GAMS Index for the NAG Fortran 90 Library

This index classifies NAG *fl*90 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 (*search also class L5*)

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 erfc x

`nag_erf` (3.3) Error function 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 $Y_1(x)$

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

G2h3b1 Smooth function and constraints
G2h3b1a User provides no derivatives
`nag_con_nlin_lsq_sol` (9.4) Finds a constrained minimum of a sum of squares
G2h3b1b User provides first derivatives of function and constraints
`nag_con_nlin_lsq_sol` (9.4) Finds a constrained minimum of a sum of squares
G4 Service routines
G4c Check user-supplied derivatives
`nag_nlin_lsq_sol` (9.2) Finds an unconstrained minimum of a sum of squares
G4d Find feasible point
`nag_qp_sol` (9.1) Solves a linear or quadratic programming problem
`nag_con_nlin_lsq_sol` (9.4) Finds a constrained minimum of a sum of squares
G4f Other
`nag_qp_cntrl_init` (9.1) Initialization procedure for `nag_qp_cntrl_`*wp*
`nag_nlin_lsq_cntrl_init` (9.2) Initialization procedure for `nag_nlin_lsq_cntrl_`*wp*
`nag_nlp_cntrl_init` (9.3) Initialization procedure for `nag_nlp_cntrl_`*wp*
`nag_con_nlin_lsq_cntrl_init` (9.4) Initialization procedure for `nag_con_nlin_lsq_cntrl_`*wp*
`nag_nlp_sparse_cntrl_init` (9.6) Initialization procedure for `nag_nlp_sparse_cntrl_`*wp*
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)
`nag_quad_1d_gen` (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
`nag_quad_1d_data` (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)
`nag_quad_1d_gen` (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
`nag_quad_1d_wt_trig` (11.1) 1-d quadrature, adaptive, finite interval, weight function $\cos(\omega x)$ or $\sin(\omega x)$
`nag_quad_1d_wt_end_sing` (11.1) 1-d quadrature, adaptive, finite interval, weight function with end-point singularities of algebraico-logarithmic type

L8g Spline (i.e., piecewise polynomial)

	`nag_spline_1d_lsq_fit` (8.2)	Generates a weighted least-squares cubic spline fit to an arbitrary 1-d data set, with given interior knots
	`nag_spline_1d_auto_fit` (8.2)	Generates a cubic spline approximation to an arbitrary 1-d data set, with automatic knot selection

L10 Time series analysis (*search also class J*)

L10a Univariate (*search also classes L3a6 and L3a7*)

L10a2 Time domain analysis

L10a2a Summary statistics

L10a2a1 Autocorrelations and autocovariances

	`nag_tsa_acf` (29.1)	Calculates the sample autocorrelation function of a univariate time series

L10a2a2 Partial autocorrelations

	`nag_tsa_pacf` (29.1)	Calculates the sample partial autocorrelation function of a univariate time series

L10a2c1 Model identification

	`nag_tsa_pacf` (29.1)	Calculates the sample partial autocorrelation function of a univariate time series

L10a3 Frequency domain analysis (*search also class J1*)

L10a3a Spectral analysis

L10a3a3 Spectrum estimation using the periodogram

	`nag_spectral_cov` (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

	`nag_spectral_data` (29.3)	Calculates the smoothed sample spectrum of a univariate time series

L10b Two time series (*search also classes L3b3c, L10c, and L10d*)

L10b3 Frequency domain analysis (*search also class J1*)

L10b3a Cross-spectral analysis

L10b3a3 Cross-spectrum estimation using the cross-periodogram

	`nag_bivar_spectral_cov` (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

	`nag_bivar_spectral_data` (29.3)	Calculates the smoothed sample cross spectrum of a bivariate time series

L10b3a6 Spectral functions

	`nag_bivar_spectral_coh` (29.3)	Calculates the squared coherency, the cross amplitude, the gain and the phase spectra
	`nag_bivar_spectral_lin_sys` (29.3)	Calculates the noise spectrum and the impulse response function from a linear system

L10c Multivariate time series (*search also classes J1, L3e3 and L10b*)

	`nag_kalman_init` (29.2)	Provides an initial estimate of the Kalman filter state covariance matrix
	`nag_kalman_predict` (29.2)	Calculates a one step prediction for the square root covariance Kalman filter
	`nag_kalman_sqrt_cov_var` (29.2)	Calculates a time-varying square root covariance Kalman filter
	`nag_kalman_sqrt_cov_invar` (29.2)	Calculates a time-invariant square root covariance Kalman filter

L12 Discriminant analysis

	`nag_canon_var` (28.2)	Performs canonical variate analysis

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