AD-A092 034 ATOMIC ENERGY RESEARCH ESTABLISHMENT HARWELL (ENGLAND) F/G 12/1
HARWELL SUBROUTINE LIBRARY. A CATALOGUE OF SUBROUTINES (1973). --ETC(U)
AUG 77 M J HOPPER
UNCLASSIFIED AERE-R-7477-SUPPL-2 NL

AERE-R-7477-Suppl-2
Supplement No. 2

HARWELL SUBROUTINE LIBRARY.

A CATALOGUE OF SUBROUTINES (1973). SUPPLEMENT ~~NO~~. 2.

Compiled by M. J. Hopper

August 1977

This supplement lists the new subroutines that have been introduced into the Harwell Subroutine Library in the period from August 1974 to July 1977.

Computer Science and Systems Division,
Atomic Energy Research Establishment,
Harwell, Oxfordshire, ENGLAND.

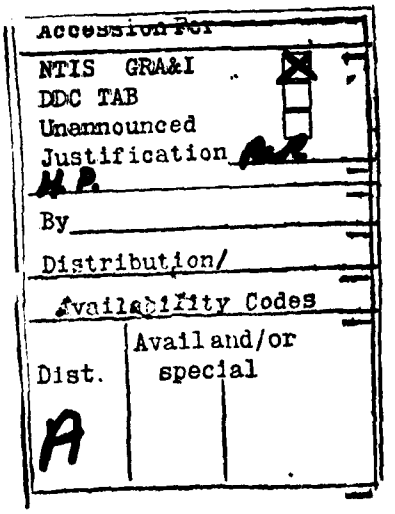
PREFACE

This is a supplement to the Harwell Subroutine Library's catalogue R.7477 and gives a list of the subroutines introduced into the library in the period from August 1974 to July 1977. We have not included a general index to the the library this time. The total list for the whole library is now covered by the report R.7477 and its two supplements (1974) and (1977) plus the deletion of the subroutines EB01A/AD, EB04A/AD, LA01A/AD, MC13A/C, OE01A/B, PA01A/AD, PA02A/AD, and TG01A/AD.

Many of our new subroutines have been written with portability in mind and it is the intention of some of our authors to write code that conforms to the Fortran IV ANSI standard (1966). To do this we use the Bell Laboratories' verifier program, described in Ryder, 'The PFORT Verifier', Software Practice and Experience, 4, (1974), to check out the new subroutines. We have indicated those subroutines which have passed the verifier test by specifying the language as 'Fortan IV (standard)'. There are many other new subroutines that we are not flagged as verified which are virtually standard but have failed the test on some small point, e.g. OE02A is standard except for a single use of the IBM END= parameter in a READ statement, and in many cases a standard version can be obtained using OE04A to process the source deck and these we have marked '(standard available)'.

Again, with portability in mind, we have begun to include Fortran versions of our machine language subroutines on the source tape and these we have flagged by specifying the language as 370/BAL (Fortran available).

Harwell Subroutine Librarian
M. J. Hopper
August 1977



EA12A

Given a **real symmetric matrix** $\mathbf{A} = \{a_{ij}\}$ of order n, **calculates the** k **largest eigenvalues,** $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_k$ and their associated eigenvectors $\mathbf{x}_i$ $i=1,2,\ldots,k$, where $\mathbf{A}\mathbf{x}_i = \lambda_i \mathbf{x}_i$.

The subroutine uses the method of simultaneous iteration and also allows the user to take advantage of sparsity. Eigensolutions from other parts of the spectrum can be obtained by using shifts of the form $\mathbf{A} - \beta \mathbf{I}$ optionally combined with inversion.

Versions: EA12A, EA12AD.
Calls: EA13A, FA01AS, FM01AS, FM02AS.
Language: Fortran IV (standard).
Date: June 1976.
Size: 8.1K; 932 cards.
Origin: I.S.Duff, A.E.R.E., Harwell.

EA13A

Given a **real symmetric matrix A , finds all its eigenvalues** λ_i **and eigenvectors** $\mathbf{x}_i$ $i=1,2,\ldots,n$, where $\mathbf{A}\mathbf{x}_i = \lambda_i \mathbf{x}_i$.

The classical Jacobi method is used.

Remark: Supersedes EA03A.

Versions: EA13A, EA13AD.
Language: Fortran IV (standard).
Date: June 1976.
Size: 3.3K; 168 cards.
Origin: I.S.Duff, A.E.R.E., Harwell.

FM01AS

To **evaluate the inner product of two real vectors,** given vectors $\mathbf{a} = \{a_i\}$ and $\mathbf{b} = \{b_i\}$ of length n, evaluates

$$w = \sum_{i=1}^{n} a_i b_i$$

The subroutine is a *FUNCTION* routine written in machine code and is fast in execution.

Remark: Supersedes MC02AS.

