AD-A092 033 ATOMIC ENERGY RESEARCH ESTABLISHMENT HARWELL (ENGLAND) F/G 12/1
HARWELL SUBROUTINE LIBRARY. A CATALOGUE OF SUBROUTINES (1973).(U)
JUL 73 M J HOPPER
UNCLASSIFIED AERE-R-7477 NL

1 OF 2
AD A
092033

1

HARWELL SUBROUTINE LIBRARY ORDER FORM

EXTERNAL USE OF THE HARWELL SUBROUTINE LIBRARY

We wish the subroutine library to be of use to many people and we are willing to provide copies in source code of individual routines or of the whole library to external users on request. Charges are made (see below) to external users to cover the cost of handling, postage and documentation involved in fulfilling their requests for copies.

The charges cover only the copying and despatching of library routines. Therefore we do not undertake to provide any assistance that may be needed to use a subroutine successfully, and we do not guarantee the efficacy of any subroutine or documentation. However we hope that deficiences in subroutines and the documentation will be brought to our attention, in order that we can improve the Harwell library.

Except for a small number of subroutines obtained from elsewhere, the subroutines in the Harwell Subroutine Library are the property of the United Kingdom Atomic Energy Authority and a potential user must accept and abide by the conditions listed below. All communication with the library by external users should be made through Mr.S.Marlow, Building 8.9, AERE Harwell, Didcot, Oxon, OX11 ORA, (Tel. Abingdon 24141 ext. 2930), who is the liaison officer for the library's external affairs.

The conditions attached to external use are as follows:

(i) the subroutines may only be used for research purposes by the person or organisation to whom they are supplied. They may not be copied for use by other persons or organisations, except with the written permission of the liaison officer.

(ii) due acknowledgement is made of the use of subroutines in any research publications resulting from their use.

(iii) the subroutines may be modified for use in research applications by external users. The nature of such modifications should be indicated in writing for information to the liaison officer. At no time however, shall the subroutines or modifications thereof become the property of the external user.

(iv) the use of the subroutines in commercial applications must be agreed in writing with AERE Harwell and on terms and conditions to be negotiated. In the first instance, anyone considering such commercial applications should write to the liaison officer.

The charges for library material are listed below. Overseas customers are charged on a different scale than that used for U.K. customers to cover the extra costs in meeting their requirements and all items are despatched by air mail. Charges to customers in the United Kingdom are subject to VAT. All prices given are valid to the 1st August 1977, and subject to review thereafter.

(1) Listings of subroutines		
U.K.(excluding VAT)	£5 each	
Overseas	£20 each	
(2) Card decks of subroutines		
U.K.(excluding VAT)	£5 per 400 cards	*(minimum order £10)*
Overseas	£15 per 400 cards	*(minimum order £30)*
(3) Subroutines on magnetic tape (including the cost of the tape which we supply)		
U.K.(excluding VAT)	£20 for the first two	subroutines plus £2 for each subroutine in excess of two.
Overseas	£40 for the first two	subroutines plus £2 for each subroutine in excess of two.
(4) Complete library on tape (including the cost of the tape which we supply and one set of specification sheets)		
U.K.(excluding VAT)	£75	
Overseas	£150	
(5) Additional complete sets of specification sheets		
U.K.(excluding VAT)	£17 each set	
Overseas	£30 each set	

N.B. We require payment with order for orders up to £50. Cheques should be made payable to AERE, Harwell. Please complete the form opposite by filling in the sections appropriate to the library material you require. Enter into the column on the right the cost of the items and fill in the total at the bottom of the form. To avoid any confusion we advise you to cross out the whole of each section that is not relevant to your order.

To: Mr. S. Marlow, Building 8.9, A.E.R.E., Harwell, Didcot, Oxon, OX11 ORA, England.

Please send me a copy of the following material from the Harwell Subroutine Library.

Name, title and mailing address (BLOCK CAPITALS please) ....

....

....

1. Listings Please send listings of the following subroutines ....

....

....

2. Card decks Please send card decks of the following subroutines ....

....

....

Normally these are punched in EBCDIC card code, see table of codes overleaf. However we will use the BCD card code if you mark this box.. ☐

3. Subroutines on magnetic tape Please send me copies of the following subroutines on magnetic tape

....

....

4. Complete library If you wish to receive a copy of the complete library on magnetic tape and a complete set of subroutine specifications, mark this box .... ☐

5. Additional complete sets of subroutine specifications If you wish to receive additional copies of sets of subroutine specification sheets for the complete library enter the number required in this box........ ☐

6. Magnetic tape parameters We prefer to write magnetic tapes in EBCDIC 8-bit code on 9-track tape, see table of codes overleaf, at a density of 800 bits per inch. However if you wish to receive EBCDIC 6-bit code on 7-track tape at a density of 556 bits per inch mark this box .... ☐
The material is blocked in fixed length records, each block containing 40 card images. If you require a different number of card images per block, write the number here ....
N.B. To get the complete library on to one tape the blocking must be at least 4. **If the options offered are not suitable then we will try and meet your requirements.** You will receive full details of the labelling, blocking and character codes of any magnetic tapes that we send you.

7. Library publications If you wish to receive free of charge library bulletins, giving information about new subroutines and modifications to the library, then mark this box.. ☐
If you wish to receive free of charge new issues of the Subroutine Library's catalogue and supplements mark this box .... ☐

8. The computer It would help us if you write down the name and model number of the computer on which you are going to use our library ....

....

VAT (UK only) ....

TOTAL COST ☐

A cheque for £......... made payable to AERE, Harwell is enclosed for my order (orders up to £50) / I agree to pay the charges indicated for my order (orders over £50). {*Please delete whichever does not apply*}.

I agree to abide by the conditions given opposite.

Date.. Signature ....

Name (BLOCK CAPITALS please) ....

for and on behalf of (BLOCK CAPITALS please) ....

THE TABLE OF CHARACTER CODES

The table below shows the character codes that we use when we supply library material on magnetic tape and punched cards. The codes that are listed are for the case when the user has requested EBCDIC and the five characters which are changed when the BCD option is requested are also shown (see second footnote). The table contains only the 64 character set related to seven track tapes and this set will include all the characters in use in the computer languages that are used in the library. The two magnetic tape codes are given in octal (3 bits per digit) and hexadecimal (4 bits per digit) and the parity bit is not included.

7-track octal	9-track hex	original symbol	card code	7-track octal	9-track hex	original symbol	card code
ØØ	4Ø	*space*	*no punches*	4Ø	6Ø	–	11
Ø1	F1	1	1	41	D1	J	11–1
Ø2	F2	2	2	42	D2	K	11–2
Ø3	F3	3	3	43	D3	L	11–3
Ø4	F4	4	4	44	D4	M	11–4
Ø5	F5	5	5	45	D5	N	11–5
Ø6	F6	6	6	46	D6	O	11–6
Ø7	F7	7	7	47	D7	P	11–7
1Ø	F8	8	8	5Ø	D8	Q	11–8
11	F9	9	9	51	D9	R	11–9
12	FØ	Ø	Ø	52	DØ	*no symbol*	11–Ø
13	7B	# {=}	3–8	53	5B	£ †	11–3–8
14	7C	@ {'}	4–8	54	5C	*	11–4–8
15	7D	' ‡	5–8	55	5D	) ‡	11–5–8
16	7E	= ‡	6–8	56	5E	;	11–6–8
17	7F	"	7–8	57	5F	¬	11–7–8
2Ø	7A	:	2–8	6Ø	5Ø	& {+}	12
21	61	/	Ø–1	61	C1	A	12–1
22	E2	S	Ø–2	62	C2	B	12–2
23	E3	T	Ø–3	63	C3	C	12–3
24	E4	U	Ø–4	64	C4	D	12–4
25	E5	V	Ø–5	65	C5	E	12–5
26	E6	W	Ø–6	66	C6	F	12–6
27	E7	X	Ø–7	67	C7	G	12–7
3Ø	E8	Y	Ø–8	7Ø	C8	H	12–8
31	E9	Z	Ø–9	71	C9	I	12–9
32	EØ	*no symbol*	Ø–2–8	72	CØ	*no symbol*	12–Ø
33	6B	,	Ø–3–8	73	4B	.	12–3–8
34	6C	% {(}	Ø–4–8	74	4C	< {)}	12–4–8
35	6D	_	Ø–5–8	75	4D	(‡	12–5–3
36	6E	>	Ø–6–8	76	4E	+ ‡	12–6–8
37	6F	?	Ø–7–8	77	4F	|	12–7–8

† The Harwell printers print the currency symbol as a £ symbol.

‡ when the BCD option is specified these five characters are translated to the positions indicated by the characters in braces. In all five cases the translation is two characters into one, e.g. both # and = get translated to hexadecimal 7B. Note that the BCD option is not a one to one translation and in particular the distinction between the + character and the & character is lost. The & character is a valid charcter in IBM Fortran and in the rare cases were it is used in library routines the BCD translation may change the meaning of the Fortran code.

(14) AERE—R.7477

(6)

HARWELL SUBROUTINE LIBRARY·

A CATALOGUE OF SUBROUTINES (1973),

Compiled by M.J. Hopper.

(10) (11) Jul 73 (12) 140

This report contains a complete list of all the subroutines currently in the Harwell Subroutine Library and gives for each one a brief outline of purpose, method, origin, language and other attributes. Also included are contributions by members of the Numerical Analysis Group on topics of general interest to library users. One of these is on how to use the library and the others cover data fitting, optimization, linear algebra and quadrature.

This edition of the catalogue supersedes the previous issue (1971) R.6912.

Acknowledgements

I wish to thank Messrs. M.J.D. Powell, R. Fletcher, J.K. Reid and A.R. Curtis for contributing the sections on Data Fitting, Optimization, Linear Algebra and quadrature.

Theoretical Physics Division,
U.K.A.E.A. Research Group,
Atomic Energy Research Establishment,
HARWELL.

July, 1973

HL.73/2891 (C13)
HMD.

p 1

CONTENTS AND SUBROUTINE CLASSIFICATION

SBN 7058 0183 7

Introduction

1. The Harwell Subroutine Library

The Harwell Subroutine Library is maintained by the Numerical Analysis Group of Theoretical Physics Division, A.E.R.E. Harwell. It is strictly a library of subprograms which must be called by a user written program. Complete programs are not included in the library although some of the library 'routines' are in fact packages of more than one subprogram. The principle language is FORTRAN but a few of the routines are written in the machine code for the current Harwell computer - an IBM model 370/165.

The library is composed mainly of mathematical and numerical analysis routines. These have in the main been written by members of the Numerical Analysis Group, past and present, and are usually of a high standard. A few of the routines have been derived from sources outside Harwell and acknowledgements are made to that effect in the catalogue entries.

The principle function of the library is to provide Harwell computer users with good numerical facilities. We do however get many requests for the library from outside Harwell and we provide copies of the library, and listings and card decks of individual routines for a small handling charge, see section A subsection 6 in part II.

The library started life in 1963 and was first used on an IBM 7030 (STRETCH) computer. In 1967 the whole library was converted to be used on an IBM 360 and at that time the machine coded routines were rewritten and the concept of having both single and double precision versions of routines was introduced.

2. The catalogue

This catalogue of the Harwell subroutine library stands as a precise definition of the library. It serves as a reference document for library facilities for users of the Harwell computer. New users of the library will find the section A in part II 'how to use the library' useful as an introduction to library facilities. Also useful, the general index at the back of this report which has been extended in scope to provide a general cross reference index to all library facilities.

This is a new issue of the catalogue and it supersedes the 1971 issue R.6912 and its two supplements. All the new library routines introduced since 1971 have been included and a few that have been discontinued have been removed. A new external users section giving details of the charges for library material has been included, this was previously published in the first supplement to R.6912. There is a new subsection of section A on library information sources and a complete list of all A.E.R.E. reports which cover library routines and include listings. The three sections on data fitting, optimization and linear algebra have been brought up to date and a new section on quadrature has been added. In the general list of routines we now give for each routine a list of any other library or user routines which are called by that routine. There is no section on the Harwell graphical package this time as this is now the responsibility of the Central Computer Group at Harwell and is covered in, R. Jones and W. Prior, 'GHOST users manual', TP 484.

The catalogue consists of two parts. Part I contains the complete list of routines that make up the library giving details of purpose, method and attributes. The details given are brief and only intended to be sufficient for a potential user to decide which of the routines, if any, are suitable for his particular problem. The routines are listed in alphabetical order and the classification used for routine names is such that the list falls naturally into sections associated with different classes of problem. There are some exceptions and general topics, such as data fitting, are covered by routines spread throughout the list. The general index provided at the end of the report can be used to help the user locate routines associated with his particular problem. At the end of part I you will find the list of A.E.R.E. reports which cover library routines.

Part II consists of sections contributed by members of the Numerical Analysis Group on topics of general interest. They are intended to cover the situation which often arises when there appears to be more than one routine in the library which could be used to solve a particular problem. The sections give guidance as to best methods and point to the routines which should be used. The sections also serve an educational purpose in showing ways of setting up problems so that best use is made of the facilities available. It is hoped that these contributions will grow in number and content with each new re-issue of the catalogue.

The first section in part II is on how to use the library. Because this report is primarily a reference document it has been placed after the list of subroutines and not at the beginning where it logically should be. Anyone unfamiliar with the library should make a point of reading that section, particularly the sections on naming conventions and write ups.

Some of the conventions used in the list of subroutines in part I will need some explanation and this is covered by the following subsections.

2.1 Language

The FORTRAN used up to Aug 1967 on the IBM 7030 was a FORTRAN closely related to FORTRAN II or basic FORTRAN. The effect on the library is that pre-1967 routines will most likely contain only FORTRAN II like features. After Aug 1967 FORTRAN IV features started to come in with some of the IBM extensions to FORTRAN IV. The routines have not been classified according to FORTRAN dialect or according to deviations from standard because in general this information is not known. We make no claims in respect to the portability of the library. We consider that a simpler user interface combined with fairly efficient code is important and this aim often conflicts with the rather crude FORTRAN standards currently in operation.

The IBM assembler language for the 360 or 370 machines has been denoted by 360/BAL where BAL stands for basic assembler language. It is a term we use loosely to cover assembly language programming including floating point and macro facilities. No routine in the library uses special 370 features but some of the system routines may be OS release dependent.

2.2 Versions

This lists the names of other versions of the subroutine and is principally used to show that a double precision version is available and give its name. Double precision names are distinguished by adding a D to the single precision name, see subsection 1 of section A in part II.

2.3 Date

This gives the approximate date that the routine was introduced into the library. If a routine has been modified since its introduction and the modification was extensive enough to consider it a new routine the date given is the date when the modification was made.

2.4 Size

The size of each routine is given in two parts; the first is the core required to load the routine given to the nearest 100 bytes (1 byte = 8 bits), the second part gives the number of cards in the source deck. Both sizes are approximate and are associated with that routine whose name appears at the head of the catalogue entry. This is usually the single precision version; double precision versions are likely to take up more space than the single precision version. Users should be careful using these sizes to estimate total core requirements and should consult the write ups for the routines to find out how much extra work space must be provided. Card deck sizes can also be misleading as some routines have a generous number of comment cards in them and so appear larger routines than they really are.

2.5 Calls

We give a list of other library routines or user provided routines which the routine calls. We give only the list for the routine named at the head of the catalogue entry, the lists for other versions can be deduced from this.

2.6 Origin

Here the author's name and the place of origin of the routine is given. Some of the routines come from external sources and we give the author's name in recognition for providing us with library material. However most of the external routines have undergone some modification for Harwell use and therefore the responsibility for maintaining their good working must rest with Harwell and not the original authors. Many of the authors listed as Harwell are no longer at this establishment and this has been indicated by an * after their name.

Queries concerning library routines should not be directed to the authors personally but through the library's queries service, see subsection 4 in section A in part II.

3. Keeping the catalogue up to date

From the experience gained with the first catalogue, R.6912, we have changed our ideas on this subject. The period between complete new issues of the catalogue will have to be more than 12 months and it is likely to be at least 2 years. In this period we shall keep the catalogue up to date by bringing out supplements which will list the new routines introduced into the library since the last issue of a catalogue or supplement. External users on our mailing list will get these supplements automatically otherwise they will be available only on demand. We recommend that any time you request a catalogue you should ask also for any supplements to the catalogue.

Comments, critical or otherwise, concerning the catalogue are always welcome, particularly on the method of keeping it up to date.

Harwell Subroutine Librarian
M.J. Hopper
July 1973

PART I: A List of Subroutines

D. Differential Equations

DA01A

To integrate a set of first order ordinary differential equations

$$y_i' = f_i(y_1,y_2,\ldots,y_n,x) \qquad i = 1,2,\ldots,n$$

given initial conditions $y_i(x^o) = y_i^o$. Each call to the subroutine advances the integration one step, the step length being set by the user. A subroutine must be provided to compute values of the functions f_i.

The subroutine uses the 4th order Runge-Kutta method proposed by Merson which attempts to estimate the truncation error at each step.

Remark: This subroutine has been superseded by DC01AD.
Versions: DA01A; DA01AD.
Calls: DYBDX (user routine).
Language: FORTRAN, Date: March 1963, Size: 1.2K; 44 cards.
Origin: D. McVicar, Harwell.

DA02A

To integrate a set of first order ordinary differential equations

$$y_i' = f_i(y_1,y_2,\ldots,y_n,x) \qquad i = 1,2,\ldots,n$$

Given initial conditions $y_i(x^o) = y_i^o$. The steplength is controlled automatically by the routine so that at each step the truncation error should satisfy an accuracy requirement specified by the user. The Runge-Kutta routine, DA01A, is called and the Merson truncation error estimate is used to determine the steplengths. The accuracy is not guaranteed.

The user must provide a subroutine to calculate the functions f_i and optionally a subroutine to print results at specified print points.

Remark: DC01AD and DC02AD provide more powerful facilities.
Versions: DA02A; DA02AD.
Calls: DA01A and MX02A.
Language: FORTRAN, Date: March 1972, Size: 2.1K; 154 cards.
Origin: A.R. Curtis and A.B. Smith*, Harwell.

DC01AD

A package of routines to integrate a system of first order ordinary differential equations

$$y_i' = f_i(y_1,y_2,\ldots,y_n,x) \qquad i = 1,2,\ldots,n$$

given initial values $y_i(x^o) = y_i^o$ using a predictor-corrector method due to C.W. Gear. Given a requested accuracy this method automatically chooses the step size and the order of the integration formula. It possesses excellent stability propoerties which enable it to take long steps even in stiff systems (i.e. those involving very short time constants).

The user must supply a subroutine to compute the functions f_i, and optionally an output subroutine to be called at specified print points. Integration details are made available to the output subroutine through named COMMON areas. There is an interpolation subroutine which may be called from the output routine to get values of $y_i(x)$ for any value of x in the current step.

Remark: DC02AD provides a more simplified calling sequence to DC01AD.
Versions: DC01AD; there is no single precision version.
Calls: PB01AD, MC01AD and MB01BD.
Language: FORTRAN, Date: May 1970, Size: 14.9K; 984 cards.
Origin: A.R. Curtis, Harwell.

DC02AD

To integrate a system of first order ordinary differential equations

$$y_i' = f_i(y_1,y_2,\ldots,y_n,x) \qquad i = 1,2,\ldots,n \qquad n \leq 10$$

given initial values $y_i(x^o) = y_i^o$. This subroutine calls DC01AD forfeiting some of its facilities to provide a more simplified calling sequence.

The user provides a routine to compute the functions f_i, but printed output is produced by the package's standard output routine. The restriction on n can be removed by simple change to the array storage allocation in a COMMON block.

