NAG C Library, Mark 5 News

1. Introduction

Mark 5 of the NAG C Library sees some major additions to the optimization chapter: nag_opt_lin_lsq (e04ncc) solves linear least-squares and convex quadratic programming (QP) problems; nag_opt_nlin_lsq (e04unc) is a variant of the existing nag_opt_nlp (e04ucc) function, specialised for nonlinear least-squares problems; and nag_opt_sparse_convex_qp (e04nkc) solves large sparse linear programming (LP) and QP problems.

Other additions to this chapter are nag_opt_one_var_no_deriv (e04abc) and nag_opt_one_var_deriv (e04bbc), for minimizing a function of a single variable, and a utility function nag_opt_estimate_deriv (e04xac) which can be used to compute an approximation to the gradient vector and/or Hessian matrix from the objective function supplied to nag_opt_nlp (e04ucc).

In the related operations research chapter, an integer programming (IP) function, nag_ip_bb (h02bbc), has been provided. This solves IP problems with linear or quadratic objective functions, using a branch and bound method.

The statistics coverage of the NAG C Library has shown a healthy growth in response to our user requirements. This is continued at the present Mark with the introduction of Chapter g03 on multi-variate analysis. This chapter covers principal component, co-ordinate and factor analysis, amongst others. We have also introduced functions for the analysis of variance (Chapter g04): two functions are available for factorial and block designs.

Linear algebra coverage has been significantly enhanced by the introduction of Chapter f11 on the solution of sparse linear equations both for symmetrical and non-symmetrical cases. Finally we have also provided functionality for the computation of selected eigenvalues and eigenvectors for general real and complex matrices.

The total number of user-callable functions in the C Library (including linear algebra support routines and complex number functions) is now 389.

2. Multi-threading

In the development of the C Library we have avoided using global variables except in Chapter g05 (Random Number Generators). At Mark 5 this software has been revised with the introduction of POSIX calls (or their Microsoft equivalent) to ensure that the random number generators work correctly in a multi-threaded environment.

With this revision and some other minor changes to the library software, we can now say with confidence that the NAG C Library is thread-safe.

Another modification has been made at Mark 5 which helps the user to call the library in a thread-safe manner. This is the introduction of a `void*` parameter to the user-defined function parameters in Chapter c05 (Roots of Transcendental Equations) and Chapter d01 (Quadrature). At earlier Marks, if a user wished to communicate between the calling program and the user-defined function, this could only be achieved by introducing global variables, which can compromise thread-safety unless handled carefully. The new `void*` parameters, similar to those which already existed for user-defined functions in the ODE (d02) and optimization (e04) chapters, now allow communication to take place without requiring the use of global variables.

The original versions of these c05 and d01 functions may well be removed at a later mark of the C Library. The affected functions are:

`c05adc` nag_zero_cont_func_bd
Zero of a continuous function of one variable

`c05nbc` nag_zero_nonlin_eqns
Solution of a system of nonlinear equations (function values only)

`c05pbc` nag_zero_nonlin_eqns_deriv
Solution of a system of nonlinear equations (using first derivatives)

`c05zbc` nag_check_deriv
Derivative checker for nag_zero_nonlin_eqns_deriv (c05pbc)

`d01ajc` nag_1d_quad_gen
1-D adaptive quadrature, allowing for badly-behaved integrands
`d01akc` nag_1d_quad_osc
1-D adaptive quadrature, suitable for oscillating functions
`d01alc` nag_1d_quad_brkpts
1-D adaptive quadrature, allowing for singularities at specified points
`d01amc` nag_1d_quad_inf
1-D adaptive quadrature over infinite or semi-infinite interval
`d01anc` nag_1d_quad_wt_trig
1-D adaptive quadrature, finite interval, sine or cosine weight functions
`d01apc` nag_1d_quad_wt_alglog
1-D adaptive quadrature, weight function with end-point singularities of algebraic-logarithmic type
`d01aqc` nag_1d_quad_wt_cauchy
1-D adaptive quadrature, weight function $1/(x-c)$, Cauchy principal value
`d01asc` nag_1d_quad_inf_wt_trig
1-D adaptive quadrature, semi-infinite interval, sine or cosine weight function
`d01bac` nag_1d_quad_gauss
1-D Gaussian quadrature rule evaluation
`d01fcc` nag_multid_quad_adapt
Multi-dimensional adaptive quadrature
`d01gbc` nag_multid_quad_monte_carlo
Multi-dimensional quadrature, using Monte Carlo method