Versions: FM01AS, FM01AD, FM01AQ.*
Language: 370/BAL (Fortran available).
Date: January 1975.
Size: .1K; 105 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

FM02AS

To **evaluate the inner product of two real vectors** when the elements of each vector are stored at some fixed displacement from neighbouring elements. Given vectors $\mathbf{a} = \{a_i\}$ and $\mathbf{b} = \{b_i\}$ of length n evaluates

$$w = \sum_{i=1}^{n} a_i b_i$$

The subroutine can be used to evaluate inner products involving rows of multi-dimensioned arrays. It is a *FUNCTION* subroutine written in machine language and is fast in execution.

Remark: Supersedes MC03AS.

Versions: FM02AS, FM02AD, FM02AQ.*
Language: 370/BAL (Fortran available).
Date: April 1975.
Size: .1K; 125 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

FM03AS

To **evaluate the triple inner product** of three real vectors, given vectors $\mathbf{a} = \{a_i\}$, $\mathbf{b} = \{b_i\}$ and $\mathbf{c} = \{c_i\}$ of length n, evaluates

$$w = \sum_{i=1}^{n} a_i b_i c_i$$

The subroutine is a *FUNCTION* routine written in machine code and is fast in execution.

Remark: Supersedes MC05AS.

Versions: FM03AS, FM03AD, FM03AQ.*
Language: 370/BAL (Fortran available).
Date: May 1975.
Size: .1K; 115 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

FM04AS

To **evaluate the triple inner product of three real vectors** when the elements of each vector are stored at some fixed displacement from neighbouring elements. Given vectors $\mathbf{a} = \{a_i\}$, $\mathbf{b} = \{b_i\}$ and $\mathbf{c} = \{c_i\}$ of length n, evaluates

$$w = \sum_{i=1}^{n} a_i b_i c_i$$

* Q versions give extended precision.

The subroutine is a *FUNCTION* routine written in machine code and is fast in execution.

Versions: FM04AS, FM04AD, FM04AQ.*
Language: 370/BAL (Fortran available).
Date: May 1975.
Size: .2K; 148 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

FM05AS

To **evaluate the inner product of two complex vectors,** given complex vectors $\mathbf{a} = \{a_i\}$ and $\mathbf{b} = \{b_i\}$ of length n, evaluates

$$u = \sum_{i=1}^{n} a_i b_i$$

or optionally evaluates

$$v = \sum_{i=1}^{n} a_i \bar{b}_i$$

where the $\bar{b}_i$s are the complex conjugates of the b_is.

The subroutine is a *FUNCTION* routine written in machine code and is fast in execution.

Versions: FM05AS, FM05AD, FM05AQ.*
Language: 370/BAL (Fortran available).
Date: June 1975.
Size: .2K; 150 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

FM06AS

To **evaluate the inner product of two complex vectors** when the elements of each vector are stored at some fixed displacement from neighbouring elements. Given complex vectors $\mathbf{a} = \{a_i\}$ and $\mathbf{b} = \{b_i\}$ of length n, evaluates

$$u = \sum_{i=1}^{n} a_i b_i$$

or optionally evaluates

$$v = \sum_{i=1}^{n} a_i \bar{b}_i$$

where the $\bar{b}_i$s are the complex conjugates of the b_is.

The subroutine can be used to evaluate inner products involving rows of multi-dimensioned arrays. It is a *FUNCTION* subroutine written in machine language and is fast in execution.

Remark: Supersedes ME06AS.

Versions: FM06AS, FM06AD, FM06AQ.*
Language: 370/BAL (Fortran available).
Date: June 1975.
Size: .2K; 180 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

LA01B

Solves the **linear programming problem,** finds $\mathbf{x} = \{x_j\}_n$ which minimizes the linear function

$$f(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j$$

subject to the linear constraints

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i \qquad i=1,2,\ldots,k$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i=k+1,\ldots,m$$

where $x_j \geqslant 0 \quad i=1,2,\ldots,n.$

The Revised Simplex method is used where an inverse of the basis is maintained and updated at each iteration. There is an option to allow the basis inverse to be *cleaned up* at the end of the calculation and the solution checked for ill-conditioning and if necessary the calculation is restarted. Another option allows the user to force specific columns of the LP tableau into the initial basis.

LA01B returns an error indicator and an estimate of the ill-conditioning. Several print options are offered.

Remark: Supersedes LA01A.

Versions: LA01B, LA01BD.
Calls: FM01AS.
Language: Fortran IV.
Date: January 1975.
Size: 8.1K; 684 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

LA05A

To **factorize a matrix, solve** corresponding systems of **linear equations** and **update the factorization** when a column of the matrix is altered, exploiting **sparsity** in all cases. Its primary application is likely to be for handling **linear programming** bases.

* Q versions give extended precision.