Versions: DC02AD; there is no single precision version.
Calls: DC01AD.
Language: FORTRAN, Date: May 1970, Size: 3.9K; 32 cards.
Origin: A.R. Curtis, Harwell.

DD01A

To solve the two point boundary value problem for the second order linear differential equation

$$y'' + f(x)y' + g(x)y = k(x) \qquad x_1 \leq x \leq x_n$$

given boundary conditions of the form $ay' + by = c$ at the two points x_1 and x_n.

A finite difference approximation is used, see L. Fox, Proc. Roy. Soc. A.190, 1947. Initially 3rd and higher differences are ignored; then successive approximations are obtained by applying correction terms based upon 3rd and 4th differences.

The values of f(x), g(x) and k(x) at the points $x_1,x_2,\ldots,x_n$ are passed to the subroutine in three arrays.

Versions: DD01A; DD01AD.
Calls: MA07A and TA03A.
Language: FORTRAN, Date: May 1965, Size: 2.6K; 172 cards.
Origin: P. Hallowell, Atlas Lab., Chilton, Berks.

DD02A

To solve the two point boundary value problem for the second order non-linear differential equation

$$y'' + f(x,y,y')y' + g(x,y,y')y = k(x,y,y') \qquad x_1 \leq x \leq x_n$$

given boundary conditions of the form $ay' + by = c$ at the two points x_1 and x_n.

Starting from an initial approximation to the solution the equation is linearized, solved by calling DD01A and then re-linearized until the required accuracy is reached. At this stage the effect of 3rd and 4th differences has been ignored and the solution is then corrected to take these into account.

The user must provide a subroutine to compute values of $f(x,y,y')$, $g(x,y,y')$ and $k(x,y,y')$.

Versions: DD02A, DD02AD.
Calls: DD01A, TA03A, TD01A and FUNCTS (a user routine).
Language: FORTRAN, Date: May 1965, Size: 2.3K; 171 cards.
Origin: P. Hallowell, Atlas Lab., Chilton, Berks.

DINT

Returns the double precision floating point value of the integer part of a real double precision floating point number. It supplements the IBM FORTRAN support functions INT, IDINT and AINT.

Versions: DINT double precision.
Language: 360/BAL, Date: Sept. 1967, Size: .1K; 15 cards.
Origin: R.C.F. McLatchie, Harwell.

DP01A

To solve the two point boundary value problem for the linear parabolic partial differential equation

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial u}{\partial x} + cu + d$$

where a,b,c and d are functions of x and t, $x_1 \leq x \leq x_n$, given boundary conditions of the form $P\frac{\partial u}{\partial t} = q + ru + s\frac{\partial u}{\partial x}$ at the points x_1 and x_n. Given δt and the solution $u(x,t)$ at t, the subroutine advances the integration one time step to obtain $u(x,t+\delta t)$.

The Crank-Nicolson integration formula is used to transform to a 2nd order linear ordinary differential equation which is solved by calling DD01A.

The user must supply a subroutine to compute values of the functions a,b,c and d given current values of x and t.

Versions: DP01A; DP01AD.
Calls: DD01A, TA03A, TD01A and FUNCTS (a user routine).
Language: FORTRAN, Date: Sept. 1965, Size: 3.5K; 197 cards.
Origin: P. Hallowell, Atlas Lab., Chilton, Berks.

DPO2A

To solve the two point boundary value problem for the non-linear parabolic partial differential equation

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial u}{\partial x} + cu + d \qquad x_1 \leq x \leq x_n$$

where a,b,c and d are functions of x,t,u and $\frac{\partial u}{\partial x}$, given boundary conditions of the form $P\frac{\partial u}{\partial t} = q + ru + s\frac{\partial u}{\partial x}$ at the points x_1 and x_n. Given δt and the solution u(x,t) at t, the subroutine advances the integration one time step to obtain $u(x,t+\delta t)$.

Using the solution at t as a first approximation the equation is linearized and successively re-linearized until the required accuracy is obtained. DPO1A is used to solve the linearized equation which in turn uses DDO1A.

The user must provide a subroutine to compute the functions a,b,c and d given current values of x,t,u and $\frac{\partial u}{\partial x}$.

Versions: DPO2A; DPO2AD.

Calls: DPO1A, TAO3A and TDO1A.

Language: FORTRAN, Date: June 1966, Size: 2.4K; 203 cards.

Origin: P. Hallowell, Atlas Lab., Chilton, Berks.

E. Eigenvalues and Eigenvectors of Matrices

EAO2A

Given a real symmetric matrix A and an estimate λ_e of one of its eigenvalues, will find to a given accuracy the eigenvalue nearest to λ_e and its corresponding eigenvector. The eigenvector is normalized to have unit length.

The power method is applied to the matrix $(A-\lambda_e I)^{-1}$ with Aitken extrapolation every 3rd iteration, see 'Modern Computing Methods'. NPL.

Remark: for matrices of small order the QR routines, EAO6C, etc. which find all eigenvalues will be more efficient.

Versions: EAO2A; EAO2AD.

Calls: MBO1B.

Language: FORTRAN, Date: July 1963, Size: 12.4K; 70 cards.

Origin: E.J. York*, Harwell.

EAO3A

Given a real symmetric matrix A, finds all its eigenvalues λ_i and eigenvectors $\underline{x}_i$, i.e. finds the non-trivial solutions of $A\underline{x} = \lambda\underline{x}$.

Jacobi's method is used, where the successive off-diagonal elements of largest modulus are reduced to zero by orthogonal transformations until all off-diagonal elements are less than some prescribed value. The transformations are simultaneously applied to the unit matrix to generate the vectors.

Remark: This routine is only to be preferred over the QR method routines when the matrix tends to be diagonally dominant.

Versions: EA03A; EA03AD.

Language: FORTRAN, Date: May 1963, Size: 3.3K; 148 cards.

Origin: J. Soper, Harwell.

EA06C

Given a real symmetric matrix A, finds all its eigenvalues λ_i and eigenvectors $\underline{x}_i$, i.e. finds the non-trivial solutions of $A\underline{x} = \lambda\underline{x}$. The eigenvectors are normalized to have unit length.

The matrix is reduced to tri-diagonal form by applying Householder transformations. The eigenvalue problem for the reduced problem is then solved by calling EA08C which uses the QR algorithm.

Versions: EA06C; EA06CD.

Calls: EA08C and MC04B.

Language: FORTRAN, Date: Feb. 1970, Size: 1.3K; 26 cards.

Origin: J.K. Reid, Harwell.

EA07C

Given a real symmetric matrix A, finds all its eigenvalues, i.e. finds the solutions λ_i of $\det(A-\lambda I) = 0$.

The matrix is reduced to tri-diagonal form by applying Householder transformations; then the eigenvalues of the reduced matrix are found by calling EA09C which uses the QR algorithm.

Versions: EA07C; EA07CD.

Calls: EA09C and MC04B.

Language: FORTRAN, Date: Feb. 1970, Size: .6K; 12 cards.

Origin: J.K. Reid, Harwell.

EA08C

Finds all the eigenvalues and eigenvectors of a real tri-diagonal symmetric matrix; the eigenvectors will have unit length.

First EA09C, which uses the QR algorithm, is used to find the eigenvalues, using these as shifts the QR algorithm is again applied but now using the plane rotations to generate the eigenvectors. Finally the eigenvalues are refined by taking Rayleigh quotients of the vectors.

Versions: EA08C; EA08CD.

Calls: EA09C.

Language: FORTRAN, Date: Feb. 1970, Size: 1.8K; 76 cards.

Origin: J.K. Reid, Harwell.

EA09C

Finds all the eigenvalues of a real tri-diagonal symmetric matrix. The QR algorithm with shifts is used.

Versions: EA09C; EA09CD.

Language: FORTRAN, Date: Feb. 1970, Size: 1.3K; 52 cards.

Origin: J.K. Reid, Harwell.

EA11A

Calculates all the eigenvalues λ_i and eigenvectors $\underline{x}_i$ of the system $A\underline{x}_i = \lambda_i B\underline{x}_i$ where A is a real symmetric matrix and B is a real symmetric positive definite matrix.

The matrix B is factorized into LL^T and the eigenvalue problem $L^{-1}A(L^T)^{-1}L^T\underline{x} = L^T\underline{x}$ is solved using the EA06 - EA09 routines.

Versions: EA11A; EA11AD.

Calls: EA06C, MA22A, MX02A and MC03AS.

Language: FORTRAN, Date: Feb. 1972, Size: 2.2K; 80 cards.

Origin: S. Marlow, Harwell.

EB01A

Finds all the eigenvalues and eigenvectors of a general real matrix. The user has options for requesting, no eigenvectors, only right eigenvectors, only left eigenvectors, or both right and left. The vectors will have unit length.

The matrix is reduced by Householder transformations to upper Hessenberg form. The eigenvalue problem for the reduced matrix is then solved by calling EB04A. The eigenvectors, if required it obtains by calling EB05A.

Remark: Superseded by EB06A.

Versions: EB01A; EB01AD.

Calls: EB04A, EB05A and MC08A.

Language: FORTRAN, Date: May 1967, Size: 2.6K; 132 cards.

Origin: E.J. York* and M. Reynolds*, Harwell.

EB04A

Finds all the eigenvalues, real and complex, of a real upper Hessenberg matrix, i.e. a matrix $A = \{a_{ij}\}$ such that $a_{ij} = 0$ for all $i < j+1$.

The double QR transformation is used, see J.G.F. Francis, Computer Journal, Vol. 4, 1961 (also 1962).

The user may limit the number of times the transformation is to be applied.

Remark: Superseded by EB09A.

Versions: EB04A; EB04AD.

Calls: MC03AS.

Language: FORTRAN, Date: Sept. 1965, Size: 4.4K; 261 cards.

Origin: P.J. Hallowell, Atlas Lab., Chilton, Berks.

EB05A

Given a real upper Hessenberg matrix A and one of its eigenvalues λ, finds the corresponding eigenvector $\underline{x}$. This may optionally be a right or left vector and will have unit length.

Inverse iteration is used, i.e. an iteration based on $(A-\lambda I)\underline{x}_{n+1} = \underline{x}_n$.

Remark: Superseded by EB08A.
Versions: EB05A; EB05AD.
Calls: MA12A and ME04B.
Language: FORTRAN, Date: Nov. 1966, Size: 4.2K; 284 cards.
Origin: M. Reynolds*, Harwell.

EB06A

Finds all the eigenvalues and right eigenvectors of a real general matrix, i.e. finds solutions λ and $\underline{x}$ of $A\underline{x} = \lambda\underline{x}$ where the matrix A is real and has no special structure. An option for balancing A is provided. The vectors are normalized to have unit length.

A QR method is used. The matrix is reduced to Hessenberg form by orthogonal similarity transformations (MC14A) and the QR method applied to obtain the eigenvalues (EB08A), see J.H. Wilkinson and C. Reinsch, 'Handbook for Automatic Computation, Linear Algebra', Springer-Verlag. Balancing is performed by MC15A.

Versions: EB06A; EB06AD.
Calls: EB08A, MC14A, MC15A and MC03AS.
Language: FORTRAN, Date: June 1973, Size: 2.7K; 162 cards.
Origin: S. Marlow, Harwell.

EB07A

Finds all the eigenvalues of a real general matrix, i.e. finds solutions λ of $A\underline{x} = \lambda\underline{x}$ where A is a real matrix with no special structure. An option for balancing A is provided.

A QR method is used, see EB06A.

Versions: EB07A; EB07AD.
Calls: EB09A, MC14A and MC15A.
Language: FORTRAN, Date: June 1973, Size: 1K; 73 cards.
Origin: S. Marlow, Harwell.

EB08A

Finds all the eigenvalues and right eigenvectors of a real upper Hessenberg matrix, i.e. finds solutions λ and $\underline{x}$ of $H\underline{x} = \lambda\underline{x}$ where H is a real matrix which has all zero elements $H_{ij} = 0$ for $i > j+1$. The eigenvectors are normalized to have unit length and there is an option which allows isolated eigenvalues to be specified.

A QR method is used, see J.H. Wilkinson and C. Reinsch, 'Handbook for Automatic Computation, Linear Algebra', Springer-Verlag.

Versions: EB08A; EB08AD.
Calls: MC03AS.
Language: FORTRAN, Date: June 1973, Size: 9.4K; 429 cards.
Origin: HQR2 from reference given above, modified for Harwell by S. Marlow.

EB09A

Finds all the eigenvalues of a real upper Hessenberg matrix, i.e. finds solutions λ of $H\underline{x} = \lambda\underline{x}$ where H is a real matrix which has all zero elements $H_{ij} = 0$ for $i > j+1$.

A QR method is used, see EB08A.

Versions: EB09A; EB09AD.
Language: FORTRAN, Date: June 1973, Size: 3.6K; 196 cards.
Origin: HQR from reference given for EB08A, modified for Harwell by S. Marlow.

EC02A

Finds all the eigenvalues of a tri-diagonal Hermitian matrix.

A bisection method based upon the Sturm's sequence is used. The eigenvalues are returned in descending order of magnit de.

Versions: EC02A; EC02AD.
Language: FORTRAN, Date: March 1965, Size: 1K; 42 cards.
Origin: E.J. York*, Harwell.

EC06C

Given a complex Hermitian matrix $A = \{a_{ij}\}$, $a_{ij} = \bar{a}_{ji}$, finds all its eigenvalues λ_i and eigenvectors $\underline{x}_i$, i.e. finds the nontrivial solutions of $A\underline{x} = \lambda\underline{x}$. The eigenvectors are normalized to have unit length.

The matrix is reduced to tri-diagonal form by Householder orthogonal transformations and the reduced eigenvalue/eigenvector problem solved by a QR method (EC08C).

Versions: EC06C; EC06CD.
Calls: EC08C, ME08A and ME06AS.
Language: FORTRAN, Date: Sept. 1971, Size: 1.4K; 31 cards.
Origin: S. Marlow, Harwell.

EC07C

Given a complex Hermitian matrix $A = \{a_{ij}\}$, $a_{ij} = \bar{a}_{ji}$, finds all its eigenvalues λ_i such that $\det(A-\lambda_i I) = 0$.

The matrix is reduced to triangular form by applying Householder orthogonal transformations and the eigenvalues of the reduced matrix are found by EA09C.

Versions: EC07C; EC07CD.
Calls: EC09C and ME08A.
Language: FORTRAN, Date: September 1971, Size: .5K; 13 cards.
Origin: S. Marlow, Harwell.

EC08C

Finds all the eigenvalues and eigenvectors of a complex Hermitian tri-diagonal matrix, the vectors are normalized to have unit length.

The matrix is transformed into a real form by a unitary diagonal transformation. EA08C is then used and the vectors recovered by re-applying the transformation.

Versions: EC08C; EC08CD.

Calls: EA08C.

Language: FORTRAN, Date: September 1971, Size: 1.2K; 47 cards.

Origin: S. Marlow, Harwell.

EC09C

Finds all the eigenvalues of a complex Hermitian tri-diagonal matrix. The matrix is transformed into a real form by a unitary diagonal transformation. The routine for the real case EA09C is then used.

Versions: EC09C; EC09CD.

Calls: EA09C.

Language: FORTRAN, Date: September 1971; Size: .5K; 12 cards.

Origin: S. Marlow, Harwell.

F. Mathematical Functions (see also section S for some functions of statistics).

FA01AS

Generates uniformly distributed pseudo-random numbers. Random numbers are generated in the ranges $0 \leq \xi \leq 1$, $-1 \leq \eta \leq 1$ and random integers in $1 \leq k \leq N$ where N is specified by the user.

A multiplicative congruent method is used where a 32 bit generator word g is maintained. On each call to the routine g_{n+1} is updated to $3^{15}g_n \bmod(2^{32})$; the initial 32 bit value of g is '01010101.......01'. Depending on the type of random number required the following are computed, $\xi = 2^{-32}g_{n+1}$; $\eta = 2^{-31}g_{n+1}-1$ or $k = \text{int.part}\,[\xi N] + 1$.

The routine also provides a facility for saving the current value of the generator word and for re-starting with any specified value.

Versions: FA01AS; there is no double precision version.

Language: 360/BAL, Date: Aug. 1967, Size: .3K; 108 cards.

Origin: R.C.F. McLatchie, Harwell.

FA02AS

To return the signed fractional part of a real floating point number.

Versions: FA02AS; FA02AD.

Language: 360/BAL, Date: 1967, Size: .1K; 45 cards.

Origin: Harwell.

FA03A

Generates pseudo-random numbers from a Gaussian distribution with mean zero and standard deviation specified by the user.

The theory that sample means have a Gaussian distribution is used and the numbers are generated by taking means of samples of size 12. There is an entry which allows the user to specify his own sample size.

Versions:	FA03A; there is no double length version.
Calls:	FA01AS.
Language:	FORTRAN, Date: November 1971, Size: .7K; 33 cards.
Origin:	A.B. Smith*, Harwell.

FB01A

Computes values of the complete elliptic integrals of the 1st and 2nd kind, viz.

$$K(m) = \int_0^{\frac{\pi}{2}} (1-m^2 \sin^2 \theta)^{-\frac{1}{2}} d\theta \qquad 0 \leq m^2 \leq 1$$

$$E(m) = \int_0^{\frac{\pi}{2}} (1-m^2 \sin^2 \theta)^{\frac{1}{2}} d\theta \qquad 0 \leq m^2 \leq 1$$

The subroutine uses an approximation of the form

$$\sum_{k=0}^{n} (1-m^2)^k \left\{a_k + b_k \log_e \left(\frac{1}{1-m^2}\right)\right\},$$

see for n = 4 C.R. Hastings, 'Approximations for digital computers'.

Accuracies:	FB01A < 10^{-6} (n=4); FB01AD < 10^{-14} (n=10).
Versions:	FB01A; FB01AD.
Language:	FORTRAN, Date: 1967, Size: .7K; 20 cards.
Origin:	S. Marlow, Harwell.

FB02A

To compute values of the incomplete elliptic integrals of the 1st and 2nd kinds, viz.

$$F(\psi,m) = \int_0^{\psi} (1-m^2\sin^2\theta)^{-\frac{1}{2}} d\theta$$

$$E(\psi,m) = \int_0^{\psi} (1-m^2\sin^2\theta)^{\frac{1}{2}} d\theta$$

where $0 \leq m^2 \leq 1$ and $0 \leq \psi \leq \frac{\pi}{2}$.

Accuracies:	FB02A < 10^{-6}. FB02AD < 10^{-14}; FB02BD < 10^{-9}; FB02CD < 10^{-5}.
Versions:	FB02A; FB02AD; FB02BD; FB02CD.
Language:	FORTRAN, Date: Revised 1967, Size: 1.8K; 78 cards.
Origin:	L. Morgan*, Harwell.

FB03A

To compute values of the complete elliptic integral of the 3rd kind, viz.

$$\Pi(n,m) = \int_0^{\frac{\pi}{2}} (1-n\sin^2\theta)^{-1}(1-m^2\sin^2\theta)^{-\frac{1}{2}}\, d\theta \qquad 0 \leqslant m^2 \leqslant 1$$

The integral is represented in terms of complete and incomplete elliptic integrals of the 1st and 2nd kind and the routines FB01A and FB02A are used to obtain the required values.

Versions: FB03A; FB03AD.
Calls: FB01A and FB02A.
Language: FORTRAN, Date: Revised 1967, Size: 1.2K; 35 cards.
Origin: M. Ruffle*, Harwell.

FC01A

To evaluate the function $e^{-z^2}\text{erfc}(-iz)$ for complex z, viz.

$$W(z) = e^{-z^2}\left(1 + \frac{2i}{\pi^{\frac{1}{2}}}\int_0^z e^{t^2} dt\right)$$

W(z) is evaluated for $z = x+iy$ $x,y > 0$ using the approximations

(a) if $|z| \leqslant 1$ integrate first few terms of expansion of e^{t^2}.