3. Year 2000 Compliance

The only references to date and time utilities in the NAG C Library are internal to the random number generators. Examination of the underlying algorithm confirms that it is unaffected by the change of millennium.

4. New Functions

`c05sdc` nag_zero_cont_func_bd_1
Zero of a continuous function of one variable, thread-safe
`c05tbc` nag_zero_nonlin_eqns_1
Solution of a system of nonlinear equations (function values only), thread-safe
`c05ubc` nag_zero_nonlin_eqns_deriv_1
Solution of a system of nonlinear equations (using first derivatives), thread-safe
`c05zcc` nag_check_deriv_1
Derivative checker for nag_zero_nonlin_eqns_deriv (c05pbc), thread-safe
`d01sjc` nag_1d_quad_gen_1
1-D adaptive quadrature, allowing for badly-behaved integrands, thread-safe
`d01skc` nag_1d_quad_osc_1
1-D adaptive quadrature, suitable for oscillating functions, thread-safe
`d01slc` nag_1d_quad_brkpts_1
1-D adaptive quadrature, allowing for singularities at specified points, thread-safe
`d01smc` nag_1d_quad_inf_1
1-D adaptive quadrature over infinite or semi-infinite interval, thread-safe
`d01snc` nag_1d_quad_wt_trig_1
1-D adaptive quadrature, finite interval, sine or cosine weight functions, thread-safe
`d01spc` nag_1d_quad_wt_alglog_1
1-D adaptive quadrature, weight function with end-point singularities of algebraic-logarithmic type, thread-safe
`d01sqc` nag_1d_quad_wt_cauchy_1
1-D adaptive quadrature, weight function $1/(x-c)$, Cauchy principal value, thread-safe
`d01ssc` nag_1d_quad_inf_wt_trig_1
1-D adaptive quadrature, semi-infinite interval, sine or cosine weight function, thread-safe
`d01tac` nag_1d_quad_gauss_1
1-D Gaussian quadrature rule evaluation, thread-safe