The subroutine decomposes the matrix into triangular factors using sparse matrix techniques similar to those described in Curtis and Reid, Harwell report R.6844, (1971).

Versions: LA05A, LA05AD.
Calls: MC20A.
Language: Fortran IV.
Date: December 1975.
Size: 20.6K; 959 cards.
Origin: J.K.Reid, A.E.R.E., Harwell.

MA25A

To **calculate the minimax solution** to a system of **linear equations subject to simple bounds** on the solution variables. Calculates the vector $\mathbf{x} = \{x_j\}_n$ that minimizes

$$\max_{1 \leqslant i \leqslant m} \left| \sum_{j=1}^{n} a_{ij} x_j + b_i \right|$$

subject to $|x_j| \leqslant g$, $j=1,2,\ldots,n$, where n need not be equal to m.

More general conditions can be applied by making a transformation of the variables and a bound on the accuracy of the maximum is returned.

The method is described in Powell, Harwell report CSS 11, (1974) and is based on the exchange algorithm with a modification which can short-cut some iterations.

Versions: MA25A, MA25AD.
Language: Fortran IV.
Date: December 1974.
Size: 7.5K; 370 cards.
Origin: M.J.D.Powell, Harwell.†

MA26A

To **solve a system of symmetric positive-definite tri-diagonal linear equations.** Given a matrix $\mathbf{T} = \{t_{ij}\}_{n \times n}$ where $t_{ij}=0$ when $i>j+1$ or $j>i+1$, solves the equations

$$\sum_{j=1}^{n} t_{ij} x_j = b_i \quad i=1,2,\ldots,n$$

The matrix is stored in a compact form and the subroutine may be re-entered with additional right hand sides to solve further equations which have the same matrix **T**.

The method uses the Choleski decomposition $\mathbf{T} = \mathbf{L}\mathbf{L}^T$.

Versions: MA26A, MA26AD.
Language: Fortran IV (standard).
Date: May 1975.
Size: 1.7K; 157 cards.
Origin: W.R.Owen, Univ. of Queensland, Australia.

MA28A

To **solve a sparse system of linear equations.** Given a sparse matrix $\mathbf{A} = \{a_{ij}\}_{n \times n}$ this subroutine **decomposes A into factors,** solves $\mathbf{Ax} = \mathbf{b}$ (or optionally $\mathbf{A}^T\mathbf{x} = \mathbf{b}$). It will decompose a new matrix having the same sparsity pattern as a previous one by using the same pivotal sequence taking much less processing time than the original factorization. The matrix **A** is also allowed to be singular or rectangular.

The method is a variant of Gaussian elimination for sparse systems and further information is given in Duff, Harwell report R.8730, (1977).

Remark: Supersedes MA18A.

Versions: MA28A, MA28AD.
Calls: MA30A, MC20A, MC22A, MC23A, MC24A.
Language: Fortran IV (standard available).
Date: April 1977.
Size: 9.6K; 653 cards.
Origin: I.S.Duff, A.E.R.E., Harwell.

MA29A

To **factorize a real symmetric matrix** and optionally **solve the corresponding system of linear equations.** The matrix need not be positive-definite. Given a matrix $\mathbf{A} = \{a_{ij}\}_{n \times n}$ there are entries to factorize **A** (permuted) into the form $\mathbf{LDL}^T$, then given a factored **A**, to solve the system of linear equations

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad i=1,2,\ldots,n,$$

by forward and back substitution, and also an entry to perform both of these steps together.

The matrix is stored in a compact form taking full account of symmetry.

The method is described in Fletcher, 'Factorizing Symmetric Indefinite Matrices', J.Linear Algebra & Applics., **14**, 3, (1976).

† No longer at A.E.R.E. Harwell.

Versions: MA29A, MA29AD.
Language: Fortran IV.
Date: August 1975.
Size: 4.7K; 306 cards.
Origin: R.Fletcher, Dundee Univ.

MA30A

Given a **sparse matrix** $\mathbf{A} = \{a_{ij}\}_{n\times n}$, **performs the LU decomposition** of the diagonal blocks of a specified permutation of **A** and **solves the linear equations** $\mathbf{Ax} = \mathbf{b}$, i.e.

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad i=1,2,\ldots,n$$

or optionally solves $\mathbf{A}^T\mathbf{x} = \mathbf{b}$.

After decomposing a matrix it is possible to decompose other matrices with the same sparsity pattern with significant savings in processing time. Also, singular and rectangular matrices can be dealt with.

The method is a variant of Gaussian elimination for sparse matrices see, Duff and Reid, Harwell report CSS 48 (1977), and Duff, Harwell report R.8730 (1977).

Remark: MA28A provides a simple user-interface to MA30A.

Versions: MA30A, MA30AD.
Language: Fortran IV (standard available).
Date: June 1977.
Size: 16.9K; 1186 cards.
Origin: I.S.Duff, A.E.R.E., Harwell.