(b) if $1 < |z| \leqslant 4$ and $y < 1.4$ integrate Taylor series expansion of e^{t^2} about points in the region at which the value of W(z) is known.

(c) if $1 < |z|$ and $y > 1.4$ numerically integrate

$$W(z) = \frac{i}{\pi}\int_{-\infty}^{\infty} \frac{e^{-t^2}}{z-t}\, dt$$

using the midpoint rule or Gauss-Hermite.

For outside the region $x,y > 0$ the relations $W(-z) = 2e^{-z^2} - W(z)$ and $W(\bar{z}) = \bar{W}(z)$ are used.

Versions: FC01A; FC01AD.
Language: FORTRAN, Date: Feb. 1964, Size: 6.1K; 298 cards.
Origin: A. Bailey*, Harwell.

FC03A

To compute values of the Gamma function

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t}\, dt$$

In the range $2 \leqslant x \leqslant 3$ an approximation of the form $\sum_{n=0}^{15} a_n(x-2)^n$ is used; for

$x > 10$ Stirling's approximation is used including up to 10 terms of the asymptotic expansion.

For other values except $x = 0$ or a negative integer the relationship $\Gamma(x+1) = x\Gamma(x)$ is used to relate the required value with the range $2 \leqslant x \leqslant 3$.

Accuracies: FC03A $< 10^{-6}\ |\Gamma(x)|$; FC03AD $< 10^{-14}\ |\Gamma(x)|$.

Remark: the IBM functions GAMMA and DGAMMA are faster routines but the former is inferior in accuracy to FC03A for large x and neither allow $x < 0$.

Versions: FC03A; FC03AD.

Language: FORTRAN, Date: March 1963, Size: 1K; 37 cards.

Origin: S. Marlow, Harwell.

FC05A

To compute values of the Beta function

$$B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\,dt$$

The relation $B(x,y) = \Gamma(x)\Gamma(y)/\Gamma(x+y)$ is used. Approximations similar to those used by FC03A are used but taking advantage of the combined form that is being evaluated.

Accuracies: FC05A $< 10^{-6}$; FC05AD $< 10^{-14}$.

Versions: FC05A; FC05AD.

Language: FORTRAN, Date: May 1963, Size: 1.4K; 64 cards.

Origin: S. Marlow, Harwell.

FC10A

Computes the real and imaginary part of the Fresnel integral

$$f(x) = C(x) + iS(x) = \frac{1}{(2\pi)^{\frac{1}{2}}}\int_0^x t^{-\frac{1}{2}}e^{-it}\,dt$$

The approximations used are of the form

(a) $0 \leqslant x \leqslant 4$ $\quad f(x) = e^{-ix}\sum_{n=0}^{11}(a_n + ib_n)\left(\frac{x}{4}\right)^{n+\frac{1}{2}}$

(b) $x > 4$ $\quad f(x) = \frac{1-i}{2} + e^{-ix}\sum_{n=0}^{11}(c_n + id_n)\left(\frac{4}{x}\right)^{n+\frac{1}{2}}$

See J. Boersman, Maths. of Computation, Vol. 14, No. 72, 1960.

Accuracies: FC10A $< 10^{-6}$, FC10AD $< 3 \times 10^{-7}$.

Versions: FC10A; FC10AD.

Language: FORTRAN, Date: July 1963 Size: 1.6K; 36 cards.

Origin: S. Marlow, Harwell.

FC11A

To compute values of the exponential integral

$$E(x) = \int_x^\infty \frac{e^{-t}}{t}\,dt \qquad x > 0$$

The subroutine uses approximations of the form

(a) for $0 \leq x \leq 4$ $\quad \sum_{n=0}^{21} a_n \left(\frac{x}{4}\right)^n + \log x$

(b) for $x > 4$ $\quad \sum_{n=0}^{20} b_n \left(\frac{4}{x}\right)^n \exp(-x)$

Accuracies: FC11A $< 10^{-6}$; FC11AD $< 10^{-12}$.
Versions: FC11A; FC11AD.
Language: FORTRAN, Date: July 1963, Size: 1.1K; 29 cards.
Origin: S. Marlow, Harwell.

FC12A

Computes the real and imaginary parts of the Plasma Dispersion Function

$$Z(z) = \frac{1}{\pi^{\frac{1}{2}}} \int_{-\infty}^{\infty} \frac{e^{-t^2}}{t-z}\,dt \qquad \text{where } z = x+iy,$$

for the case $y > 0$, and the analytic continuation of this for $y < 0$ as defined by Fried and Conte, 'The Plasma Dispersion Function', Academic Press, 1961. The derivative $Z'(z) = -2(1+zZ(Z))$ is also computed.

If $y \geq 2.75$ or if $y \geq 2$ and $x \geq 4$ an asymptotic continued fraction due to Fried and Conte is used, otherwise if $x \geq 6.25$ a rational approximation from Abramowitz and Stegun is used, otherwise a Taylor series is used.

Accuracy: approx. 10^{-6} absolute.
Versions: FC12A; FC12AD.
Language: FORTRAN, Date: March 1973, Size: 2.8K; 127 cards.
Origin: R. Fletcher, Harwell.

FC13A

To compute values of Dawson's Integral

$$F(x) = e^{-x^2} \int_0^x e^{t^2}\,dt$$

for x real.

The following approximations are used,

(a) $x^2 < 6$ a Taylor series expansion.

(b) $6 \leqslant x^2 \leqslant 36$ a series expansion in $(x^2-x_0^2)$ where x_0 is a lower limit of one of eight subranges in $[6,36]$.

(c) $x^2 \geqslant 36$ an asymptotic series.

Accuracies: FC13A $< 10^{-5}$ $|F(x)|$; FC13AD $< 10^{-13}$ $|F(x)|$.
Versions: FC13A; FC13AD.
Calls: PB01AD.
Language: FORTRAN, Date: May 1966, Size: 5.8K; 154 cards.
Origin: A.R. Curtis, Harwell.

FD01AS

Given an integer n will compute 2^n.

Versions: FD01AS; there is no double length version.
Language: 360/BAL, Date: November 1971, Size: .1K; 73 cards.
Origin: M.J. Hopper, Harwell.

FD01BS

Given an integer n will compute 16^n.

Versions: FD01BS; there is no double length version.
Language: 360/BAL, Date: November 1971, Size: .1K; 66 cards.
Origin: M.J. Hopper, Harwell.

FF01A

Computes values of the Bessel functions $J_0(x)$ and $Y_0(x)$. A Chebyshev series in x is used if $0 \leqslant x \leqslant 8$ and a similar series in $\frac{1}{x}$ if $x > 8$, see, C.W. Clenshaw, 'Mathematical Tables', Vol. 5, NPL.

Accuracies: FF01A 6 sig. figs.; FF01AD 9 sig. figs.; except near $x = 8$ where the accuracies may be inferior to those given.
Versions: FF01A; FF01AD.
Language: FORTRAN, Date: April 1963, Size: 1.9K; 102 cards.
Origin: S. Marlow, Harwell.

FF02A

To compute values of the Bessel functions $J_1(x)$ and $Y_1(x)$. A Chebyshev series in x is used if $0 \leqslant x \leqslant 8$ and a similar series in $\frac{1}{x}$ if $x > 8$, see, C.W. Clenshaw, 'Mathematical Tables', Vol. 5, NPL.

Accuracies: FF02A 6 sig. figs.; FF02AD 9 sig. figs.; except near $x = 8$ where the accuracies will be inferior to those given.
Versions: FF02A; FF02AD.
Language: FORTRAN, Date: June 1963, Size: 1.9K; 102 cards.
Origin: S. Marlow, Harwell.

FF03A

Computes values of the Bessel functions $I_0(x)$ and $K_0(x)$. A Chebyshev series in x is used if $0 \leq x \leq 8$ and a similar series in $\frac{1}{x}$ if $x > 8$, see, C.W. Clenshaw, 'Mathematical Tables', Vol. 5, NPL.

Accuracies:	FF03A 6 sig. figs.; FF03AD 9 sig. figs.; except near $x = 8$ where the accuracies will be inferior to those given.
Versions:	FF03A; FF03AD.
Language:	FORTRAN, Date: Dec. 1966, Size: 1.8K; 97 cards.
Origin:	S. Marlow, Harwell.

FF04A

Computes values of the Bessel functions $I_1(x)$ and $K_1(x)$. A Chebyshev series in x is used if $0 \leq x \leq 8$ and a similar series in $\frac{1}{x}$ if $x > 8$, see, C.W. Clenshaw, 'Mathematical Tables', Vol. 5, NPL.

Accuracies:	FF04A 6 sig. figs.; FF04AD 9 sig. figs.; except near $x = 8$ where the accuracies will be inferior to those given.
Versions:	FF04A; FF04AD.
Language:	FORTRAN, Date: Dec. 1966, Size: 1.7K; 94 cards.
Origin:	S. Marlow, Harwell.

FF05A

Given $x \geq 0$ computes the values of the Spherical Bessel functions

$$j_n(x) = \left(\frac{\pi}{2x}\right)^{\frac{1}{2}} J_{n+\frac{1}{2}}(x)$$

for $n = 0$ up to N, $N \leq 29$.

The method used is based upon the recurrence relation

$$j_{n+1}(x) = \left(\frac{2n+1}{x}\right) j_n(x) - j_{n-1}(x)$$

given by F.J. Corbaló and J.L. Uretsky, J.A.C.M., Vol. 6, No. 3.

Accuracies:	FF05A 6 sig. figs.; FF05AD 8 sig. figs.
Versions:	FF05A; FF05AD.
Language:	FORTRAN, Date: Dec. 1963, Size: 3.2K; 99 cards.
Origin:	F.R. Hopgood*, Harwell.

FF06A

Given $x > 0$ computes values of all the Bessel functions ber(x), bei(x), ker(x), kei(x), ber'(x), bei'(x), ker'(x) and kei'(x).

A Chebyshev series in x is used if $x \leq 10$ and a similar series in $\frac{1}{x}$ if $x > 10$, See, F.D. Burgoyne, Maths. Comp., Vol. 17, No. 83, 1963.

Accuracies: FF06A sig. figs.; FF06AD 8 sig. figs. for $x \leqslant 10$, 6 sig. figs. otherwise.

Versions: FF06A; FF06AD.

Language: FORTRAN, Date: Oct. 1964, Size: 3.1K; 83 cards.

Origin: S. Marlow, Harwell.

FG01A

To compute the various vector coupling coefficients (3-j, 6-j, 9-j and their kindred) of the theory of angular momentum in quantum mechanics, i.e. the Wigner 3-j, 6-j and 9-j symbols, the Clebsch-Gordan and Wigner coefficient, the Racah coefficient and Jahn's U-function.

Versions: FG01A; FG01AD.

Language: FORTRAN, Date: Aug. 1969, Size: 10.7K; 448 cards.

Origin: J. Soper, Harwell.

FT01A

Calculates discrete fourier transforms. Given equally spaced complex data $f(n)$ $n = 0,1,2,\dots,N-1$, of period N, it calculates the transform

$$\hat{f}(m) = \sum_{n=0}^{N-1} f(n) \exp\left\{\frac{2\pi imn}{N}\right\} \qquad m = 0,1,2,\dots,N-1$$

or alternatively given $\hat{f}(m)$ $m = 0,1,2,\dots,N-1$ calculates the inverse transform

$$f(n) = \frac{1}{N} \sum_{m=0}^{N-1} \hat{f}(m) \exp\left\{\frac{-2\pi imn}{N}\right\} \qquad m = 0,1,2,\dots,N-1$$

where in both cases N must be a power of 2.

The 'Fast Fourier Transform' method is used, see, W.M. Gentleman and G. Sande, 'Fourier Transforms in Place', Proc. Fall Joint Computer Conference, 1966.

Versions: FT01A; FT01AD.

Language: FORTRAN, Date: Sept. 1967, Size: 1.7K; 116 cards.

Origin: A.R. Curtis, Harwell.

G. Geometrical Problems

GA01A

To calculate the cartesian co-ordinates x,y,z of a point given in spherical co-ordinates r,θ,ϕ; or vice versa.

Versions: GA01A; GA01AD.

Language: FORTRAN, Date: April 1964, Size: .8K; 30 cards.

Origin: A. Hearn*, Harwell.

GA02A

Calculates the area bounded by a contour f(x,y) = c and the side(s) of a triangle. The triangle is assumed to have vertices (0,0), (2.0), (2.2) and the user must provide values of the function f(x,y) at the vertices and mid-points of the sides of the triangle. A point where the contour cuts the triangle must also be given.

The function f(x,y) is approximated over the triangle by a quadratic form defined using the six given function values.

Versions: GA02A; GA02AD.
Language: FORTRAN, Date: Aug. 1964, Size: 6.1K; 320 cards.
Origin: D. Miller*, Harwell.

GA03A

Constructs a system of plane contours $f(x,y) = c_k$ $k=1,2,\ldots,N$ over the rectangular region $x_1 \leq x \leq x_n$, $y_1 \leq y < y_n$ and calculates the areas between successive contours.

A mesh of isoceles triangles is constructed over the region and the contours are generated using linear interpolation.

The user must provide code to evaluate f(x,y) at any point in the region.

Versions: GA03A; GA03AD.
Language: FORTRAN, Date: May 1967, Size: 3.1K; 117 cards.
Origin: E.J. York*, Harwell.

GA04A

To compute the solid angle subtended by a disc of unit radius from a general point (r,h) in the plane perpendicular to the plane of the disc.

The integral expression for the solid angle is represented in terms of complete and incomplete elliptic integrals of the first kind, see M. Ruffle, AERE - R.5419.

Versions: GA04A; GA04AD.
Calls: FB01A and FB02A.
Language: FORTRAN, Date: April 1966, Size: 1.1K; 34 cards.
Origin: M. Ruffle*, Harwell.

GA05A

To efficiently determine whether a point (x,y) is interior or exterior to a given closed region in the x,y plane. The boundary of the region may be specified as a polygon or by a pair of parametric cubic splines, x(t) and y(t). Polar co-ordinates may optionally be used.

The method is described in J.K. Reid, AERE - R.7298.

Versions: GA05A; GA05AD.
Calls: KB01A and NB01A.
Language: FORTRAN, Date: October 1972, Size: 7.9K; 400 cards.
Origin: J.K. Reid, Harwell.

I. Integer FUNCTIONS: including system facilities for FORTRAN programmers (see also section Z).

IA01AS

Provides the FORTRAN user with facilities to allocate main storage during execution, i.e. provides the user with the 360/OS GETMAIN and FREEMAIN facilities.

The routine can be used to get any number of areas of main core, to free areas and to obtain the size of the largest currently available contiguous free area. The areas obtained by IA01AS are referenced using normal FORTRAN arrays but using a displacement on the subscript which is supplied by IA01AS.

Remark: The routine provides a limited form of dynamic allocation of arrays at execution time but is highly system dependent and should be avoided in programs intended to be computer independent.

Language: 360/BAL, Date: June 1970, Size: .5K; 192 cards.
Origin: M.J. Hopper, Harwell.

IC01AS

Given a character string and a search character locates the position of the first occurrence of the character in the string or optionally locates the first non-occurrence of the character.

In either case the search may be made in a forward direction from the beginning or in a backward direction from the end.

Versions IC01AS.
Language: 360/BAL, Date: October 1971, Size: .1K; 79 cards.
Origin: M.J. Hopper, Harwell.

IC02AS

To compare two character strings giving a less than, an equal to or a greater than result.

Versions: IC02AS.
Language: 360/BAL, Date: October 1971, Size: .1K; 54 cards.
Origin: M.J. Hopper, Harwell.

ID01AS

Computes the integer part of $\log_2(|x|)$ given a real floating point number x.

Versions: ID01AS; there is no double length version.
Language: 360/BAL, Date: November 1971, Size: .1K; 66 cards.
Origin: M.J. Hopper, Harwell.

ID01BS

Computes the integer part of $\log_{16}(|x|)$ given a real floating point number x.

Versions: ID01BS; there is no double length version.
Language: 360/BAL, Date: November 1971, Size: .1K; 55 cards.
Origin: M.J. Hopper, Harwell.

ID02A

Finds K the H.C.F. of two given integers I and J. It also finds integers M and N such that

$$M*I - N*J = K \qquad K > 0$$

and M*I, N*J $\geq$ 0 and such that $\max\{|N|,|M|\}$ is minimized.

Language: FORTRAN, Date: July 1964, Size: .6K; 36 cards.
Origin: A. Gavan*, Harwell.

ID03A

Calculates the number of seconds elapsed between two given times given in units of years, months, days and hours. The times must lie in the range 1st March, 1900 to 28th Feb. 2000.

Language: FORTRAN, Date: Aug. 1964, Size: 1K; 44 cards.
Origin: A. Bailey*, Harwell.

K. Sorting and using sorted information

KA01AS

To locate a specified entry in a given table. The fixed length entries in the table are assumed to have been ordered on a key field within each entry into either ascending or descending order, the ordering being specified in an index array. The key field is assumed to contain non-numeric information and may be any length up to 256 characters.

A simple binary search technique is used.

Remark: The subroutine has been designed to be used on tables sorted by KB10AS but may be used on its own to generate and maintain ordered tables.
Language: 360/BAL, Date: Aug. 1970, Size: .2K; 120 cards.
Origin: M.J. Hopper.

KB01A

To sort an array of numbers into ascending order. The 'Quicksort' method is used, see, C.A.R. Hoare, 'Quicksort', Computer Journal, April 1962.

Versions: KB01A; KB01AD; KB01B integers; KB01C halfword integers.
Language: FORTRAN, Date: May 1966, Size: 2.7K; 122 cards.
Origin: M. Reynolds*, Harwell.

KB02A

To sort an array of numbers into descending order. The 'Quicksort' method is used, see, C.A.R. Hoare, 'Quicksort', Computer Journal, April 1962.

Versions: KB02A; KB02AD; KB02B integers; KB02C halfword integers.
Language: FORTRAN, Date: May 1966, Size: 2.6K; 115 cards.
Origin: M. Reynolds*, Harwell.

KB03A

To sort an array of numbers into ascending order maintaining an index array to preserve a record of the original order.

The 'Quicksort' method is used, see, C.A.R. Hoare, 'Quicksort', Computer Journal, April 1962.

Versions: KB03A; KB03AD; KB03B integers; KB03C halfword integers.
Language: FORTRAN, Date: May 1966, Size: 2.9K; 124 cards.
Origin: M. Reynolds*, Harwell.

KB04A

To sort an array of numbers into descending order maintaining an index array to preserve a record of the original order.

The 'Quicksort' method is used, see, C.A.R. Hoare, 'Quicksort', Computer Journal, April 1962.

Versions: KB04A; KB04AD; KB04B integers; KB04C halfword integers.
Language: FORTRAN, Date: May 1966, Size: 2.9K; 124 cards.
Origin: M. Reynolds*, Harwell.

KB10AS

To sort a table of fixed length entries into ascending or descending order, sorting on a key field within each entry. The keyfield is assumed to contain non-numeric information.

The entries in the table are not moved but rather an index array is returned specifying the required ascending or descending order.

Subsorting on several fields may be performed.

Remark: Excellent for sorting text type information.
Language: 360/BAL, Date: May 1969, Size: .7K; 393 cards.
Origin: K. Moody, IBM.

KB11A

To sort n numbers from an array of m numbers, n < m. Options are provided for sorting either the first n smallest or the first n largest numbers either in terms of their algebraic values or their absolute values.