`d01wcc` nag_multid_quad_adapt_1
Multi-dimensional adaptive quadrature, thread-safe
`d01xbc` nag_multid_quad_monte_carlo_1
Multi-dimensional quadrature, using Monte Carlo method, thread-safe
`e02adc` nag_1d_cheb_fit
Computes the coefficients of a Chebyshev series polynomial for arbitrary data
`e02aec` nag_1d_cheb_eval
Evaluates the coefficients of a Chebyshev series polynomial
`e02afc` nag_1d_cheb_interp_fit
Computes the coefficients of a Chebyshev series polynomial for interpolated data
`e04abc` nag_opt_one_var_no_deriv
Minimizes a function of one variable, using function values only
`e04bbc` nag_opt_one_var_deriv
Minimizes a function of one variable, requires first derivatives
`e04hdc` nag_opt_check_2nd_deriv
Checks 2nd derivatives of a user-defined function.
`e04lbc` nag_opt_bounds_2nd_deriv
Solves bound constrained problems. 1st and 2nd derivatives are required.
`e04myc` nag_opt_sparse_mps_free
Free memory allocated by nag_opt_sparse_mps_read (e04mzc)
`e04mzc` nag_opt_sparse_mps_read
Read MPSX data for sparse LP or QP problem from a file
`e04ncc` nag_opt_lin_lsq
Solves linear least-squares and convex quadratic programming problems (non-sparse)
`e04nkc` nag_opt_sparse_convex_qp
Solves sparse linear programming or convex quadratic programming problems
`e04unc` nag_opt_nlin_lsq
Solves nonlinear least-squares problems using the sequential QP method
`e04xac` nag_opt_estimate_deriv
Computes an approximation to the gradient vector and/or the Hessian matrix for use with nag_opt_nlp (e04ucc) and other nonlinear optimization functions
`f02ecc` nag_real_eigensystem_sel
Computes selected eigenvalues and eigenvectors of a real general matrix
`f02gcc` nag_complex_eigensystem_sel
Computes selected eigenvalues and eigenvectors of a complex general matrix
`f11dac` nag_sparse_nsym_fac
Incomplete LU factorization (nonsymmetric)
`f11dcc` nag_sparse_nsym_fac_sol
Solver with incomplete LU preconditioning (nonsymmetric)
`f11dec` nag_sparse_nsym_sol
Solver with no/Jacobi/SSOR/preconditioning (nonsymmetric)
`f11jac` nag_sparse_sym_chol_fac
Incomplete Cholesky facroization (symmetric)
`f11jcc` nag_sparse_sym_chol_sol
Solver with incomplete Cholesky preconditioning (symmetric)
`f11jec` nag_sparse_sym_sol
Solver with Jacobi, SSOR, or no preconditioning (symmetric)
`f11zac` nag_sparse_nsym_sort
Sparse sort (nonsymmetric)
`f11zbc` nag_sparse_sym_sort
Sparse sort (symmetric)
`g03aac` nag_mv_prin_comp
Principal component analysis
`g03acc` nag_mv_canon_var
Canonical variate analysis
`g03adc` nag_mv_canon_corr
Canonical correlation analysis

`g03bac` nag_mv_orthomax
Orthogonal rotations for loading matrix
`g03bcc` nag_mv_procustes
Procrustes rotations
`g03cac` nag_mv_factor
Maximum likelihood estimates of parameters
`g03ccc` nag_mv_fac_score
Factor score coefficients, following nag_mv_factor (g03cac)
`g03dac` nag_mv_discrim
Test for equality of wthin-group covariance matrices
`g03dbc` nag_mv_discrim_mahaldist
Mahalanobis squared distances, following nag_mv_discrim (g03dac)
`g03dcc` nag_mv_discrim_group
Allocation of observations to groups, following nag_mv_discrim (g03dac)
`g03eac` nag_mv_distance_mat
Compute distance (dissimilarity) matrix
`g03ecc` nag_mv_hierar_cluster_analysis
Performs hierarchical cluster analysis
`g03efc` nag_mv_kmeans_cluster_analysis
K-means
`g03ehc` nag_mv_dendrogram
Construct dendogram following nag_mv_hierar_cluster_analysis (g03ecc)
`g03ejc` nag_mv_cluster_indicator
Construct clusters following nag_mv_hierar_cluster_analysis (g03ecc)
`g03fac` nag_mv_prin_coord_analysis
Principal co-ordinate analysis
`g03fcc` nag_mv_ordinal_multidimscale
Multidimensional scaling
`g03xzc` nag_mv_dend_free
Frees memory allocated to the dendrogram array in nag_mv_dendrogram (g03ehc)
`g03zac` nag_mv_z_scores
Standardize values of a data matrix
`g04bbc` nag_anova_random
General block design or completely randomized design
`g04cac` nag_anova_factorial
Complete factorial design
`g04czc` nag_anova_factorial_free
Complete factorial design
`h02bbc` nag_ip_bb
Solves integer programming problems using a branch and bound method
`h02buc` nag_ip_mps_read
Read MPSX data for IP, LP or QP problem from a file
`h02bvc` nag_ip_mps_free
Free memory allocated by nag_ip_mps_read (h02buc)
`h02xxc` nag_ip_init
Initialize option structure to null values
`h02xyc` nag_ip_read
Read optional parameter values from a file
`h02xzc` nag_ip_free
Free NAG allocated memory from option structures