MB01C

Given a real matrix $\mathbf{A} = \{a_{ij}\}_{n\times n}$ **evaluates the inverse matrix** $\mathbf{A}^{-1}$.

The method is simple Gaussian elimination with row interchanges. No checks on the condition of the matrix are made and no indication of the accuracy of the inverse is given. If the user is concerned about such matters he should consider the alternative subroutines in the library.

Remark: supersedes MB01B (removes the restriction on the order n).

Versions: MB01C, MB01CD.
Calls: MC03AS.
Language: Fortran IV.
Date: February 1975.
Size: 2.6K; 91 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

MC13D

Given the pattern of nonzeros of a **sparse matrix A** , finds a **symmetric permutation** that makes the matrix block upper triangular, i.e. finds **P** such that $\mathbf{U} = \mathbf{PAP}^{-1}$ *is block upper triangular.*

The method is that of Tarjan, SIAM J. Computing, Vol 1, (1972), and is also described by Duff and Reid, Harwell report CSS 29 (1976).

Versions: MC13D.
Language: Fortran IV (standard available).
Date: March 1976.
Size: 2.2K; 131 cards.
Origin: I.S.Duff, A.E.R.E., Harwell.

MC16A

Appends an $n+1$ vector to an $n\times n$ triangular matrix to form an $(n+1)\times(n+1)$ triangular matrix, i.e. given an upper triangular matrix $\mathbf{U} = \{u_{ij}\}_{n\times n}$ and a vector $\mu = \{\mu_1, \mu_2, \ldots, \mu_{n+1}\}$ this subroutine forms the triangular matrix

$$\tilde{\mathbf{U}} = \left(\begin{array}{c|c} \mathbf{U} & \\ \cdots & \mu \\ \mathbf{0} & \end{array}\right)$$

Both $\tilde{\mathbf{U}}$ and **U** are stored in compact form.

Remark: the matrix storage format is that of MC11A.

Versions: MC16A, MC16AD.
Language: Fortran IV.
Date: September 1974.
Size: .7K; 29 cards.
Origin: M.J.D.Powell, A.E.R.E., Harwell.†

MC17A

To **delete a column from an $n\times n$ triangular matrix** to get $\mathbf{V} = \{v_{ij}\}_{n\times(n-1)}$ and return a related triangular matrix $\mathbf{U} = \{u_{ij}\}_{(n-1)\times(n-1)}$ such that $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V}$.

Both the original matrix and **U** are stored in a compact form.

---

† No longer at A.E.R.E. Harwell.

Remark: the matrix storage format is that of MC11A.

Versions: MC17A, MC17AD.
Calls: MC11A.
Language: Fortran IV.
Date: September 1974.
Size: .9K; 38 cards.
Origin: M.J.D.Powell, A.E.R.E., Harwell.†

MC18A

Given a **real symmetric matrix** A computes the **Householder transformation** $QAQ^T = T$ where T is symmetric tri-diagonal. Also given a vector x it will form Qx or optionally $Q^T x$.

The method is described in Ralston and Wilf, 'Mathematical Methods for Digital Computing', Vol. 1.

Versions: MC18A, MC18AD.
Calls: FM02AS.
Language: Fortran IV (standard).
Date: May 1975.
Size: 1K; 112 cards.
Origin: W.R.Owen, Univ. of Queensland.

MC19A

To **calculate scaling factors for a sparse matrix** $A = \{a_{ij}\}_{n\times n}$. They may be used, for instance, to scale the matrix prior to solving a corresponding set of linear equations, and are chosen so that the scaled matrix has its non-zeros near to unity in the sense that the sum of the squares of the logarithms of the non-zeros is minimized. The natural logarithms of the scaling factors $(r_i, c_j,\ i,j=1,2,\ldots,n)$ for the rows and columns are returned so that the scaled matrix has elements

$$b_{ij} = a_{ij}\exp(r_i + c_j).$$

The subroutine expects a square $n\times n$ matrix but tolerates rows or columns consisting entirely of zeros. Therefore a rectangular matrix can be handled by embedding it in a square matrix whose additional rows or columns consist entirely of zeros.

The method is described in Curtis and Reid, 'On the Automatic Scaling of Matrices for Gaussian Elimination'. J. Inst. Maths. Applics. (1972), **10**, pp. 118-124.

† No longer at A.E.R.E. Harwell.

Versions: MC19A, MC19AD.
Language: Fortran IV (standard available).
Date: March 1977.
Size: 2.8K; 167 cards.
Origin: J.K.Reid, A.E.R.E., Harwell.

MC20A

Sorts the nonzeros of a sparse matrix from arbitrary order to an ordering of columns or complete ordering of columns and rows.