Versions: KB11A; KB11AD; KB11AI.
Calls: MX01A and MX02A.
Language: FORTRAN, Date: July 1972, Size: 2.7K; 160 cards.
Origin: M.J. Hopper, Harwell.

KC01A

Given a set of intervals on the real line, the routine finds the smallest set of disjoint intervals whose union is the union of the original set, i.e. given intervals which overlap one another it will reduce the number of intervals by merging together overlapping intervals.

Versions: KC01A; KC01AD.

Language: FORTRAN, Date: Aug. 1964, Size: .5K; 19 cards.

Origin: D. Willis*, Harwell.

KC02A

Given two sets of non-overlapping intervals, the routine merges the two sets together and returns the smallest set of non-overlapping intervals which are common to both the original sets.

Versions: KC02A; KC92AD.

Language: FORTRAN, Date: Aug. 1964, Size: .7K; 32 cards.

Origin: D. Willis*, Harwell.

L. Linear programming

LA01A

Solves the general linear programming problem, i.e., find $\underline{x}$ which minimizes the linear function

$$f(\underline{x}) = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

Subject to linear constraints

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \leq b_i \qquad i=1,2,\dots,l$$

$$a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n = b_j \qquad j=l+1,\dots,m$$

where $x_i \geqslant 0 \quad i=1,2,\dots,n.$

The Revised Simplex method is used where an inverse of the basis matrix is maintained and updated at each iteration.

Versions: LA01A; LA01AD.

Calls: MC03AS.

Language: FORTRAN, Date: Jan. 1966, Size: 50.8K; 359 cards.

Origin: M.J. Hopper, Harwell.

LA02A

To find a feasible point to a set of linear constraints, i.e. find values $x_1, x_2, \dots, x_n$ which satisfy given constraints of the form

$$l_j \geqslant x_j \geqslant u_j \qquad j=1,2,\dots,n$$

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \geqslant d_i \qquad i=1,2,\dots,m$$

where additionally any of the inequalities may be strict equalities.

The method is described in R. Fletcher, 'The calculation of feasible points for linearly constrained optimization problems', AERE - R.6354.

Versions:	LA02A; LA02AD.
Calls:	MB01B and MC03AS.
Language:	FORTRAN, Date: July 1970, Size: 6.6K; 238 cards.
Origin:	R. Fletcher, Harwell.

LA03A

To factorize a sparse matrix A = LU, solve corresponding systems of equations, and update the factorization when a column of A is modified or replaced.

The method of factorization and treatment of sparsity is similar to that used in MA18A, which is documented in the A.E.R.E. report R.6844.

The subroutine has been written primarily to handle the functions normally performed on bases in linear programming problems, but its usefulness is not necessarily restricted to that field.

Versions:	LA03A; LA03AD.
Calls:	KB10AS.
Language:	FORTRAN, Date: June 1973, Size: 8.5K; 471 cards.
Origin:	J.K. Reid, Harwell.

M. Linear Algebra (see also section E)

MA01B

To solve a system of n linear algebraic equations in n unknowns with one or more right-hand sides

$$\sum_{j=1}^{n} a_{ij}x_{jl} = b_{il} \qquad i=1,2,\ldots,n \qquad l=1,2,\ldots,k$$

and optionally compute the inverse matrix A^{-1} of the equation coefficient matrix $A = \{a_{ij}\}$.

Gaussian elimination with partial pivoting is used, see 'Modern Computing Methods', NPL, 1957, with double length accumulation of inner products.

Remark:	Superseded by MA21A.
Versions:	MA01B; MA01BD.
Calls:	MC03AS.
Language:	FORTRAN, Date: Feb. 1963, Size: 4.4K; 112 cards.
Origin:	E.J. York*, Harwell.

MA07A

To solve a system of n linear algebraic equations in n unknowns

$$\sum_{j=1}^{n} a_{ij}x_{j} = b_{i} \qquad i=1,2,\ldots,n$$

when the equation coefficient matrix $A = \{a_{ij}\}$ is band structured.

The equations are solved by the method of Gaussian elimination without interchanges of rows so that stability is not guaranteed. The matrix A is presented to the routine in a compact form.

When it is required to solve several systems which have identical left-hand sides A, the routine can either be re-entered in a way that saves repeating the elimination phase, or will except more than one right-hand side.

Remark:	The pivoting strategy used makes the routine unreliable for systems which are not positive definite or diagonally dominant, try MA07B.
Versions:	MA07A; MA07AD.
Language:	FORTRAN, Date: June 1964, Size: 1.7K; 61 cards.
Origin:	D. Russell, Atlas Laboratory, Chilton, Berks.

MA07B

To solve a system of n linear algebraic equations in n unknowns,

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

where the equation coefficient matrix $A = \{a_{ij}\}$ has a band structure.

The equations are solved by the method of Gaussian elimination with partial pivoting. The matrix A is passed to the routine in a compact form.

When several systems with identical left-hand side matrices A are to be solved the routine may be re-entered in a way that avoids repeating the elimination phase.

Remark:	For positive definite band systems see also MA15A.
Versions:	MA07B; MA07BD.
Language:	FORTRAN, Date: Jan. 1970, Size: 2.4K; 91 cards.
Origin:	J.K. Reid, Harwell.

MA08A

Forms the normal equations of the linear least squares problem, i.e. given an overdetermined system of linear algebraic equations

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

or more compactly $A\underline{x} = \underline{b}$, the subroutine sets up the equations $A^TA\underline{x} = A^T\underline{b}$. There may be more than one right-hand size.

Remark:	If the solution is required see MA09A.
Versions:	MA08A; MA08AD.
Calls:	MC03AS.
Language:	FORTRAN, Date: June 1964, Size: 1.3K; 19 cards.
Origin:	M.J. Hopper, Harwell.

MA09A

Solves the linear least squares problem by the so-called normal equations method, i.e. given an overdetermined system of linear algebraic equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \qquad i=1,2,\ldots,m \qquad m \geqslant n; \qquad A\underline{x} = \underline{b}$$

sets up and solves the system $A^TA\underline{x} = A^T\underline{b}$. The solution so obtained is such that the sum of squares of the equation residuals $\sum_{i=1}^{m}\left\{\sum_{j=1}^{n} a_{ij}x_j - b_i\right\}^2$ is a minimum.

Cholesky decomposition is used to solve the system. Equations with more than one right-hand side can be solved and the user has options to obtain equation residuals, sum of squares value and the inverse $[A^TA]^{-1}$ which is usually required for the variance-covariance matrix.

Remark:	For a large number of unknowns the method is likely to give poor results particularly when applied to fitting polynomials; try MA14A or for polynomials VC01A.
Versions:	MA09A; MA09AD.
Calls:	MA08A, MA10A and MC03AS.
Language:	FORTRAN, Date: June 1964; Size: 1.5K; 19 cards.
Origin:	M.J. Hopper, Harwell.

MA10A

To solve a system of n linear algebraic equations in n unknowns

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

where the coefficient matrix $A = \{a_{ij}\}$ is symmetric positive definite. The inverse matrix A^{-1} is optionally computed.

Symmetric Cholesky decomposition is used with inner products accumulated double length.

Remark:	Superseded by MA22A.
Versions:	MA10A; MA10AD.
Calls:	MC03AS.
Language:	FORTRAN, Date: May 1964, Size: 3.4K; 88 cards.
Origin:	M.J. Hopper, Harwell.

MA11B

To solve an overdetermined system of m linear algebraic equations in n unknowns in the minimax sense, i.e. given equations

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

find the solution x_j $j=1,2,\dots,n$ such that

$$\max_i \left\{ \left| \sum_{j=1}^{n} a_{ij}x_j - b_i \right| \right\}$$

is minimized.

The problem is posed as an n by m dual linear programming problem which is solved using a special adaptation of the Simplex algorithm.

The routine returns residual values and may be requested to print solution details.

The routine may be applied to the problem of approximation by linear combinations of general functions over a discrete point set.

Versions: MA11B; MA11BD.
Language: FORTRAN, Date: Sept. 1965, Size: 50.7K; 245 cards.
Origin: M.J. Hopper, Harwell.

MA12A

To solve a system of n linear algebraic equations in n unknowns

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

when the coefficient matrix $A = \{a_{ij}\}$ is upper Hessenberg or upper Hessenberg squared.

The routine may be re-entered to provide additional right-hand sides for the economic solution of systems with the same coefficient matrix A.

Gaussian elimination with partial pivoting is used accumulating inner products double length.

Versions: MA12A; MA12AD.
Calls: MC03AS.
Language: FORTRAN, Date: July 1966, Size: 3.6K; 167 cards.
Origin: M. Reynolds*, Harwell.

MA14A

To calculate a least squares solution to an over-determined system of m linear equations in n unknowns, i.e. given equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \qquad i=1,2,\dots,m \qquad m \geqslant n$$

calculate the solution vector $\underline{x} = \{x_j\}$ such that the sum of squares of residuals

$$F(\underline{x}) = \sum_{i=1}^{m} \left\{ \sum_{j=1}^{n} a_{ij}x_j - b_i \right\}^2$$

is minimised. The user may specify that the first k equations, $n \geq k \geq 0$, are satisfied exactly in which case the least squares solution to the constrained problem is calculated.

There is a re-entry facility which allows further systems having the same left-hand sides to be solved economically. Another entry may be called to obtain solution standard deviations and the variance-covariance matrix for the previous calculation. The automatic printing of results and the calculation of equation residuals are additional options.

The routine can be used to solve the general linear least squares data fitting problem with, or without, equality side conditions.

Versions:	MA14A; MA14AD.
Calls:	OA02A and MC03AS.
Language:	FORTRAN, Date: June 1968, Size: 7.8K; 398 cards.
Origin:	M.J. Hopper, Harwell.

MA15C

To solve a system of n linear algebraic equations in n unknowns

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

where the coefficient matrix $A = \{a_{ij}\}$ is band structured, symmetric and positive definite.

Symmetric (LDL^T) decomposition is used and full advantage is taken of any variation in band width. For very large systems it uses scratch space on backing store, otherwise it uses fast core storage.

There is a re-entry facility which allows further systems with the same coefficient matrix to be solved economically. The user must supply a subroutine to pass to MA15C the row elements of A.

The method is described in J.K. Reid, AERE Report - R.7119.

Remarks:	This is an improved version of MA15A which has now been removed from the Library.
Versions:	MA15C; MA15CD.
Calls:	MC02AS.
Language:	FORTRAN, Date: April 1972, Size: 3.4K; 147 cards.
Origin:	J.K. Reid, Harwell.

MA16A

To solve a system of n linear algebraic equations in n unknowns

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

when the coefficient matrix $A = \{a_{ij}\}$ is symmetric positive definite and is very large and sparse. It uses the method of conjugate gradients, see, J.K. Reid, AERE - R.6545.

The user is required to write code to multiply the matrix A into a vector where full advantage of the sparsity of A may be taken into account in the code.

Versions: MA16A; MA16AD.
Language: FORTRAN, Date: Oct. 1970, Size: 1.1K; 55 cards.
Origin: J.K. Reid, Harwell.

MA17A

To solve a system of n linear algebraic equations in n unknowns

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \qquad i=1,2,\ldots,n; \qquad A\underline{x} = \underline{b}$$

where the coefficient matrix $A = \{a_{ij}\}$ is large and sparse and symmetric positive definite.

It provides facilities to (a) decompose the matrix A into factors LDL^T, where L is lower triangular and D diagonal, (b) to solve the system $A\underline{x} = \underline{b}$ or compute the product $A\underline{b}$, (c) factorize economically a new matrix which has the same sparsity structure as a previous one.

MA17A is a variant of MA18A the subroutine for the general linear sparse case and provides similar options, see MA18A.

The method is described in J.K. Reid, AERE Report - R.7119.

Versions: MA17A; MA17AD.
Calls: KB10AS.
Language: FORTRAN, Date: April 1972, Size: 7K; 355 cards.
Origin: J.K. Reid, Harwell.

MA18A

To solve a system of n linear algebraic equations in n unknowns

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \qquad i=1,2,\ldots,n; \qquad A\underline{x} = \underline{b}$$

where the coefficient matrix $A = \{a_{ij}\}$ is large and sparse, i.e. contains mostly zero elements.

It provides facilities to (a) decompose the matrix A into triangular factors, (b) to solve the system $A\underline{x} = \underline{b}$ or solve $A^T\underline{x} = \underline{b}$ or carry out either of the operations $A\underline{b}$ or $A^T\underline{b}$, (c) to factorize economically a new matrix which has the same sparsity structure as a previous one, and, (d) load a new matrix into the list processing form required by facility (c). For (d) the user must supply a subroutine to pass over the columns of the new matrix one at a time.

Optionally, automatic implicit scaling of equations and unknowns is performed. Gaussian elimination is used combined with list processing methods to economise on storage. The pivotal strategies used are designed to keep down the growth in the number of non-zero elements. The degree to which the routine does this is under the control of the user giving a choice of error growth against savings in storage. A measure of the error growth is returned.

The method is described in A.R. Curtis and J.K. Reid, AERE - R.6844.

Versions:	MA18A; MA18AD.
Calls:	MC12A and KB10AS.
Language:	FORTRAN, Date: June 1971, Size: 12.5K; 636 cards.
Origin:	A.R. Curtis and J.K. Reid, Harwell.

MA20A

To solve an overdetermined system of m linear equations in n unknowns in the L_1 sense, i.e. given equations

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

find a solution x_j $j = 1,2,\ldots,n$ such that

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

is minimized. There is an option of constraining the solution so that $x_j \geqslant 0$ $j = 1,2,\ldots,n$.

The problem is posed as a primal LP problem and solved using a variant of the standard Simplex method which has the ability to pass through several Simplex vertices at each iteration, see I. Barrowdale and F.D.K. Roberts; an improved algorithm for discrete L_1 linear approximation; Maths. Dept. preprint, Univ. of Victoria, Canada.

The routine can be used for L_1 linear data fitting problems.

Versions:	MA20A; MA20AD. MA20B; MA20BD, B versions constrain $x_j \geqslant 0$ $j = 1,2,\ldots,n$.
Language:	FORTRAN, Date: June 1972, Size: 3.9K; 172 cards.
Origin:	I. Barrowdale, Univ. of Victoria, Canada.

MA21A

Given an n by n matrix $A = \{a_{ij}\}$ performs one or more of the following tasks.

(a) solves the system of linear algebraic equations

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

given the right-hand sides b_i $i=1,2,\ldots,n$, and provides a re-entry facility for the rapid solution of further systems of equations which have the same elements a_{ij}.

(b) computes the inverse matrix A^{-1} of A.

(c) computes the value of the determinant of A.

The subroutine will optionally perform iterative refinement in order to both improve the accuracy of the answer (solution or inverse) and provide error estimates based either on the precision of the computer or on user supplied accuracy information. An option for scaling the matrix is provided.

The method is basically Gaussian elimination with partial pivoting, implicit scaling and iterative refinement and applying small random perturbations in order to estimate errors, see S. Marlow and J.K. Reid, AERE - R.6899.

Versions:	MA21A; MA21AD.
Calls:	FA01AS, MC03AS and MC10A.
Language:	FORTRAN, Date: Aug. 1971, Size: 9K; 353 cards.
Origin:	S. Marlow and J.K. Reid, Harwell.

MA22A

Given a symmetric positive definite n by n matrix $A = \{a_{ij}\}$ this subroutine performs one or more of the following tasks.

(a) Solves the system of linear algebraic equations

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

given the right-hand sides b_i $i=1,2,\ldots,n$, and provides a re-entry facility for the rapid solution of further systems of equations which have the same elements a_{ij}.

(b) computes the inverse matrix A^{-1} of A.

(c) computes the value of the determinant of A.

This subroutine provides for the symmetric positive definite case all the facilities provided by MA21A for the general real case. Symmetric decomposition is used in the elimination phase otherwise the method details are the same as for MA21A.

Versions:	MA22A; MA22AD.
Calls:	FA01AS and MC03AS.
Language:	FORTRAN, Date: Aug. 1971, Size: 8K; 285 cards.
Origin:	S. Marlow and J.K. Reid, Harwell.

MA23A

Given a complex n by n matrix $A = \{a_{ij}\}$ and a complex right hand side b_i $i=1,n$ this subroutine will solve the complex linear equations, compute the inverse matrix or evaluate the determinant. It is a complex version of MA21A and offers identical facilities.

The FORTRAN COMPLEX facility is used.

Versions: MA23A; MA23AD.
Calls: FA01AS, ME06AS and ME07A.
Language: FORTRAN, Date: Aug. 1971, Size: 12.3K; 363 cards.
Origin: S. Marlow and J.K. Reid, Harwell.

MA24A

Given a complex Hermitian n by n matrix $A = \{a_{ij}\}$ and a complex right hand side b_i $i=1,n$ this subroutine will solve the complex linear equations, compute the inverse matrix or evaluate the determinant. It is a Hermitian version of MA22A and offers identical facilities.

The FORTRAN COMPLEX facility is used.

Versions: MA24A; MA24AD.
Calls: FA01AS and ME06AS.
Language: FORTRAN, Date: Aug. 1971, Size: 10.9K; 285 cards.
Origin: S. Marlow and J.K. Reid, Harwell.

MB01B

To compute the inverse matrix A^{-1} of an n by n real matrix $A = \{a_{ij}\}$.

Gaussian elimination with partial pivoting is used with inner products accumulated double length.

Remark: Superseded by MA21A.
Versions: MB01B, MB01BD.
Calls: MC03AS.
Language: FORTRAN, Date: Jan. 1964, Size: 3.4K; 88 cards.
Origin: L. Morgan*, Harwell.

MB02A

To evaluate the adjoint (or adjugate) matrix of an n by n real matrix $A = \{a_{ij}\}$, $n \leqslant 20$, and A may be singular.

The determinant is also evaluated and optionally the inverse of A (A non-singular) instead of the adjoint. The relationship $\mathrm{Adj}(A) = \det(A).A^{-1}$ is used.

Gaussian elimination with partial pivoting is used.

Remark: There are more efficient routines in the library for evaluating determinants and inverses.
Versions: MB02A; MB02AD.
Language: FORTRAN, Date: Dec. 1963, Size: 4K; 98 cards.
Origin: G. Packham*, Harwell.

MB04A

Given A^{-1} the inverse matrix of a partitioned n by n real symmetric matrix

$$A = \left[\begin{array}{c|c} A_1 & A_2^T \\ \hline A_2 & A_3 \end{array}\right]$$

where A_1 is (n-1) by (n-1), A_2 a row vector and A_3 a scalar; this subroutine calculates the (n-1) by (n-1) inverse matrix A_1^{-1} of the submatrix A_1.

Versions: MB04A; MB04AD.
Language: FORTRAN, Date: May 1964, Size: .6K; 14 cards.
Origin: M.J.D. Powell, Harwell.

MB05A

Given a partitioned n by n real symmetric matrix

$$A = \left[\begin{array}{c|c} A_1 & A_2^T \\ \hline A_2 & A_3 \end{array}\right]$$

where A_1 is (n-1) by (n-1), A_2 is a row vector and A_3 a scalar, and given A_1^{-1} the inverse of the submatrix A_1; the subroutine calculates A^{-1} the inverse of A.

Versions: MB05A; MB05AD.
Language: FORTRAN, Date: May 1964, Size: 1K; 26 cards.
Origin: M.J.D. Powell, Harwell.

MB10A

To calculate A^+ the generalized inverse of an m by n rectangular matrix A, i.e. such that $AA^+A = A$.

The user may either specify the rank of A or provide an estimate of the accuracy of the elements in A in order that the rank may be estimated automatically.

