM Trans. Math. Software* **11** 313–355
- [3] Boisvert R F (1989) The guide to available mathematical software advisory system *Math. Comput. Simul.* **31** 453–464