The subroutine allows users with sparse problems to input the nonzeros in a natural order more suited to the individual problem. The resulting ordered matrix can subsequently be presented to other sparse matrix subroutines in the library.

An in-place sort algorithm is used which handles each item to be sorted exactly three times, so the number of operations for the method is of the order of the number of nonzeros.

Versions: MC20A, MC20AD.
Language: Fortran IV.
Date: November 1975.
Size: 1.6K; 103 cards.
Origin: J.K.Reid, A.E.R.E., Harwell.

MC21A

Given the pattern of nonzeros of a **sparse matrix** this subroutine attempts to **find a row permutation** that makes the matrix have **no zeros on its diagonal.**

The method used is a simple depth first search with a look ahead and is described by Duff, Harwell report CSS 49 (1977).

Versions: MC21A, MC21AD.
Language: Fortran IV (standard available).
Date: April 1977.
Size: 2.6K; 139 cards.
Origin: I.S.Duff, A.E.R.E., Harwell.

MC22A

Given a **sparse matrix** $A = \{a_{ij}\}_{n\times n}$ and a row permutation matrix P and a column permutation matrix Q, this subroutine **performs the permutation** $\bar{A} = PAQ$.

The nonzero elements of A are stored by rows in a compact form and the user defines the permutation matrices P and Q by index vectors of length n.

Versions: MC22A, MC22AD.
Language: Fortran IV (standard available).
Date: November 1976.
Size: 1.3K; 78 cards.
Origin: I.S.Duff, A.E.R.E., Harwell.

MC23A

Permutes a sparse matrix to a block lower triangular form, i.e. given a sparse matrix $\mathbf{A} = \{a_{ij}\}_{n\times n}$ it attempts to find permutation matrices **P** and **Q** such that $\tilde{\mathbf{A}} = \mathbf{PAQ}$ is block lower triangular and it returns $\tilde{\mathbf{A}}$ to the caller.

Versions: MC23A, MC23AD.
Calls: MC13D, MC21A.
Language: Fortran IV (standard available).
Date: April 1977.
Size: 3.2K; 202 cards.
Origin: I.S.Duff, A.E.R.E., Harwell.

MC24A

To **obtain an estimate of the largest element encountered during Gaussian elimination on a sparse matrix** $\mathbf{A} = \{a_{ij}\}_{n\times n}$. The subroutine is given the **LU** decomposition factors of **A** obtained from the elimination. If the matrix has been scaled this estimate will give an indication of the numerical accuracy of the decomposition.

The estimate is obtained by using Hadamard's inequality on the expression for the matrix elements in terms of the **LU** factors of the decomposition. Further details are given in Erisman and Reid, 'Monitoring the Stability of the Triangular Factorization of a Sparse Matrix', Numer. Math., 22 (1974).

Versions: MC24A, MC24AD.
Language: Fortran IV (standard available).
Date: March 1977.
Size: .9K; 46 cards.
Origin: I.S.Duff, A.E.R.E., Harwell.

NB02A

To **find a real zero of a continuous real function** $y(x)$ in a given interval $a\leqslant x\leqslant b$, i.e. solve the nonlinear equation in one variable

$$y(\xi) = 0 \quad a\leqslant\xi\leqslant b$$

The *limits* a and b must be chosen so that $sign\{y(a)\} \neq sign\{y(b)\}$ and the zero is then located by a combination of linear interpolation and binary subdivision. The user must supply code to evaluate $y(x)$ at any point in the interval and a limit can be specified for the number of function evaluations. The subroutine attempts to refine the bracket down to two points $x\leqslant\xi\leqslant x+\varepsilon$ where the accuracy parameter ε is specified by the user.

Remark: similar to NB01A except the convergence criterion is based on x not $y(x)$.

Versions: NB02A, NB02AD.
Language: Fortran IV.
Date: December 1975.
Size: 1.2K; 113 cards.
Origin: A.R.Curtis, A.E.R.E., Harwell.

OB02A

Provides a **simple graph drawing facility.** It is intended for very simple graphs and makes GHOST easier to use by standardizing options so that the user has less to remember, while still offering some choice.

There are four options

(a) take a new page and draw a pair of labelled axes (restricted to the positive quadrant).

(b) plot one of eight different symbols at a specified point.

(c) connect a sequence of points by straight lines, a kind of curve drawing option.

(d) print titles below the graph.

The subroutine uses the GHOST graphics package but mixing of GHOST subroutine calls and OB02A is not recommended.

Versions: OB02A.
Calls: IC01AS, and various GHOST subroutines.
Language: Fortran IV.
Date: November 1974.
Size: 3.8K; 159 cards.
Origin: A.R.Curtis and M.J.Hopper, A.E.R.E., Harwell.

OB13A

To **plot the part of a conic section that is inside a given triangle.** The conic section is defined by the equation $Q(x,y) = \eta$, where η is the contour height and where $Q(x,y)$ is a quadratic function that is specified by

its value at the vertices and midpoints of the triangle sides.