Versions: MB10A; there is no double precision version.
Calls: MC03AS.
Language: FORTRAN, Date: June 1970, Size: 4.1K; 144 cards.
Origin: M.J.D. Powell, Harwell.

MB11A

To calculate A^+ the generalized inverse of an m by n $(m \leq n)$ rectangular matrix A in the special case that the rank of A is equal to m, i.e. such that $AA^+A = A$ which with full rank can be defined as $A^+ = A^T(AA^T)^{-1}$.

Householder type orthogonal transformations with row and column interchanges are used in a method described in M.J.D. Powell, AERE - R.6072.

Versions: MB11A; MB11AD.
Language: FORTRAN, Date: May 1969, Size: 3.3K; 153 cards.
Origin: M.J.D. Powell, Harwell.

MC01AS

To calculate the product of two matrices, i.e. given compatible matrices $A = \{a_{ij}\}$ and $B = \{b_{ij}\}$ forms the product matrix $C = \{c_{ij}\}$ where

$$c_{ij} = \sum_{K=1}^{n} a_{iK} b_{Kj} \qquad i=1,2,\ldots,l; \qquad j=1,2,\ldots,m$$

Inner products are accumulated double length.

Versions:	MC01AS; MC01AD.
Language:	360/BAL, Date: Sept. 1967, Size: .2K; 74 cards.
Origin:	R.C.F. McLatchie, Harwell.

MC02AS

To evaluate the inner product $\underline{a}^T\underline{b}$ of two vectors $\underline{a} = \{a_i\}$ and $\underline{b} = \{b_i\}$ of length n, i.e. evaluate

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

The products are accumulated double length, and use of MC02AS will in general result in faster execution compared with any equivalent FORTRAN code.

Versions:	MC02AS; MC02AD; MC02BS; MC02BD; in the B versions the elements of the vectors can be stored in any regular fashion, i.e. the user gives the first two elements of each vector and the subroutine locates the rest by using the displacement of the second from the first and the length of the vector.
Language:	360/BAL, Date: Aug. 1967, Size: .1K; 26 cards.
Origin:	R.C.F. McLatchie, Harwell.

MC03AS

To evaluate the sum of a number c and an inner product $\underline{a}^T\underline{b}$ of two vectors $\underline{a} = \{a_i\}$ and $\underline{b} = \{b_i\}$ using double length accumulation of intermediate results to minimize rounding errors, i.e. evaluate

$$s = \pm c \pm \sum_{i=1}^{n} a_i b_i$$

the signs being specified. The elements $\underline{a}$ and $\underline{b}$ may be stored in any regular fashion. The first two elements of each vector are given and the displacement of the second from the first and the length n are used to locate the remainder. The use of MC03AS will in general result in faster execution than any equivalent FORTRAN code.

Versions:	MD03AS; MC03AD; the result s will on request be returned to full double precision in the single precision version MC03AS.
Language:	360/BAL, Date: Aug. 1967, Size: .2K; 85 cards.
Origin:	R.C.F. McLatchie, Harwell.

MC04B

Transforms an n by n real symmetric matrix A in to a tri-diagonal matrix having the same eigenvalues as A.

Householder's method is used, see, J.H. Wilkinson, Numerische Mathematik, Dec. 1962.

Versions: MC04B; MC04BD.
Language: FORTRAN, Date: 1966, Size: 2.1K; 61 cards.
Origin: Harwell.

MC05AS

To evaluate the sum of triple products of elements of three given vectors $\underline{a} = \{a_i\}$, $\underline{b} = \{b_i\}$ and $\underline{c} = \{c_i\}$ of length n, i.e.evaluate

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

the products are accumulated double length and the use of MC05AS will in general result in faster execution time than any equivalen FORTRAN code.

Remark: Useful for weighted linear least squares problems where an inner product of two vectors $\underline{x}$ and $\underline{y}$ modified by a vector of weights $\underline{w}$ is required.
Versions: MC05AS; MC05AD.
Language: 360/BAL, Date: December 1971, Size: .1K; 66 cards.
Origin: M.J. Hopper, Harwell.

MC06A

Applies the Gram-Schmidt orthogonalization process to n vectors in an m dimensional space, $n \leq m$.

The vectors are passed as adjacent columns in a two dimensional array. Double length accumulation of inner products is performed.

Versions: MC06A; MC06AD.
Calls: MC02BS.
Language: FORTRAN, Date: Sept. 1963, Size: 2.1K; 32 cards.
Origin: Osbourne*, Harwell.

MC07A

Given an n by n real symmetric matrix A calculates the coefficients of its characteristic polynomial the roots of which will be the eigenvalues of A.

The matrix is reduced to tri-diagonal form by Householder's method and the coefficients of the polynomial calculated using a recurrence relation.

Remark: This method should not be used to find eigenvalues but rather EA07C.
Versions: MC07A; MC07AD.
Calls: MC04B.
Language: FORTRAN, Date: Jan. 1964, Size: 1.8K; 39 cards.
Origin: M.J.D. Powell, Harwell.

MC08A

Transforms an n by n real matrix A to an upper Hessenberg matrix having the same eigenvalues as A.

Householder's method is used, see, J.H. Wilkinson, Numerische Mathematik, 1962.

Versions: MC08A; MC08AD.
Language: FORTRAN, Date: Nov. 1966, Size: 2.3K; 59 cards.
Origin: E.J. York*, Harwell.

MC09A

Given a sparse matrix A stored in a compact form and two vectors $\underline{x}$ and $\underline{y}$, evaluates either of the matrix-vector products $\underline{a} = \underline{y} + A\underline{x}$ or $\underline{b} = \underline{y} + A^T\underline{x}$.

Versions: MC09A; MC09AD.
Language: FORTRAN, Date: February 1972, Size: .8K; 21 cards.
Origin: J.K. Reid, Harwell.

MC10A

Calculates scaling factors for the rows and columns of an n by n real matrix.

If the scaling is applied before Gaussian elimination with pivoting the choice of pivots will more likely lead to low growth in round-off errors. The scaling factors are returned as integral powers of 16.

The method is described in A.R. Curtis and J.K. Reid, AERE - TP.444, 1971.

Versions: MC10A; MC10AD.
Language: FORTRAN, Date: June 1971, Size: 3.7K; 151 cards.
Origin: A.R. Curtis and J.K. Reid, Harwell.

MC11A

To carry out a rank one update to a given positive definite or semi-definite symmetric matrix which is stored in a factorized form $A = LDL^T$, i.e. given a rank one matrix $\sigma\underline{w}\underline{w}^T$ forms $\tilde{A} = A + \sigma\underline{w}\underline{w}^T$.

The routine was written to be used by optimization routines and will also (i) accumulate a sum of rank one updates, (ii) carry out projection and allied operations on A which reduce the rank, and (iii) update rank deficient matrices where it is known from other considerations that the rank remains unchanged.

There are additional entry points which, factorize $A = LDL^T$, recover A from its factors, compute $A\underline{x}$ or $A^{-1}\underline{x}$, and obtain A^{-1} in factored form.

The method is described in M.J.D. Powell and R. Fletcher, Harwell Theoretical Physics report TP.519.

Versions: MC11A; MC11AD.
Language: FORTRAN, Date: Jan. 1973, Size: 4.9K; 275 cards.
Origin: R. Fletcher, Harwell.

MC12A

Calculates scaling factors for the rows and columns of an n by n real sparse matrix.

If the scaling is applied before Gaussian elimination with pivoting the choice of pivots will more likely lead to low growth of round-off errors. The factors returned are integral powers of 16.

The method is described in A.R. Curtis and J.K. Reid, AERE - TP.444, 1971.

The matrix is stored in the condensed form used by MA18A.

Versions: MC12A; MC12AD.
Language: FORTRAN, Date: June 1971, Size: 3.4K; 153 cards.
Origin: A.R. Curtis, Harwell.

MC13A

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 $U = PAP^{-1}$ is block upper triangular.

The method is that of R.W.H. Sargent and A.W. Westerberg, Trans. Inst. Chem. Engineers (1964), 42, 190-197.

Versions: MC13A; MC13AD; MC13C; MC13CD; the A versions use INTEGER*2s and the C versions INTEGER*4s.
Language: FORTRAN, Date: Feb. 1973, Size: 2.2K; 218 cards.
Origin: I. Duff, Oxford Univ., J.K. Reid, Harwell.

MC14A

Transforms an n by n real matrix A to an upper Hessenberg matrix having the same eigenvalues as A.

Transformations of the form $I - \alpha \underline{u}\underline{u}^T$ are used, see J.H. Wilkinson and C. Reinsch, 'Handbook for Automatic Computation - Linear Algebra', Springer-Verlag.

There is an option for specifying A as a submatrix of a larger matrix.

Versions: MC14A; MC14AD.
Calls: MC03AS.
Language: FORTRAN, Date: June 1973, Size: 1.6K; 648 cards.
Origin: ORTHES from the above reference, modified for Harwell by S. Marlow.

MC15A

To scale the rows and columns of a given real matrix A so that its elements are suitably balanced for accurate computation of its eigenvalues and eigenvectors.

Diagonal similarity transformations are used based on powers of 16 so that on a hexadecimal based computer only the exponent fields of the matrix elements are modified. Searches along rows and columns for isolated eigenvalues are also made and the matrix permuted so as to group such eigenvalues together. For the method see J.H. Wilkinson and C. Reinsch, 'Handbook for Automatic Computation - Linear Algebra', Springer-Verlag.

Versions: MC15A; MC15AD.

Language: FORTRAN, Date: June 1973, Size: 2.1K; 118 cards.

Origin: BALANCE from the above reference, modified for Harwell by S. Marlow.

MD03A

To evaluate the determinant of an n by n real symmetric matrix.

Gaussian elimination is used with a diagonal pivotal strategy which preserves symmetry (and can be unstable unless the matrix is positive definite).

Remark: Superseded by MA22A.

Versions: MD03A; MD03AD.

Language: FORTRAN, Date: Feb. 1964, Size: 1.8K; 99 cards.

Origin: M.J.D. Powell, Harwell.

ME01B

Transforms an n by n Hermitian matrix into a tri-diagonal matrix having the the same eigenvalues. Householder's method is used, see, J.H. Wilkinson, Numerische Mathematik, Dec. 1962.

The FORTRAN COMPLEX facility is not used.

Versions: ME01B; ME01BD.

Language: FORTRAN, Date: 1966, Size: 3.4K; 78 cards.

Origin: Harwell.

ME02A

Given an Hermitian matrix H and a complex vector $\underline{z}$, computes the complex product $\underline{x} = H\underline{z}$.

The FORTRAN COMPLEX facility is not used.

Versions: ME02A; ME02AD.

Language: FORTRAN, Date: July 1963, Size: 1K; 20 cards.

Origin: E.J. York*, Harwell.

ME04A

To solve a system of n complex linear algebraic equations in n unknowns.

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

Gaussian elimination is used. The FORTRAN COMPLEX facility is not used.

Remark: Superseded by MA23A.

Versions: ME04A; ME04AD; ME04B; ME04BD; B versions solve system which is complex upper Hessenberg.

Calls: MC03AS.

Language: FORTRAN, Date: July 1966, Size: 2.8K; 108 cards.

Origin: Harwell.

MEO6AS

To evaluate the sum of a complex constant c and an inner product of two complex vectors $\underline{a} = \{a_i\}$ and $\underline{b} = \{b_i\}$ accumulating intermediate results double length, i.e. evaluate

$$s = \pm c \pm \sum_{i=1}^{n} a_i b_i$$

the signs being specified.

The elements of the vectors $\underline{a}$ and $\underline{b}$ may be stored in any regular fashion. The first two elements of each vector are given and the displacement of the second from the first and the length n are used to locate the remainder. There is an option which allows the user to specify the vector $\underline{b}$ to be the complex conjugate of the actual vector passed to the routine. The constant and the elements of the vectors must be FORTRAN COMPLEX variables.

Versions: MEO6AS; MEO6AD; s may optionally be returned to full double precision when using the single precision version MEO6AS.
Language: 360/BAL, Date: June 1971, Size: .3K; 121 cards.
Origin: S. Marlow, Harwell.

MEO7A

Calculates scaling factors for the rows and columns of an n by n complex matrix.

If the scaling is applied before Gaussian elimination with pivoting the choice of pivots will more likely lead to low growth in round-off errors. The scaling factors are returned as integral powers of 16.

See, A.R. Curtis and J.K. Reid, AERE - TP.444, 1971.

Versions: MEO7A; MEO7AD.
Language: FORTRAN, Date: Aug. 1971, Size: 3.9K; 151 cards.
Origin: S. Marlow, Harwell.

MEO8A

Transforms an n by n complex Hermitian matrix $A = \{a_{ij}\}$ $a_{ij} = \bar{a}_{ji}$, to a tri-diagonal Hermitian matrix having the same eigenvalues as A.

Complex Householder orthogonal transformations of the form $(I - \theta \underline{\bar{u}}\underline{u}^T)$ are applied, see Wilkinson, J.H. Numerische Mathematik, December 1962.

Versions: MEO8A; MEO8AD.
Calls: MEO6AS.
Language: FORTRAN, Date: September 1971; Size: 3.2K; 77 cards.
Origin: S. Marlow, Harwell.

MX01A

To extend the MAX and MIN facilities of FORTRAN. They find the position of the algebraic minimum or maximum element in a vector of numbers.

The vector may be stored in any regular fashion, the user specifies the first and second elements and the number of elements in the vector.

The most common application is to find the minimum or maximum element in a simple FORTRAN array.

Versions: MX01A; MX01AD; MX01I integers.
Language: 360/BAL, Date: Dec. 1969, Size: .1K; 45 cards.
Origin: M.J. Hopper, Harwell.

MX02A

To extend the MAX and MIN facilities of FORTRAN. They find the position of the minimum or maximum elements of absolute value in a vector of numbers.

The vector may be stored in any regular fashion, the user specifies the first and second elements and the number of elements in the vector.

The most common application is to find the minimum or maximum element of absolute value in a simple FORTRAN array.

Versions: MX02A; MX02AD; MX02I integers.
Language: 360/BAL, Date: Dec. 1969, Size: .1K; 49 cards.
Origin: M.J. Hopper, Harwell.

N. Non-linear Equations

NB01A

To find a real zero of a continuous function f(x) of one variable in a given interval $a \leq x \leq b$, i.e. solve the non-linear equation in one variable

$$f(x) = 0 \qquad a \leq x \leq b$$

The user must supply code to evaluate f(x) at any point in the interval.

A binary search of the interval is made to find an interval $[\alpha,\beta]$ such that the signs of $f(\alpha)$ and $f(\beta)$ differ. The zero is then located by a combination of linear interpolation and binary subdivision.

The user can limit the number of function evaluations and must specify an ε so that any point ξ such that $|f(\xi)| < \varepsilon$ is considered a zero of f(x).

Versions: NB01A; NB01AD.
Language: FORTRAN, Date: Feb. 1972, Size: 1.2K; 110 cards.
Origin: M.J.D. Powell and S. Marlow, Harwell.

NS01A

To solve a system of n non-linear equations in n unknowns, i.e. solve

$$f_i(x_1,x_2,\dots,x_n) \qquad i=1,2,\dots,n$$

The user must supply initial estimates of $x_1,x_2,\dots,x_n$ and a subroutine to compute values of the functions f_i $i=1,2,\dots,n$.

The method uses the ideas of Newton-Raphson and Steepest descent coupled with Broyden's method for improving Jacobi matrices, see, M.J.D. Powell, AERE - R.5947.

Derivatives are not required but the user must indicate a reasonable step size to be used for approximating derivatives by finite differences. A solution to the system is accepted when $\sum_{i=1}^{n} f_i^2 \leqslant \varepsilon$, ε specified by the user.

Versions: NS01A; NS01AD.

Calls: MB01B and CALFUN (a user routine).

Language: FORTRAN, Date: May 1968, Size: 6.6K; 360 cards.

Origin: M.J.D. Powell, Harwell.

NS03A

To solve a system of m nonlinear equations in n unknowns, $\underline{x} = x_1,x_2,\dots,x_n$, of the form

$$r_i(\underline{x}) = f_i(\underline{x}) + \sum_{j=1}^{n} a_{ij}x_j = 0 \qquad i=1,2,\dots,m \qquad m \geqslant n$$

where the matrices $\{a_{ij}\}$ and $\{\partial f_i/\partial x_j\}$ are sparse.

The overdetermined case, $m > n$, is handled by taking the solution to be that which minimizes the sum of squares $s = \sum_{i=1}^{m} [r_i(\underline{x})]^2$. This makes the routine suitable for the nonlinear data fitting problem and parameter variances and covariances can be generated. Derivatives are optional and when not provided by the user are estimated using a definite difference approximation.

The algorithm is based on Fletcher's version of the Marquardt method, see R. Fletcher, AERE Report - R.6799, and is described in J.K. Reid, AERE Report R.7293.

The user must supply $\{a_{ij}\}$ in a condensed form, an initial estimate of $\underline{x}$, and a subroutine to calculate $f_i(\underline{x})$ and optionally $\partial f_i/\partial x_j$.

Versions: NS03A; NS03AD.

Calls: KB10AS, MA17A, MC02AS, MC09A and TD02A.

Language: FORTRAN, Date: October 1972, Size: 12.3K; 473 cards.

Origin: J.K. Reid, Harwell.

O. Input and Output

OA01A

To print out an m by n matrix in an easily read format. The FORTRAN format used to print the elements is fixed. There is no limit to the size of matrix and the output will extend over several numbered pages if necessary.

Versions: OA01A; OA01AD.
Language: FORTRAN, Date: April 1963, Size: 1K; 45 [illegible]rds.
Origin: E.J. York*, Harwell.

OA02A

To print out the contents of a single dimensioned array in tabular form.

The array is printed as a series of subtables, column length given by the user, of numbers printed in an E FORMAT to a number of significant figures specified by the user. Each member of the array will be identified by an integer printed alongside its entry in the table.

Versions: OA02A; OA02AD.
Language: FORTRAN, Date: June 1968, Size: 1.1K; 63 cards.
Origin: M.J. Hopper, Harwell.

OA03A

To print out in tabular form the contents of two single dimensioned arrays.

The FORMAT types (F, E or D) and the number of significant figures to be printed must be separately specified for both arrays.

The arrays are tabulated as a series of subtables, column length specified by the user, with corresponding elements from each array forming columns of pairs of numbers. As many columns as possible are fitted across the page and as many subtables are produced as are required to exhaust the arrays.

It is useful for tabulating a function y(x) and its argument.

Versions: OA03A; OA03AD.
Calls: MX02A.
Language: FORTRAN, Date: Dec. 1970, Size: 2.9K; 208 cards.
Origin: M.J. Hopper, Harwell.

OB11A

To draw on the graph plotter a smooth, tangentially continuous curve through a given sequence of points $x_i y_i$ i=1,2,...,n where optionally, the directions at each point may be specified. Cusps and closed curves can be drawn if required.

The smooth curve is generated as a sequence of straight line segments. The method is described in S. Marlow and M.J.D. Powell, AERE Report - R.7092 and is based on a method due to D.J. McConalogue.

The user is expected to have initialized graph plotting through the GHOST system and set the scaling before calling the subroutine.

Versions: OB11A.

Calls: NB01A and SHVECS (a GHOST routine).

Language: FORTRAN, Date: March 1972, Size: 5.1K; 408 cards.

Origin: S. Marlow and M.J.D. Powell, Harwell.

OB12A

To draw on the graph plotter a cubic spline defined by its knots, function values and first derivative values at the knots.

The spline is approximated by a sequence of straight line segments to within an accuracy under the control of the user. The method is described in S. Marlow and M.J.D. Powell, AERE Report R.7470.

The user is expected to have initialized graph plotting through the GHOST system and set the scaling before calling the routine.