The subroutine uses the GHOST graphics package but communicates with GHOST through only one subroutine which makes it easily adapted for other graphics packages. The method is described in Marlow and Powell, 'A Fortran Subroutine for Plotting the Part of a Conic that is Inside a Given Triangle', Harwell report R.8336 (1976).

Versions: OB13A.
Calls: SHVECS (a GHOST subroutine).
Language: Fortran IV.
Date: March 1976.
Size: 8.4K; 370 cards.
Origin: S.Marlow, A.E.R.E., Harwell.

OB14A

To **draw contours of a function** $f(x,y)$ whose values are specified **over an** $n \times m$ **regular rectangular grid.** Specifically, the values of $f(x,y)$ must be specified at the grid points $(x_s + i\Delta x, y_s + j\Delta y)$ $0 \leqslant i \leqslant m-1$, $0 \leqslant j \leqslant n-1$, and where Δx and Δy define the uniform spacing of the grid. The contours at the boundary of the rectangular region can be improved if the user can provide extra values of $f(x,y)$ immediately outside the region, i.e. along any or all of the lines $x=x_s-\Delta x$, $x=x_s+m\Delta x$, $y=y_s-\Delta y$ and $y=y_s+n\Delta y$.

The subroutine uses the GHOST graphics package but draws only the contours and does not draw axes or perform mapping functions. It communicates with GHOST through the library subroutine OB13A and is easily adapted for other graphics packages.

The method is described in Powell, Harwell report TP 531, (1973), and Marlow and Powell, Harwell report R.8336, (1977).

Versions: OB14A.
Calls: OB13A.
Language: Fortran IV.
Date: August 1977.
Size: 8.3K; 557 cards.
Origin: S.Marlow and M.J.D.Powell†, A.E.R.E., Harwell.

OE02A

To insert extra statements in a Fortran source deck to **count the number of times statements are executed** during a run. The user can insert controls into the source to switch the counting on and off at selected points and also request a print out of the counts. The Fortran is assumed to be either standard or IBM Fortran IV.

To use OE02A an additional job step is executed in which OE02A will read in the user's source, insert the extra statements and then pass the modified source on to the next job step which will usually be the compile step of a compile-and-go procedure.

Versions: OE02A.
Language: Fortran IV (standard available).
Date: March 1976.
Size: 14.9K; 426 cards.
Origin: J.K.Reid, A.E.R.E., Harwell.

OE04A

Allows a single Fortran deck to contain several versions of a program (or subroutine) by holding alternative statements as special comment cards, e.g. single and double precision versions of a subroutine which would differ only in statements involving precision specifications.

OE04A is in essence a preprocessor which can perform either a reversible, or a non-reversible transformation of one version of a program to another.

Versions: OE04A.
Language: Fortran IV (standard available).
Date: January 1976.
Size: 9.7K; 138 cards.
Origin: J.K.Reid, A.E.R.E., Harwell.

PE09A

Given a polynomial

$$P(x) = \sum_{j=0}^{m} a_j q_j(x)$$

expressed as a sum of **orthogonal polynomials** $q_j(x)$ over a point set x_i $i=1,2,\ldots,n$ this subroutine **computes values of** $P(x)$, $\frac{dP(x)}{dx}$ **and** $\frac{d^2P(x)}{dx^2}$ **at any point** x.

The method is that given by Smith, Math. Comp., 19, (1965).

Remark: similar to PE07A but provides derivative values.

† No longer at A.E.R.E. Harwell.

Versions: PE09A, PE09AD.
Language: Fortran IV (standard).
Date: May 1975.
Size: .9K; 97 cards.
Origin: W.R.Owen, Univ. of Queensland.

QG03A

Given a **piecewise linear function** $\lambda(x)$, defined by points ξ_i and function values

$$\lambda_i = \lambda(\xi_i), \quad i=1,2,\ldots,n, \quad n \geqslant 2$$

and given **integration limits** a **and** b, **and parameters** φ **and** $\bar{x}$ the subroutine **evaluates**

$$Q(\varphi,\bar{x}) = \int_a^b e^{-|\varphi|(x-\bar{x})^2} \lambda(x)\, dx$$

The subroutine can be applied to the problem of **folding a Gaussian into a function.**

Versions: QG03A, QG03AD.
Language: Fortran IV.
Date: May 1975.
Size: 2K; 140 cards.
Origin: S.Marlow, A.E.R.E., Harwell.

QG04A

Given a **cubic spline** $s(x)$ defined by knots ξ_i, function values $s_i = s(\xi_i)$ and derivative values $g_i = \frac{d}{dx}s(\xi_i)$, $i=1,2,\ldots,n$, $n \geqslant 2$ and given **integration limits** a **and** b, **and parameters** φ and $\bar{x}$ the routine **evaluates**

$$Q(\varphi,\bar{x}) = \int_a^b e^{-|\varphi|(x-\bar{x})^2} s(x)\, dx$$

The subroutine is not restricted to cubic splines and may be used with any piecewise cubic function that has a continuous first derivative and which can be specified by its values and first derivative values at the joins.