Remark: The routine is dependent on the local graph plotting system only through one routine and that is one for drawing a sequence of line segments.

Versions: OB12A.

Calls: SHVECS (a GHOST routine).

Language: FORTRAN, Date: October 1972, Size: 2.9K; 125 cards.

Origin: S. Marlow and M.J.D. Powell, Harwell.

OC01A

To produce a graph of several dependent variables against an independent variable on the line printer or teletype.

The dependent variables are identified on the graph by symbolic names chosen by the user. The graph is orientated so that the independent variable axis runs down the page thus allowing graphs of arbitrary length to be produced. A graph may be put out in sections and different graphs can be put on different output streams. The user must supply upper and lower bounds on the dependent variables for scaling purposes.

Versions: OC01A; no double precision version.

Language: FORTRAN, Date: Dec. 1969, Size: 3.7K; 170 cards.

Origin: A.R. Curtis, Harwell.

OD01A

To allow the FORTRAN programmer to read or write paper tape. This is a two stage process; in stage one of reading a paper tape the tape is first read by one of the 360's peripheral computers and the data is stored on disk; then stage two the user calls OD01A in a program run on the 360 to read the data from disk. To write a paper tape the process is reversed. See, Chapter 13.5 in the Harwell 'Computer User's Manual'.

Language: FORTRAN, Date: 1967, Size: 7.4K; 384 cards.

Origin: D. McVicar, Harwell.

ODO2A

To allow the FORTRAN programmer to read or write Dectape. This is a two stage process similar to that for reading and writing paper tape, see ODO1A.

Language: FORTRAN, Date: 1967, Size: 9.3K; 617 cards.
Origin: R. Garside*, Harwell.

OEO1A

Provides simple editing facilities for the modification of source language and data which may be stored on any sequentially organised backing store or presented as a card deck.

The editing is performed according to commands given by the user on control cards. There are facilities for replacing, inserting and deleting statements, and changing specified fields in a statement. The new version of the source statements can optionally be re-sequence numbered. The new version, along with changes made, is listed.

The user controls the destination of the output from the routine by supplying an appropriate DD card and by specifiing the stream number.

Calls: INTRAN and OUTRAN (two user routines).
Language: FORTRAN, Date: July 1971, Size: 5.1K; 251 cards.
Origin: A.R. Curtis, Harwell.

ONO1B

To provide a 'free format' input facility for the FORTRAN user.

The subroutine is designed to read data items as opposed to records (i.e. cards) although it does provide facilities for monitoring record type input.

It offers facilities to, read numeric data, identify keywords held in a dictionary, build and add to the dictionary, recognize bad input data and allow the user to take corrective action, recognise and identify special characters, list each input record as it is read in, provide and allow the user to reset a pointer within the record to the current item, read BCD information.

Language: FORTRAN, Date: 1967, Size: 10.9K; 558 cards.
Origin: M. Ruffle*, Harwell.

ONO2A

Given a data card containing numeric data items constructs the FORTRAN format required to read the numbers from the card.

The format is returned in an array for the user to use in a READ statement. The subroutine was designed for use with the Harwell teletype system when it is not always convenient to type in data in a fixed format.

Calls: ZAO3AS.
Language: FORTRAN, Date: 1968, Size: 1.6K; 110 cards.
Origin: R.C.F. McLatchie, Harwell.

P. Polynomials and Rational Functions

PA01A

To find all the real and complex roots of a polynomial with real coefficients, i.e. calculate the zeros of

$$a_0+a_1x+a_2x^2+\ldots+a_nx^n = 0$$

The method is that of D.E. Muller, MTAC, 1956.

Versions: PA01A; PA01AD.

Language: FORTRAN, Date: March 1963, Size: 2.5K; 155 cards.

Origin: SHARE program C2* MLF HPRS, translated for Harwell by D. McVicar.

PA02A

To find all the real roots of a polynomial with real coefficients that fall within a given interval $a \leq x \leq b$, i.e. calculate the zeros of

$$a_0+a_1x+a_2x^2+\ldots+a_nx^n = 0 \qquad a \leq x \leq b \qquad n \leq 49$$

The subroutine will also return the number of real roots above, below and within the interval and there is an option to request all the real roots.

Sturm's sequence polynomials are generated to bracket the roots then a combination of Newton Raphson and bisection is used to refine each root.

Versions: PA02A. PA02AD; PA02B; PA02BD; the B versions provide an estimate of error growth induced by using a recurrence relation to generate the Sturm's sequence.

Language: FORTRAN, Date: Sept. 1965, Size: 3.9K; 234 cards.

Origin: M.J. Hopper, Harwell.

PA03A

To find all the roots of a cubic polynomial, i.e. calculate the zeros of

$$a_0+a_1x+a_2x^2+a_3x^3 = 0$$

A non-iterative method is used.

Versions: PA03A; PA03AD.

Language: FORTRAN, Date: Aug. 1963, Size: 1.5K; 49 cards.

Origin: H. Simmonds*, Harwell.

PA04A

To find the number of real roots above, below and within a specified interval $a \leq x \leq b$.

Sturm's sequence polynomials are used. The degree of the polynomial must be less than 21.

Versions: PA04A; PA04AD.

Language: FORTRAN, Date: Aug. 1963, Size: 3.1K; 63 cards.

Origin: M. Lancefield*, Harwell.

PA05A

To find all the roots of a quartic polynomial, i.e. the zeros of

$$a_0+a_1x+a_2x^2+a_3x^3+a_4x^4 = 0$$

The method is non-iterative, see, Maths. of Computation, 279-281, July 1960.

Remark: An iterative method is likely to produce more accurate results, try PA02B.

Versions: PA05A; PA05AD.

Calls: PA03A.

Language: FORTRAN, Date: Jan. 1963, Size: 1.8K; 96 cards.

Origin: S. Marlow, Harwell.

PB01AS

To compute the value of a polynomial

$$P(x) = a_0+a_1x+a_2x^2+\ldots+a_nx^n$$

The value is calculated using the nested form

$$P(x) = (\ldots((a_nx+a_{n-1})x+\ldots+a_1)x+a_0$$

accumulating intermediate results double length.

Versions: PB01AS; PB01AD.

Language: 360/BAL, Date: Aug. 1967, Size: .1K; 34 cards.

Origin: R.C.F. McLatchie, Harwell.

PB02AS

To compute the complex value of the real polynomial

$$P(z) = a_0+a_1z+a_2z^2+\ldots+a_nz^n$$

for complex argument z.

The method is synthetic division by a quadratic factor, see, R. Butler and E. Kerr, 'Introduction to Numerical Methods', Pitman.

P(z) and z must be FORTRAN COMPLEX variables; the calculation is carried out in real arithmetic accumulating intermediate results double length.

Versions: PB02AS; PB02AD.

Language: 360/BAL, Date: July 1967, Size: .3K; 124 cards.

Origin: M.J. Hopper, Harwell.

PC01A

To calculate the coefficients of a polynomial given all its roots, the roots must be real, i.e. given real numbers $\xi_1,\xi_2,\ldots,\xi_n$ calculate $a_0,a_1,\ldots,a_n$ with $a_n = 1$ such that

$$a_0+a_1x+a_2x^2+\ldots+a_nx^n \equiv (x-\xi_1)(x-\xi_2)\ldots.(x-\xi_n)$$

Versions: PC01A; PC01AD.

Language: FORTRAN, Date: Aug. 1967, Size: .5K; 45 cards.

Origin: W.E. Hart*, Harwell.

PC02A

Given n+1 points x_i y_i i=0,1,2,...,n calculates the coefficients of the polynomial that passes through all n+1 points, i.e. the interpolation polynomial

$$P(x) = a_0+a_1x+a_2x^2+\ldots+a_nx^n$$

such that $P(x_i) = y_i \quad i=0,1,2,\ldots,n$

The coefficients of the Lesbegue polynomials $L_K(x)$, i.e. such that $L_K(x_i) = 0$ $i \neq K$ and $L_K(x_K) = 1$, are calculated using PC01AS and PB01AS, the coefficients $a_0,a_1,\ldots,a_n$ are then obtained from

$$P(x) \equiv \sum_{K=0}^{n} y_K L_K(x)$$

Versions: PC02A; PC02AD.

Calls: PB01AS and PC01A.

Language: FORTRAN, Date: May 1964, Size: .7K; 23 cards.

Origin: L. Morgan*, Harwell.

PD01AS

Given a real root of a polynomial divides out the root to obtain the coefficients of the reduced polynomial, i.e. given a root ξ and polynomial $a_0+a_1x+\ldots+a_nx^n$ calculates b_j j=0,1,...,n-1 such that

$$a_0+a_1x+\ldots+a_nx^n \equiv (x-\xi)(b_0+b_1x+\ldots+b_{n-1}x^{n-1})+r$$

where [illegible] assumed to be zero.

Synthetic division of a polynomial by a linear factor is used accumulating intermediate results double length. The root is divided out from the end least likely to produce error growth in the b_js. The user has an option to force the division to be carried out from either end.

Versions: PD01AS; PD01AD.

Language: 360/BAL, Date: May 1967, Size: .2K; 80 cards.

Origin: Harwell.

PD02A

To find the first m terms of the Taylor series expansion of functions of a given polynomial (or polynomials).

Let the polynomials be $A(x) = a_0+a_1x+\dots+a_nx^n$ and $B(x) = b_0+b_1x+\dots+b_Kx^K$ and suppose the Taylor series to be $T(x) = t_0+t_1x+\dots+t_mx^m+\dots$ the expansions are obtained by considering identities between A(x), B(x) and T(x). The functions covered and identities used are

PD02A:	$T(x) = 1/A(x)$;	$A(x)T(x) \equiv 1$
PD02B:	$T(x) = \log_e[A(x)]$;	$T'(x)A(x) \equiv A'(x)$, $T(0)=\log_e(a_0)$
PD02C:	$T(x) = \exp[A(x)]$;	$T'(x) \equiv A'(x)T(x)$, $T(0)=\exp(a_0)$
PD02D:	$(T_s(x)=\sin[A(x)]$; $(T_c(x)=\cos[A(x)]$;	$T_s'(x) \equiv A'(x)T_c(x)$, $T_s(0)=\sin(a_0)$; $T_c'(x) \equiv A'(x)T_s(x)$, $T_c(0)=\cos(a_0)$
PD02E:	$T(x) = [A(x)]^r$;	$A(x)T'(x) \equiv rA'(x)T(x)$, $T(0) = a_0r$
PD02F:	$T(x) = A(x)B(x)$;	$T(x) \equiv A(x)B(x)$
PD02G:	$T(x) = A(x)/B(x)$;	$T(x)B(x) \equiv A(x)$

Versions: PD02A; PD02AD; PD02B; PD02BD; PD02C; PD02CD; PD02D; PD02DD; PD02E; PD02ED; PD02F; PD02FD; PD02G; PD02GD.
Calls: MC02BS.
Language: FORTRAN, Date: Dec. 1970, Size: .7K; 43 cards.
Origin: M.J. Hopper, Harwell.

PD03A

Given a polynomial in x, i.e.

$$P(x) = a_0+a_1x+\dots+a_nx^n \qquad n \leqslant 50$$

calculates the coefficients b_j $j=0,1,\dots,n$ of the polynomial under a change of variable $z=ux+v$, i.e. such that

$$a_0+a_1x+\dots+a_nx^n \equiv b_0+b_1(ux+v)+\dots+b_n(ux+v)^n$$

Versions: PD03A; PD03AD.
Calls: PB01AS.
Language: FORTRAN, Date: June 1966, Size: .8K; 29 cards.
Origin: A.R. Curtis, Harwell.

PE01A

Calculates the coefficients of a polynomial given the coefficients of its expansion in terms of Chebyshev polynomials, i.e. given b_j $j=0,1,\dots,n$ find a_j $j=0,1,\dots,n$ such that

$$a_0+a_1x+\dots+a_nx^n \equiv b_0T_0(x)+b_1T_1(x)+\dots+b_nT_n(x) \qquad n \leqslant 40$$

If the polynomial is known to be an odd or even function the calculation can be simplified and the routine can take advantage of this.

Versions: PE01A; PE01AD.

Calls: MC03AS.

Language: FORTRAN, Date: June 1963, Size: 3K; 65 cards.

Origin: S. Marlow, Harwell.

PE03A

Given the coefficients of a polynomial calculates the coefficients of its expansion in terms of Chebyshev polynomials, i.e. given a_j $j=0,1,\ldots,n$ find b_j $j=0,1,\ldots,n$ such that

$$b_0T_0(x)+b_1T_1(x)+\ldots+b_nT_n(x) \equiv a_0+a_1x+\ldots+a_nx^n \qquad n \leq 40$$

Versions: PE03A; PE03AD.

Language: FORTRAN, Date: Sept. 1963, Size: .8K; 34 cards.

Origin: C. Brittian*, Harwell.

PE04A

Given a function f(x) in the range $u \leq x \leq v$ finds a rational approximation of the form

$$R_{m,n}(z) = \frac{a_0T_0(z)+a_1T_1(z)+\ldots+a_mT_m(z)}{T_0(z)+b_1T_1(z)+\ldots+b_nT_n(z)}$$

where $T_K(z)$ *is the Chebyshev polynomial of order K and* $-1 \leq z = \frac{2x-u-v}{v-u} \leq 1$ and $0 \leq m \leq 10$, $0 \leq n \leq 10$.

The approximation found is such that $R_{m,n}(z_K) = f[\frac{1}{2}(z_K(v-u)+v+u)]$ $K=0,1,\ldots,m+n$ where the z_K are the zeros of $T_{m+n+1}(z)$.

The routine returns a guide to how close the approximation is to the 'best approximation' in the minimax sense. Also returned are the coefficients of the equivalent form

$$\hat{R}_{m,n}(x) = \frac{\alpha_0+\alpha_1x+\ldots\alpha_nx^n}{\beta_0+\beta_1x+\ldots\beta_nx^n}$$

The user must provide a function subroutine to evaluate f(x).

Versions: PE04A; PE04AD.

Calls: MA21A and FUNCT (a user routine).

Language: FORTRAN, Date: Aug. 1963, Size: 7.7K; 300 cards.

Origin: A.T. Sinden*, Harwell.

PE05AD

Given a smooth function f(x) in the range $u \leq x \leq v$ finds a weighted rational approximation of the form

$$R_{m,n}(x) = \frac{a_0+a_1x+\ldots+a_mx^m}{b_0+b_1x+\ldots+b_nx^n}$$

which is 'best' in the sense that

$$\max_{u \leq x \leq v} \left| \frac{f(x) - R_{m,n}(x)}{W(x)} \right|$$

is minimized, W(x) is a positive weight function. The cases n=0 or m=0 are handled but there is a restriction $0 \leq m \leq 10$, $0 \leq n \leq 10$ and $m+n \leq 15$.

The method is iterative and uses the two stage Maehly algorithm, see, H.J. Maehly, J. Ass. Comp. Mach., 1963, with third order inverse iteration to solve the eigenvalue problem, see, A.R. Curtis and Osborne, Computer Journal, Vol. 9, no. 3, 1966.

The user must supply code to evaluate f(x) and W(x).

Versions: PE05AD; there is no single precision version.
Calls: MA01BD, MC01AD, MC03AD, PB01AD, PD03AD and ZE01AD.
Language: FORTRAN, Date: 1967, Size: 32.1K; 630 cards.
Origin: A.R. Curtis, Harwell.

PE06A

Given a polynomial C(x) of degree K finds the coefficients a_j j=0,1,...,m and b_j j=1,2,...,n of the Padé approximation

$$P_{m,n}(x) = \frac{a_0+a_1x+\ldots+a_mx^m}{1+b_1x+\ldots+b_nx^n} \qquad m+n \leq K \qquad m,n \leq 20$$

such that the first m+n+1 coefficients of the Taylor series expansion of $P_{m,n}(x)$ are the same as the first m+n+1 coefficients of c(x).

The m zeros, the n poles and residues of the approximation are also returned.

Versions: PE06A; PE06AD.
Calls: MA21A and PA01A.
Language: FORTRAN, Date: Jan. 1964, Size: 3.5K. 86 cards.
Origin: L. Morgan*, Harwell.

PE07A

To compute the value of a polynomial P(x) of degree n which is expressed as a linear combination of orthogonal polynomials $Q_K(x)$ K=0,1,...,n, i.e.

$$P(x) = c_0Q_0(x)+c_1Q_1(x)+\ldots+c_nQ_n(x)$$

where the polynomials $Q_K(x)$ are defined by the recurrence relation

$$Q_0(x) = 1, \qquad Q_1(x) = x-\alpha_0$$

$$Q_{K+1}(x) = (x-\alpha_K)Q_K(x) - \beta_KQ_{K-1}(x) \qquad K=1,2\ldots.$$

See F.J. Smith, 'Maths. of Computation', Jan. 1965.

Remark: Can be used to compute values of polynomial fits obtained by VC01A.

Versions: PE07A; PE07AD.

Language: FORTRAN, Date: July 1964, Size: .5K; 20 cards.

Origin: S. Northcliffe*, Harwell.

PE08A

To obtain the coefficients a_j $j=0,1,\ldots,n$ of a polynomial $P(x)$ of degree n which is expressed as a linear combination of orthogonal polynomials $Q_K(x)$ $K=0,1,\ldots,n$, i.e.

$$a_0+a_1x+\ldots+a_nx^n \equiv c_0Q_0(x)+c_1Q_2(x)+\ldots+c_nQ_n(x)$$

where the polynomials $Q_K(x)$ are defined by the recurrence relation

$$Q_0(x) = 1, \qquad Q_1(x) = x-\alpha_0$$

$$Q_{K+1}(x) = (x-\alpha_k)Q_K(x)-\beta_KQ_{K-1}(x) \qquad K=1,2,\ldots$$

Remark: Can be used to obtain the coefficients of polynomial fits produced by VC01A.

Versions: PE08A; PE08AD.

Language: FORTRAN, Date: April 1964, Size: 1.8K; 45 cards.

Origin: S. Northcliffe*, Harwell.

PE11A

Given m values of a function $f(x)$ calculates the nth degree minimax polynomial approximation $P(x)$, $m \geq n$, $2 \leq n \leq 25$, such that

$$\max_{1 \leq K \leq m} \left| f(x_K) - P(x_K) \right|$$

is minimized.

See P.C. Curtiss and W.L. Frank, Journal, A.C.M., 1959.

Versions: PE11A; PE11AD.

Language: FORTRAN, Date: Nov. 1966, Size: 5.7K; 338 cards.

Origin: S. Marlow, Harwell.

Q. Numerical Integration

QA01AS

To evaluate the integral $\int_a^b f(x)dx$ using one of five Newton-Cotes formulae, i.e. Trapezoidal rule, Simpson's rule, the ⅜ rule and the five and six point formulae.

The user must supply values of the integrand $f(x)$ tabulated at equal intervals in $a \leq x \leq b$, hence the interval size must be chosen by the user.

Double length accumulation of intermediate results is carried out to minimize rounding errors.

Remark: One of the adaptive quadrature routines, QA02A, QA04A or QA05A may give better results than QA01AS.

Versions: QA01AS; QA01AD.

Language: 360/BAL, Date: Aug. 1967, Size: .5K; 185 cards.

Origin: M.J. Hopper, Harwell.

QA02A

To evaluate the integral $\int_a^b f(x)dx$ to a specified absolute accuracy.

The subroutine uses a variable step Simpson's rule using at each step an integration step based on 4th differences which is chosen small enough to achieve the required accuracy.

The user must specify a minimum integration step-size and provide a subroutine to evaluate the integrand f(x).

Remark: Unless only limited accuracy is required, or core store is limited, either QA04A or QA05A may be better, especially on 'difficult integrands'.