The subroutine can be applied to the problem of **folding a Gaussian into a function.**

Versions: QG04A, QG04AD.
Calls: QG02A.
Language: Fortran IV.
Date: May 1975.
Size: 2.1K; 137 cards.
Origin: S.Marlow, A.E.R.E., Harwell.

TB06A

This subroutine **computes the spline, of specified degree and specified knots, which interpolates** n **arbitrarily prescribed function values.** Specifically, given function values $f_1,\ldots,f_n$ at the n data points $x_1 < x_2 < \ldots < x_n$, and given $n-k$ knots η_i, $i=1,2,\ldots,n-k$, $(1 \leqslant k \leqslant n)$, in the open interval $x_1 < x < x_n$, this subroutine computes the coefficients $a_1,\ldots,a_n$ of the unique spline function

$$S(x) = \sum_{i=1}^{n} a_i N_{k,i}(x)$$

of degree $k-1$ which has the knots $\{\eta_i\}$ and which satisfies the interpolation conditions

$$\sum_{i=1}^{n} a_i N_{k,i}(x_j) = f_j, \quad j=1,2,\ldots,n.$$

N.B. The function $N_{k,i}(x)$ is a normalized B-spline of degree $k-1$.

As a by-product of the calculation of the coefficients $a_1,\ldots,a_n$ the subroutine also computes and returns the value of the integral

$$\int_{x_1}^{x_n} S(x)\, dx$$

Versions: TB06A, TB06AD.
Language: Fortran IV (standard).
Date: December 1976.
Size: 3.4K; 253 cards.
Origin: P.W.Gaffney, A.E.R.E., Harwell.†

TB07A

Computes the optimal interpolation formula $\bar{S}(x)$, **of specified degree, which interpolates** n **prescribed function values.** Specifically, given function values $f_1,f_2,\ldots,f_n$ at the n data points $x_1 < x_2 < \ldots < x_n$ this subroutine computes the knots $\eta_1,\ldots,\eta_{n-k}$ and the coefficients $a_1,\ldots,a_n$ of the spline function

$$\bar{S}(x) = \sum_{i=1}^{n} a_i N_{k,i}(x)$$

of degree $k-1$, which satisfies the interpolation conditions

$$\sum_{i=1}^{n} a_i N_{k,i}(x_j) = f_j, \quad j=1,2,\ldots,n$$

† No longer at A.E.R.E. Harwell.

and which, whenever $|f^{(k)}|$, $1 \leqslant k \leqslant n$, is bounded, and the value of the bound is unknown, provides the smallest possible value of $B(x)$ in the error bound

$$|f(x) - S(x)| \leqslant B(x) \max_{x_1 \leqslant x \leqslant x_n} |f^{(k)}(x)|.$$

N.B. The function $N_{k,i}(x)$ is a normalised B-spline of degree $k-1$.

As a by-product of the calculation the subroutine also computes the value of the integral

$$\int_{x_1}^{x_n} S(x)\,dx.$$

The method is described in the Harwell reports Gaffney, CSS 52 (1977) and Gaffney, R.8781 (1977).

Versions: TB07A, TB07AD.
Calls: TB06A, TB08A.
Language: Fortran IV (standard).
Date: June 1977.
Size: 1.8K; 103 cards.
Origin: P.W.Gaffney, A.E.R.E., Harwell.†

TB08A

To **compute the knots of the optimal spline interpolation formula.** Specifically, the function $S(x)$ which interploates values of a function of one variable $f(x)$ at n distinct points $x_1 < x_2 < \ldots < x_n$, and which, whenever the kth derivative of $f(x)$ is bounded and the value of the bound is unknown, provides the smallest possible value of $B(x)$ in the error bound

$$|f(x) - S(x)| \leqslant B(x) \max_{x_1 \leqslant x \leqslant x_n} |f^{(k)}(x)|.$$

$S(x)$ is a unique spline function of degree $k-1$ with $n-k$ knots $\eta_1, \ldots, \eta_{n-k}$.

The method is described in the Harwell reports Gaffney, CSS 52 (1977) and Gaffney, R.8781 (1977).

Versions: TB08A, TB08AD.
Language: Fortran IV (standard).
Date: June 1977.
Size: 8K; 550 cards.
Origin: P.W.Gaffney, A.E.R.E., Harwell.†

TG03A

Evaluates a spline $S(x)$ of degree $k-1$ and its derivatives using the B-spline representation of $S(x)$. Specifically, given knots and coefficients $a_1, \ldots, a_{m+k}$ in the representation

$$S(x) = \sum_{i=1}^{m+k} a_i N_{k,i}(x), \quad m \geqslant 0, \quad k \geqslant 1,$$

this subroutine computes the values of

$$\frac{d^{j-1}}{dx^{j-1}} S(x), \quad j = 1,2,\ldots,r, \quad r \leqslant k$$

at a specified point x.