Versions: QA02A; QA02AD. Both use double precision arithmetic.

Calls: CALCIN (a user routine).

Language: FORTRAN, Date: March 1963, Size: 3.3K; 175 cards.

Origin: M.J.D. Powell and A.R. Curtis, Harwell.

QA03A

To tabulate an integral function of the form

$$g(x) = g(a) + \int_a^x f(t)dt$$

at points x=a, a+h, a+2h,...,a+mh to a specified accuracy.

The subroutine uses a variable step Simpson's rule using at each step an integration step, based on 4th differences, which is chosen small enough to achieve the required accuracy.

The user must specify a minimum integration step-size and provide a subroutine to evaluate the integrand f(x).

Versions: QA03A; QA03AD.

Calls: CALCIN (a user routine).

Language: FORTRAN, Date: March 1963, Size: 2.6K; 138 cards.

Origin: M.J.D. Powell, Harwell.

QA04A

To evaluate the integral $\int_a^b f(x)dx$ to a specified relative or absolute accuracy.

An adaptive scheme is used based on a three point Gaussian quadrature, see I.G.A. Robinson, 'Adaptive Gaussian Integration', Australian Comp. Journal, Vol. 3, no. 3. The user must provide a FUNCTION subprogram to calculate values of the integrand f(x).

Remark: Shorter than QA05A but less efficient.

Versions: QA04A; QA04AD. Both use double precision arithmetic.

Language: FORTRAN, Date: May 1972, Size: 5.7K; 186 cards.

Origin: I.G.A. Robinson, Univ. of Melbourne, implement for Harwell by A.B. Smith*.

QA05A

To evaluate the integral $\int_a^b f(x)dx$ to a specified accuracy which may be relative or absolute.

The subroutine uses an adaptive scheme based on Romberg extrapolation and Trapezoidal rule, and is described under the name CADRE in C. deBoor, 'CADRE: an algorithm for numerical quadrature' in 'Mathematical Software' Ed. J.R. Rice, Academic Press.

The user must supply a subroutine to evaluate f(x) $a \leqslant x \leqslant b$. The subroutine returns error estimates and attempts to identify any singularities and discontinuities in f(x).

Remark: Powerful and efficient, but rather large in core requirement.

Versions: QA05A; QA05AD. Both use double precision arithmetic.

Language: FORTRAN, Date: October 1972, Size: 26K; 353 cards.

Origin: Subroutine CADRE, C. de Boor, Purdue Univ.; modified for Harwell by A.R. Curtis.

QB01A

To evaluate the multi-dimensioned integral

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n} f(x_1,x_2,\ldots,x_n)dx_1,dx_2,\ldots,dx_n \qquad 1 \leqslant n \leqslant 6$$

where the limits a_K and b_K can be functions of $x_1,x_2,\ldots,x_{K-1}$ and a_1 and b_1 constants. The user must specify for each dimension which type of quadrature is to be used, Simpson's Rule, Gauss quadrature or a quadrature based on Chebyshev polynomials.

A relative accuracy can be requested and a subroutine to compute the values of the limits and integrand must be provided.

Versions: QB01A; QB01AD.

Calls: LIMITS (a user routine).

Language: FORTRAN, Date: March 1963, Size: 10.5K; 345 cards.

Origin: F.R. Hopgood*, Harwell.

QC02AD

To provide weights and zeros for Gaussian type quadratures of integrals of the form

$$\int_0^\infty e^{-x}\, f(x)\, dx$$

The weights and zeros are calculated to give exact results when f(x) is a polynomial of degree 2k-1 in $z = \frac{x}{a+x}$ where a is chosen by the user. A range of quadratures may be requested. The calculations are relatively expensive and the routine should only be used when the quadrature is to be applied to many integrals.

Remark: Use of this routine is likely to give better results than Gauss-Laguerre for bounded f(x).

Versions: QC02AD: there is no single precision version.

Calls: NB01AD and QA05AD.

Language: FORTRAN, Date: 1965, Size: 3.2K; 116 cards.

Origin: A.R. Curtis, Harwell.

QD01A

To evaluate integrals of the form

$$\int_a^b f(t) \sin xt \, dt \quad \text{and} \quad \int_a^b f(t) \cos xt \, dt$$

to a specified absolute accuracy.

Filon's method is used where a quadrature formula is derived by approximating to f(x) by a piece-wise quadratic interpolant.

The user must provide a subroutine to evaluate the function f(x).

Remark: The method is to be preferred to Simpson's Rule if $|x(b-a)| \gtrsim 10$.

Versions: QD01A; QD01AD.

Calls: F (a user routine).

Language: FORTRAN, Date: 1968, Size: 2.8K; 103 cards.

Origin: W.E. Hart*, Harwell.

QM01A

Estimate the value of the n dimensional integral

$$\int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_n}^{b_n} f(x_1,x_2,\ldots,x_n)\, dx_1,dx_2,\ldots,dx_n \qquad n \leq 20$$

to a specified accuracy by a Monte Carlo method.

The method is that of 'Weighted Uniform Sampling', see, M.J.D. Powell and J. Swann, AERE - TP.207, 1966.

The sampling method requires an approximation to f(x) which has the properties

$$\int_R g(\underline{x})d\underline{x} = 1 \qquad R = \text{region of integration}$$

and $$f(\underline{x}) \approx \alpha g(\underline{x}) \qquad \alpha \quad \text{a constant.}$$

The user must provide a subroutine to compute values of f(x) and g(x).

Versions: QM01A; QM01AD.
Calls: FA01AS and EVAL (a user routine).
Language: FORTRAN, Date: June 1966, Size: 1.4K; 43 cards.
Origin: D. Green*, Harwell.

S. Statistics

SA01A

To evaluate the cumulative chi-squared probability function, i.e. given a statistic x distributed as χ^2 with n degrees a freedom evaluate the probability of $x \geq X$

$$Q(X,n) = \frac{1}{2^{\frac{n}{2}}\,\Gamma(\frac{n}{2})} \int_X^{\infty} t^{\frac{n}{2}-1} e^{-\frac{t}{2}} dt$$

to an absolute accuracy, where $x \geq 0$ and n an integer ≥ 1.

The approximations used are; a convergent series based on the expansion for the integral $Q(x,n)$; an asymptotic series for the complement probability $P(x,n) = 1-Q(x,n)$; and for very large x and n an approximation based on a Gaussian distribution. Empirical boundaries $Q(x,n) \leq 10^{-\beta}$ and $Q(x,n) \geq 1 - 10^{-\beta}$ are used to define regions in which the probability, within the guaranteed accuracy, can be set to 0 or 1 and allow truncated forms of the series to be used giving savings in computer time.

Accuracies: SA01A $< 5\times10^{-5}$; SA01AD $< 5\times10^{-10}$.
Versions: SA01A; SA01AD.
Language: FORTRAN, Date: Aug. 1971, Size: 2.3K; 214 cards.
Origin: M.J. Hopper and J. Hedger, Harwell.

SA02A

To evaluate the one-sided cumulative distribution function of Student's t distribution with n degrees of freedom, i.e. evaluate

$$P(n,t) = \frac{1}{B(\frac{1}{2},\frac{n}{2})n^{\frac{1}{2}}} \int_{-\infty}^{t} \left\{1 + \frac{\theta^2}{n}\right\}^{-\frac{n+1}{2}} d\theta \qquad -\infty \leq t \leq \infty$$

A series expansion is used, let $\alpha = \tan^{-1} \frac{t}{n^{\frac{1}{2}}}$ then if n is even

$$P(n,t) = \tfrac{1}{2}+\tfrac{1}{2} \sin\alpha \left\{1+\tfrac{1}{2}\cos^2\alpha+\ldots+\frac{1.3.5\ldots(n-3)}{2.4.6\ldots(n-2)}\cos^{n-2}\alpha\right\}$$

and if n is odd

$$P(n,t) = \tfrac{1}{2}+\frac{\alpha}{\pi}+\frac{1}{\pi}\sin\alpha\left\{\cos\alpha+\tfrac{2}{3}\cos^3\alpha+\ldots+\frac{2.4\ldots(n-3)}{3.5\ldots(n-2)}\cos^{n-2}\alpha\right\}$$

Versions: SA02A; no double length version.

Language: FORTRAN, Date: Dec. 1970, Size: .8K; 29 cards.

Origin: D.G. Papworth, MRC, Harwell.

SA03A

To evaluate the complement of the cumulative distribution function of the variance ratio distribution with (n,m) degrees of freedom.

$$P(n,m,F) = \frac{n^{\frac{n}{2}}\, m^{\frac{m}{2}}}{B(\frac{n}{2},\frac{m}{2})} \int_F^\infty \frac{f^{\frac{n-2}{2}}}{(nf+m)^{\frac{n+m}{2}}}\, df \qquad 0 \leq F \leq \infty$$

Series expansions in sin α and cos α are used for the integral, where $\alpha = \tan^{-1}\sqrt{\frac{nF}{m}}$.

Versions: SA03A; no double length version.

Language: FORTRAN, Date: Dec. 1970, Size: 1.3K; 57 cards.

Origin: D.G. Papworth, MRC, Harwell.

SV01A

To extract from a minimization of a sum of squares by VA02A an approximation to the variance-covariance matrix at the minimum.

Given that the sum of squares of functions

$$F(x_1,x_2,\ldots,x_n) = \sum_{K=1}^{m} f_K^2\,(x_1,x_2,\ldots,x_n)$$

has been minimized by VA02A, calculates an approximation to the inverse matrix G^{-1} where $G = \{g_{ij}\}$ is given by

$$g_{ij} = \sum_{K=1}^{m} \frac{\partial f_K(\underline{x})}{\partial x_i} \cdot \frac{\partial f_K(\underline{x})}{\partial x_j} \qquad i=1,2,\ldots,n; \qquad j=1,2,\ldots,n$$

Finite difference approximations are used for the derivatives.

Versions: SV01A; SV01AD.

Language: FORTRAN, Date: Dec. 1964, Size: 1.6K; 57 cards.

Origin: M.J.D. Powell, Harwell.

T. Interpolation and Approximation

TA01A

To compute and print out a table of differences of a function when the function values are given at equal intervals.

The function values must be integer and passed to the routine in an array. The user specifies the highest order difference required up to a maximum of 14.

Versions: TA01A; TA01AD.

Language: FORTRAN, Date: Feb. 1963, Size: 2.1K; 81 cards.

Origin: M.J.D. Powell, Harwell.

TA02A

To evaluate the divided differences of a function f(x) tabulated at points $x_1 < x_2 < \ldots < x_n$ not necessarily equally spaced.

The values of the function and the points x_j must be passed to the routine in an array. The user specifies the order differences required and may direct the subroutine to return the difference table in various compact forms. The table is not printed out by the subroutine.

Versions: TA02A; TA02AD.
Language: FORTRAN, Date: Feb. 1965, Size: 1.5K; 63 cards.
Origin: D. Russell, Atlas Laboratory, Chilton, Berks.

TA03A

To evaluate the central differences of a function tabulated at equal intervals.

The function values must be passed to the subroutine in an array. The user specifies the order of the highest order differences required and may direct the routine to return the difference table in various compact forms. No printing is done.

Versions: TA03A; TA03AD.
Language: FORTRAN, Date: Feb. 1965, Size: 1.3K; 55 cards.
Origin: D. Russell, Atlas Laboratory, Chilton, Berks.

TB01A

To interpolate the value of an even function f(x), i.e. such that $f(-x) = f(x)$, given n+1 function values f_i at points x_i $i=1,2,\ldots,n+1$ not necessarily equally spaced.

A polynomial P(x) of degree 2n is constructed such that $P(x_i) = f_i$ and $P(-x_i) = f_i$ $i=1,2,\ldots,n+1$ and based on the Lagrange interpolation formula. The coefficients of P(x) are not computed.

Versions: TB01A; TB01AD.
Language: FORTRAN, Date: March 1963, Size: .7K; 29 cards.
Origin: A.G. Hearn*, Harwell.

TB02A

To interpolate the value of a function given n+1 function values f_i at points x_i $i=1,2,\ldots,n+1$ not necessarily equally spaced.

The interpolation is based on the nth degree polynomial which passes through the n+1 points obtained by the Langrange interpolation formula. The coefficients of the polynomial are not computed.

Versions: TB02A; TB02AD.
Language: FORTRAN, Date: May 1963, Size: .5K; 14 cards.
Origin: A.G. Hearn*, Harwell.

TB03A

Calculates the coefficients of the piece-wise cubic function which interpolates n+1 given function values f_i at points x_i $i=1,2,\dots,n+1$.

The interpolation function derived will have the following properties; it will be continuous, have continuous first derivative and if the f_is defined a quadratic polynomial would represent it exactly. The routine returns the coefficients of the n cubics $C_i(\theta)$ corresponding to the n intervals x_{i+1} to x_i in the transformed variable $\theta = \frac{x - x_i}{x_{i+1}-x_i}$, i.e.

$$C_i(\theta) = a_0+a_1\theta+a_2\theta^2+a_3\theta^3 \qquad 0 \leq \theta \leq 1$$

Versions: TB03A; TB03AD.
Language: FORTRAN, Date: July 1964, Size: 1.6K; 46 cards.
Origin: D. Miller*, Harwell.

TB04A

Given function values $f_1,f_2,\dots,f_n$ at points $x_1 < x_2 < \dots < x_n$, not necessarily equally spaced, finds a cubic spline S(x) that interpolates the n function values, i.e. $S(x_i) = f_i$ $i=1,2,\dots,n$ where S(x) has knots at the points x_i $i=1,2,\dots,n$.

The 3rd derivative at the points x_2 and x_{n-1} is forced to be continuous.

The spline is defined on return by the knots x_i, its values at the knots f_i and its first derivative values at the knots.

Versions: TB04A; TB04AD.
Language: FORTRAN, Date: Feb. 1970, Size: 1.6K; 56 cards.
Origin: J.K. Reid, Harwell.

TB05A

Given function values f_i $i=1,2,\dots,n$ at points $x_1 < x_2 < \dots < x_n$, not necessarily equally spaced, finds a periodic cubic spline S(x) that interpolates the n function values, i.e.

$$S(x_i) = f_i \qquad i=1,2,\dots,n \qquad f_n=f_1$$

where S(x) has knots x_i $i=1,2,\dots,n$.

The spline is defined on return by the knots x_i, its values at the knots and its first derivative values at the knots.

Versions: TB05A; TB05AD.
Language: FORTRAN, Date: July 1970, Size: 1.5K; 59 cards.
Origin: J.K. Reid, Harwell.

TD01A

To estimate first or second derivatives given a table of function values and finite differences. The two formulae

$$hf'_o = (\mu\,\delta_o - \frac{1}{6}\,\mu\,\delta_o^3 + \frac{1}{30}\,\mu\,\delta_o^5 - \ldots)\,f_o$$

$$h^2 f''_o = (\delta_o^2 - \frac{1}{12}\,\delta_o^4 + \frac{1}{90}\,\delta_o^6 - \ldots)\,f_o$$

are used and the user specifies the highest order of differences which can be used and the required accuracy. The routine will return a condition flag indicating whether the number of differences supplied was sufficient to achieve the accuracy. The routine does not use special formulae at the ends of the range and the user is expected to extend the table so that there are a sufficient number of differences for the end values.

Versions: TD01A; TD01AD.
Language: FORTRAN, Date: 1965, Size: 3K; 136 cards.
Origin: P. Hallowell, Atlas Laboratory, Chilton, Berks.

TD02A

Given functions $f_i(x_1,x_2,\ldots,x_n,t)$ $i=1,2,\ldots,m$ evaluates an approximation to the Jacobian matrix $J = \{\partial f_i/\partial x_j\}$ using finite differences. The routine is intended for the case when J is sparse or band structured and has additional entries which given the functions f_i construct the sparsity pattern for J.

Method references: A.R. Curtis, M.J.D. Powell and J.K. Reid, A.E.R.E. Report TP.476 and A.R. Curtis and J.K. Reid, A.E.R.E. Report TP.477.

Derivatives are estimated by

$$\partial f_i/\partial x_j = \frac{1}{2h_j}\,\{f_i(x_1,\ldots,x_j+h_j,\ldots,x_n,t) - f_i(x_1,\ldots,x_j-h_j,\ldots,x_n,t)\}$$

where the steplengths h_j are automatically chosen by the routine within bounds specified by the user.

Versions: TD02A; TD02AD.
Language: FORTRAN, Date: February 1972, Size: 4.2K; 172 cards.
Origin: J.K. Reid, Harwell.

TG01A

To compute the value of a cubic spline when the spline is defined over a given range a to b in the standard format.

The value outside the range is defined to be the value at the nearest limit point a or b.

Versions: TG01A; TG01AD.
Language: FORTRAN, Date: Nov. 1966, Size: 1K; 43 cards.
Origin: M.J.D. Powell, Harwell.

TG01B

To compute the value of a cubic spline given in terms of its knot points and its values and first derivative values at the knots.

The spline value is defined to be zero outside the range. A facility is provided for reducing the search time for the knot interval containing the point at which the spline value is required; this makes tabulation of a spline economical.

Versions: TG01B; TG01BD.
Language: FORTRAN, Date: 1966, Size: 1K; 83 cards.
Origin: M.J. Hopper, Harwell.

TS01A

Finds a cubic spline approximation to a given smooth function $f(x)$ within a prescribed accuracy ε, i.e. finds a cubic spline $S(x)$ such that

$$|S(x) - f(x)| < \varepsilon \qquad a \leqslant x \leqslant b$$

The method is given in M.J.D. Powell, AERE - R.7308.

The user must provide a subroutine to evaluate $f(x)$. The spline is returned in terms of the knots, the function values and first derivative values at the knots.

Remark: TG01B may be used to obtain values of the spline.
Versions: TS01A; TS01AD.
Language: FORTRAN, Date: Dec. 1972, Size: 4.8K; 278 cards.
Origin: M.J.D. Powell, Harwell.

V. Minimization and Nonlinear Data Fitting

VA02A

To minimize a sum of squares of m functions of n variables, i.e. find $\underline{x} = x_1, x_2, \ldots, x_n$ to minimise

$$F(\underline{x}) = \sum_{i=1}^{m} \{f_i(\underline{x})\}^2 \qquad m \geqslant n$$

Derivative values are not required.

The method is described in M.J.D. Powell, Computer Journal, Vol. 7, No. 4, 1965.

A common application is that of fitting a general function $G(t,\underline{x})$ of n variables $\underline{x} = x_1, x_2, \ldots, x_n$ to discrete data t_i y_i $i=1,2,\ldots m$, in the least squares sense.

An initial estimate of the solution and the accuracy required in the variables must be provided. The length of step taken at each iteration can be restricted.

Remark: Superseded by VA05A.

Calls: VD01A and CALFUN (a user routine).

Versions: VA02A; VA02AD.

Language: FORTRAN, Date: 1964, Size: 5.9K; 322 cards.

Origin: M.J.D. Powell, Harwell.

VA03A

Reduces the problem of finding the minimum of a function of n variables to that of finding a minimum of a function of one variable many times, i.e. find the minimum of $F(x_1,x_2,\dots,x_n)$ given that the user can always supply the minimum of

$$g(\lambda) = F(x_1+\lambda d_1,x_2+\lambda d_2,\dots,x_n+\lambda d_n)$$

for any values of x_i and d_i $i=1,2,\dots,n$ the subroutine provides.

The method used is the same as that used by VA04A, except that for VA03A the user has to calculate λ. Thus better accuracy and faster computation times can be obtained.

The user must give an initial estimate of the minimum position and specify the accuracy required in the final solution. A subroutine is required to calculate the minimum of $g(\lambda)$.