The method is based on De Boor, 'On Calculating with B-splines', J. App. Theory, **6**, (1972).

Versions: TG03A, TG03AD.
Language: Fortran IV (standard).
Date: January 1977.
Size: 2K; 131 cards.
Origin: P.W.Gaffney, A.E.R.E., Harwell.†

TG04A

Given the n points $x_1 \leqslant x_2 \leqslant \ldots \leqslant x_n$ and a value of x, this subroutine **computes the values of the k B-splines of degree $k-1$**

$$M_{k,i}(x) \quad 1 \leqslant i \leqslant n-k,$$

which have knots at the points $x_i, x_{i+1}, \ldots, x_{i+k}$ and which cover the point x, and it also computes the values of the corresponding integrals

$$\int_x^x M_{k,i}(\xi)\,d\xi, \qquad 1 \leqslant i \leqslant n-k.$$

The subroutine uses the recurrence relation due to De Boor, 'On Calculating with B-splines', J. Approx. Theory, **6**, (1972), to evaluate the B-spline. The integrals are evaluated using the method due to Gaffney, Harwell report CSS 10, (1974).

Versions: TG04A, TG04AD.
Language: Fortran IV (standard).
Date: December 1976.
Size: 3.3K; 235 cards.
Origin: P.W.Gaffney, A.E.R.E., Harwell.†

VA13A

To calculate the **minimum of a general function** of several variables when values of the **derivatives** with

---

† No longer at A.E.R.E. Harwell.

respect to the variables can be provided, that is, find $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ to minimize a function $F(\mathbf{x})$ given $\frac{\partial F}{\partial x_j}$ $j=1,2,\ldots,n$.

The subroutine uses the BFGS variable metric method without line searches of the type analysed by Powell, Harwell report CSS 15, (1975).

Versions: VA13A, VA13AD.
Calls: MC11A.
Language: Fortran IV.
Date: May 1975.
Size: 4.1K; 180 cards.
Origin: M.J.D.Powell, A.E.R.E., Harwell.†

VA14A

To calculate the **minimum of a general function of many variables** when values of the **derivatives** with respect to the variables can be provided, that is, find $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ to minimize a function $F(\mathbf{x})$ given $\frac{\partial F}{\partial x_j}$ $j=1,2,\ldots,n$.

The subroutine is intended for problems that are so large that an $n \times n$ matrix cannot be stored conveniently.

A conjugate gradient method is used that includes an automatic restart procedure and it is described in Powell, Harwell report CSS 24, (1975).

Versions: VA14A, VA14AD.
Language: Fortran IV.
Date: November 1975.
Size: 3.8K; 189 cards.
Origin: M.J.D.Powell, A.E.R.E., Harwell.†

VD03A

Given a curve and a data point this subroutine will **compute the point on the curve which is nearest, in a weighted least squares sense, to the data point.** That is, given a function $y(x)$ representing the curve and a point (ξ,η) with associated weights w_x and w_y, the subroutine minimises

$$S(x) = w_\xi(\xi - x)^2 + w_\eta(\eta - y)^2$$

Values of the **first and second derivates** of $y(x)$ must be supplied by the user.

The method is Newton-Raphson with a bound on the x value to ensure convergence.

Versions: VD03A, VD03AD.
Language: Fortran IV (standard).
Date: November 1974.
Size: 2.6K; 342 cards.
Origin: W.R.Owen, Univ. of Queensland.

ZA19AS

Allows a Fortran program **to disable or enable exponent underflow interrupts.** The old status is always returned to the user, thus allowing him to reset to a previous condition if desired.

Fortran error processing of exponent underflows can be expensive. If a program is beyond its developement stage and many valid underflows are expected, which are to be fixed to a zero result, then ZA19AS should be used to save c.p.u. time.

Versions: ZA19AS.
Language: 370/BAL.
Date: May 1975.
Size: .1K; 91 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

ZA21AS

To allow the user of the IBM Fortran DEBUG facility **to switch the INIT option on and off** at chosen points in a program.

Use of ZA21AS can reduce the amount of printed output produced by a debug run and it also makes some reduction in the execution time.

ZA21AS is based on IBM code published in the S.E.A.S. Newsletter, **4**, (June 1975), under the name INIT.

Versions: ZA21AS.
Language: 370/BAL.
Date: November 1975.
Size: .1K; 40 cards.
Origin: M.J.Hopper, A.E.R.E., Harwell.

---

† No longer at A.E.R.E. Harwell.