Versions: VA03A; VA03AD.

Calls: MINLIN (a user routine).

Language: FORTRAN, Date: July 1963, Size: 1.2K; 69 cards.

Origin: M.J.D. Powell, Harwell.

VA04A

To find the minimum of a general function of n variables, i.e. $F(x_1,x_2,\dots,x_n)$. Values of the derivatives are not required.

A variation of the simple method of changing one variable at a time is used. The method is such that when applied to a quadratic form, it causes conjugate directions of search to be chosen, so when applied to a general function the ultimate rate of convergence is fast, see, M.J.D. Powell, Computer Journal, Vol. 7, No. 2, 1964.

The method is iterative and requires the user to give an initial estimate of the minimum position. The required accuracies in the solution must be specified and a subroutine to calculate values of $F(x_1,x_2,\dots,x_n)$ provided.

Versions: VA04A; VA04AD.

Calls: CALCFX (a user routine).

Language: FORTRAN, Date: Feb. 1964, Size: 4.3K; 270 cards.

Origin: M.J.D. Powell, Harwell.

VA05A

To minimise a sum of squares of m functions in n variables, i.e. find $\underline{x} = x_1, x_2, \ldots, x_n$ to minimise

$$F(\underline{x}) = \sum_{i=1}^{m} \{f_i(\underline{x})\}^2 \qquad m \geq n$$

Values of derivatives are not required.

A hybrid method is used combining features from the Newton-Raphson, Steepest descent and Marquardt methods and calculating and maintaining an approximation to the first derivative matrix using the ideas of Broyden.

The subroutine can be applied to the problem of fitting a general function $G(t,\underline{x})$ of n variables $\underline{x} = x_1, x_2, \ldots, x_n$ to data points t_i y_i $i=1,2,\ldots,m$ where

$$F(\underline{x}) = \sum_{i=1}^{m} \{y_i - G(t_i, \underline{x})\}^2$$

would be the function to be minimised.

It is important that the variables are scaled so that their magnitudes are similar. An initial estimate of the solution is required and the user must specify the accuracy with which the minimum value of $F(\underline{x})$ is to be found. A subroutine to compute values of the m functions $f_i(\underline{x})$ must be provided.

Versions: VA05A; VA05AD.

Calls: MB11A and CALFUN (a user routine)

Language: FORTRAN, Date: May 1969, Size: 8.8K; 509 cards.

Origin: M.J.D. Powell, Harwell.

VA06A

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

The method is a hybrid one based on the steepest descent algorithm and on the generalised Newton iteration, see M.J.D. Powell, AERE - R.6469, 1970.

The user must provide initial estimates of the solution and provide a subroutine to calculate values of the function and its derivatives.

Versions: VA06A; VA06AD.

Calls: CALCFG (a user routine).

Language: FORTRAN, Date: June 1970, Size: 6.5K; 344 cards.

Origin: M.J.D. Powell, Harwell.

VA07A

To minimise a sum of squares of m functions of n variables when values of the derivatives of the functions with respect to the variables are available, i.e., find $\underline{x} = x_1, x_2, \ldots, x_n$ to minimise

$$F(\underline{x}) = \sum_{i=1}^{m} \{f_i(\underline{x})\}^2 \qquad m > n \qquad m < 200,\ n < 25$$

given $\dfrac{\partial f_i(\underline{x})}{\partial x_j}$ $j=1,2,\ldots,n;$ $i=1,2,\ldots,m.$

The subroutine uses a modified Marquardt method, see, R. Fletcher, AERE - R.6799.

The method is iterative and the user must supply an initial estimate of the final solution. It allows extra side conditions to be applied in a limited way.

A subroutine must be provided to evaluate the $f_i(\underline{x})$ and derivatives. There are three options for implicit scaling of the variables. The user must specify the accuracy required in the solution.

Typically the functions $f_i(\underline{x})$ might be residuals of a non-linear least squares data fitting problem.

Versions: VA07A; VA07AD.
Calls: MA10A and MC03AS.
Language: FORTRAN, Date: April 1971, Size: 5.8K; 163 cards.
Origin: R. Fletcher, Harwell.

VA08A

To find the minimum of a general function $f(\underline{x})$ of several variables $\underline{x} = x_1, x_2, \ldots, x_n$ given that values of the derivatives $\partial f/\partial x_i$ can be calculated. The subroutine should be used on large problems when storage space is at a premium.

The method of conjugate gradients is used, see R. Fletcher and C.M. Reeves, Computer Journal, Vol. 7, p.149.

Versions: VA08A; VA08AD.
Calls: MC03AS.
Language: FORTRAN, Date: January 1972, Size: 2.1K; 84 cards.
Origin: R. Fletcher, Harwell.

VA09A

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

The method is a quasi-Newton method and is described in R. Fletcher, Computer Journal, Vol. 13, 1970.

The user must provide initial estimates of the solution and provide a subroutine to calculate the function and its derivatives.

Remark: This routine supersedes VA01A and complements VA06A.

Versions: VA09A; VA09AD.

Calls: MC11A.

Language: FORTRAN, Date: April 1972, Size: 3.1K; 137 cards.

Origin: R. Fletcher, Harwell.

VA10A

To find a minimum of a general function $F(x_1,x_2,\dots,x_n)$ of n variables. It is assumed that the function is differentiable although values of the derivatives are not required.

A quasi-Newton method is used where derivatives are estimated by differences and is described in R. Fletcher, 'FORTRAN subroutines for minimization by quasi-Newton methods', AERE Report - R.7125.

The user is required to provide an initial estimate of the minimum position and a subroutine to evaluate the function. There are options for providing an estimate of the Hessian matrix. One of these, to use the Hessian from a previous problem, allow problems that are similar to be solved economically.

Remark: VA10A is likely to be more efficient and less likely to be effected by round off error than VA04A.

Versions: VA10A; VA10AD.

Calls: MC11A.

Language: FORTRAN, Date: April 1972, Size: 3.9K; 193 cards.

Origin: R. Fletcher, Harwell.

VA11A

To calculate the minimum of a general function of n variables when values of both the first and second derivatives with respect to the variables can be provided, i.e. find $\underline{x} = x_1,x_2,\dots,x_n$ to minimize a function $F(\underline{x})$ given $\frac{\partial F}{\partial x_j}$ $j=1,2,\dots,n$ and $\frac{\partial^2 F}{\partial x_j \partial x_i}$.

The method is based on the Newton method and is described in M.D. Hebden, AERE Report - R.7160.

The user can specify the absolute accuracies required in each variable and must give an initial estimate of the solution and provide subroutines to evaluate $F(\underline{x})$ and its first and second derivatives.

Versions: VA11A; VA11AD.

Calls: MC03AS.

Language: FORTRAN, Date: October 1971, Size: 9.2K; 400 cards.

Origin: M.D. Hebden, Harwell.

VB01A

To calculate the best least squares fit to given data t_i y_i w_i $i=1,2,\ldots,m$ by $f(t,\underline{x})$ a general function of t and n parameters $\underline{x} = x_1,x_2,\ldots,x_n$. Values of derivatives are required, i.e. find $\underline{x}$ such that

$$F(\underline{x}) = \sum_{i=1}^{m} w_i \{y_i - f(t_i,\underline{x})\}^2 \qquad m \geq n$$

is minimized given $\partial f_i/\partial x_j$ $i=1,2,\ldots,m;$ $j=1,2,\ldots,n.$

The method used is a modified Marquardt method, see R. Fletcher, AERE - R.6799. An initial estimate of the solution is required and also a subroutine to calculate values of $f(t_i,\underline{x})$ and its derivatives.

An estimate of the standard deviations of the final parameters and the variance-covariance matrix for the fit are returned to the user.

Remark:	Replaces the old version of VB01A/AD which used a Newton method.
Versions:	VB01A; VB01AD.
Calls:	MA10A, MC03AS, OA01A, SA01A and DERIV (a user routine).
Language:	FORTRAN, Date: June 1972, Size: 9.3K; 209 cards.
Origin:	R. Fletcher, Harwell.

VB03A

To minimise a sum of squares of m functions of n variables when values of derivatives with respect to the variables are available, i.e. find $\underline{x}=x_1,x_2,\ldots,x_n$ to minimise

$$F(\underline{x}) = \sum_{i=1}^{m} \{f_i(\underline{x})\}^2 \qquad m \geq n$$

given $\frac{\partial f_i}{\partial x_j}$ $i=1,2,\ldots,m;$ $j=1,2,\ldots,n.$

The method is an iterative descent method with quadratic convergence, see, R. Fletcher and M.J.D. Powell, Computer Journal, Vol. 6, No. 2, July 1963.

An initial estimate of the solution and the accuracy required in the solution must be specified. A subroutine must be provided for evaluating $f_i(\underline{x})$ $i=1,2,\ldots,m$ and the derivatives.

The subroutine may be applied to least squares data fitting problems.

Versions:	VB03A; VB03AD.
Calls:	MB01B, VD02A and CALCFG (a user routine).
Language:	FORTRAN, Date: May 1963, Size: 3.1K; 163 cards.
Origin:	M.J.D. Powell, Harwell.

VB05B

To calculate the best weighted least squares fit to given data x_i y_i w_i $i=1,2,\ldots,m$ by a cubic spline $S(x)$ which has knots ξ_j $j=1,2,\ldots,n$ specified by the user, i.e. finds $S(x)$ such that

$$F = \sum_{i=1}^{m} w_i^2 \{y_i - S(x_i)\}^2 \qquad 2 \leqslant n$$

is minimised and $\xi_1 \leqslant x_i \leqslant \xi_n$ for all i.

The spline is represented in terms of fundamental splines to obtain a sparse overdetermined system of linear equations in parameters defining $S(x)$. The equations are triangularised and solved using Householder type orthogonal transformations. Full advantage is taken of the sparsity structure resulting in a minimum storage requirement.

The spline $S(x)$ is represented on return to the user by the knots, and its values and the values of its first derivatives at the knots.

Remark: Values of $S(x)$ can be computed using TG01B.
Versions: VB05B; VB05BD.
Language: FORTRAN, Date: 1966, Size: 6.1K; 266 cards.
Origin: M.J.D. Powell, Harwell.

VB06A

To calculate a least squares fit to given data x_i y_i w_i $i=1,2,\ldots,m$ by a cubic spline (x), which has knots ξ_i $i=1,2,\ldots,n$ specified by the user, and subject to smoothing conditions imposed under the control of the user, i.e. given n-2 smoothing factors θ_j $j=2,3,\ldots,n-1$ minimises

$$F = \sum_{i=1}^{m} w_i^2 \{y_i - S(x_i)\}^2 + \sum_{j=2}^{n-1} \theta_i^2 d_i^2 \qquad m > 3$$

where d_j $j=2,3,\ldots,n-1$ are the discontinuities in third derivative of $S(x)$ at the knots ξ_j $j=2,3,\ldots,n-1$.

The spline is represented in terms of fundamental splines to obtain a sparse overdetermined system of linear equations which is triangularised and solved using Householder type orthogonal transformations. Full advantage is taken of the sparsity structure resulting in a minimal storage requirement. For discussion of the smoothing term see, M.J.D. Powell, AERE - TP.308, 1967.

The spline is represented on return by the knots, its values at the knots and its first derivative values at the knots.

Remark: Values of $S(x)$ can be computed using TG01B.
Versions: VB06A; VB06AD.
Language: FORTRAN, Date: July 1963, Size: 6.5K; 287 cards.
Origin: M.J.D. Powell, Harwell.

VC01A

To calculate a 'best' weighted least squares fit to given data x_i y_i w_i w_i $i=1,2,\dots,m$ by a polynomial $P_n(x)$ of specified degree n, i.e. find $P_n(x)$ such that

$$F = \sum_{i=1}^{m} w_i \{y_i - P_n(x_i)\}^2 \qquad m \geqslant n$$

is minimised.

The polynomial $P_n(x)$ is represented as a linear combination of orthogonal polynomials $Q_j(x)$ $j=0,1,\dots,n$ orthogonal over the point set x_i $i=1,2,\dots,m$, see, G.E. Forsythe, Journal of S.I.A.M., Vol. 5, 1957.

In addition to the recurrence relation parameters defining the $Q_j(x)$s the subroutine returns and prints the parameter variances, number of residual sign changes and residual variances.

Remark: PE07A and PE08A can be used to evaluate $P_n(x)$ or obtain its coefficients.
Versions: VC01A; VC01AD.
Language: FORTRAN, Date: July 1963, Size: 2.3K; 84 cards.
Origin: E.J. York*, Harwell.

VC03A

To calculate a smooth weighted least squares fit to given data x_i y_i w_i $i=1,2,\dots,m$ by a cubic spline S(x). The subroutine automatically chooses the 'best' knots ξ_i $i=1,2,\dots,n$ and smoothing factors θ_j $j=2,3,\dots,n-1$ so as to minimise

$$F = \sum_{i=1}^{m} w_i^2 \{y_i - S(x_i)\}^2 + \sum_{j=2}^{n-1} \theta_j^2 d_j^2$$

where d_j is the third derivative discontinuity at the jth knot. While maintaining a degree of smoothness the routine also attempts to follow trends in the data, see M.J.D. Powell, AERE - TP.307, 1967. The method is iterative; VB06A is used to solve the intermediate least squares problems.

The spline is represented by the knots, its values at the knots and its first derivative values at the knots.

Remark: Values of S(x) can be obtained using TG01B.
Versions: VC03A; VC03AD.
Calls: VB06A.
Language: FORTRAN, Date: Oct. 1967, Size: 5.1K; 267 cards.
Origin: M.J.D. Powell, Harwell.

VC04A

To fit a straight line to data, i.e. given data points x_i y_i $i=1,2,\dots,m$ and optionally weights w_i $i=1,2,\dots,m$ computes the best straight line fit, $y(x) = a + bx$, which minimizes the sum of squares

$$s = \sum_{i=1}^{m} w_i \{y_i - (a + bx_i)\}^2 \qquad m \geq 2$$

There are options for obtaining the fit x(y) = p + qy and information is also available for the general analysis of the *correlation problem.*

Versions: VC04A; VC04AD.

Calls: MC02AS, MC05AS, MX02A and OA03A.

Language: FORTRAN, Date: July 1972, Size: 4.2K; 240 cards.

Origin: J. Ledger*, Harwell.

VC05A

To calculate the best weighted least squares fit to given data x_i, y_i, w_i, i=1,2,...,m by a sum of n decaying exponentials, i.e. find a_j and b_j j=1,2,...,n to minimize

$$S = \sum_{i=1}^{m} w_i^2 \left(y_i - \sum_{j=1}^{n} a_j e^{b_j x_i}\right)^2 \qquad m \geq 2n,\ b_j \leq 0\ j=1,2,\ldots,n$$

A method is used (VA11A) which utilizes both first and second derivatives. Code for calculating values of the derivatives is not required of the user but is included in VC05A itself.

The user must specify the accuracy required in the b_js (the a_js are obtained via MA14A). The routine returns the parameter variance-covariance matrix, the sum of squares and the residuals.

Versions: VC05A; VC05AD.

Calls: MA14A, MA22A and VA11A.

Language: FORTRAN, Date: July 1973, Size: 7K; 181 cards.

Origin: M.D. Hebden, Harwell.

VD01A

To find a minimum of a general function f(x) of one variable to within a specified accuracy.

The method is iterative and an estimate of the minimum position at each stage is predicted by constructing a quadratic function defined by three previous estimates of the position.

The user must supply the initial estimate and indicate a reasonable change to be made in the variable to start off the search. Code must be provided to evaluate f(x).

Versions: VD01A; VD01AD.

Language: FORTRAN, Date: May 1964, Size: 1.4K; 97 cards.

Origin: M.J.D. Powell, Harwell.

VD02A

To find a minimum of a general function f(x) of one variable to within a specified accuracy when values of its first derivative can be provided.

The position of the minimum is predicted using a cubic defined by the function and derivative values at two previous estimates of the minimum.

The initial estimate is supplied by the user and an indication must be given as to a reasonable change to make to start off the search. Code must be provided to evaluate f(x) and its first derivative.

Versions: VD02A; VD02AD.
Language: FORTRAN, Date: May 1964, Size: 1.2K; 62 cards.
Origin: M.J.D. Powell, Harwell.

VE01A

To find the minimum of a general function $f(x_1,x_2,\ldots,x_n)$, of n variables, subject to the variables satisfying m linear inequality constraints

$$\sum_{j=1}^{m} c_{ij}\, x_j \geqslant d_i \qquad i=1,2,\ldots,m$$

when values of the derivatives $\frac{\partial f}{\partial x_j}$ $j=1,2,\ldots,n$ are available.

The method is that of Davidon's, based on using an approximation H to the inverse Hessian matrix of $f(\underline{x})$, but enabling linear inequality constraints to be dealt with by projection techniques.

An initial estimate of the solution which satisfies the constraints must be given. The user must also specify the accuracy required and give a lower bound for the value of $f(\underline{x})$. Values of $f(\underline{x})$ and $\frac{\partial f_i}{\partial x_j}$ must be provided through a subroutine.

VE01A provides additional options which allow it to be used efficiently for solving parametric programming problems, i.e. when it is required to vary the definition of $f(\underline{x})$ or vary the constraints in some way.

Remark: See also VE03A and VE05A.
Versions: VE01A; VE01AD.
Calls: MC03AS and FUNCT (a user routine).
Language: FORTRAN, Date: Dec. 1969, Size: 13K; 359 cards.
Origin: R. Fletcher, Harwell.

VE02A

To find a minimum of quadratic function of the form

$$f(\underline{x}) = \tfrac{1}{2}\underline{x}^T A\underline{x} - \underline{b}^T\underline{x}$$

where $A = \{a_{ij}\}$ is a symmetric matrix, $\underline{b}$ a vector and $\underline{x}$ the vector of n variables $x_1,x_2,\ldots,x_n$ which are sought subject to the linear constraints

$$l_i \leqslant x_i \leqslant u_i \qquad i=1,2,\ldots,n$$

and $$\sum_{j=1}^{n} c_{ij}\, x_n \geqslant d_i \qquad i=1,2,\ldots,m$$

Any of the inequalities may be designated as strict qualities.

For the method see, R. Fletcher, AERE - TP.401 and R. Fletcher, AERE - R.6370.

The subroutine offers several modes of operation including the case when A is positive definite and the case of parametric programming when f($\underline{x}$) and the constraints may be varied slightly.

Versions:	VE02A; VE02AD.
Calls:	LA02A, MB01B and MC03AS.
Language:	FORTRAN, Date: July 1970, Size: 9.4K; 303 cards.
Origin:	R. Fletcher, Harwell.

VE03A

To find the minimum of a general function f($\underline{x}$) of n variables $\underline{x} = x_1,x_2,\ldots,x_n$, subject to the variables satisfying m linear constraints

$$l_i \leqslant x_i \leqslant u_i$$

and $$\sum_{j=1}^{n} c_{ij}\, x_j \geqslant d_i \qquad i=1,2,\ldots,m$$

where any of the constraints may be made strict equalities. Derivatives $\frac{\partial f}{\partial x_j}$ j=1,2,...,n are required.

The method is given in R. Fletcher, AERE - TP.431. There are several modes of operation and certain modes require the user to supply an initial feasible solution. Scaling can be specified and a subroutine must be provided to compute values of the function f($\underline{x}$) and its derivatives.

Remark:	See also VE01A and VE05A.
Versions:	VE03A; there is no double precision version.
Calls:	LA02A, MC03AS and VE02A.
Language:	FORTRAN, Date: June 1971, Size: 17.9K; 536 cards.
Origin:	R. Fletcher, Harwell.

VE04A

To find $\underline{x} = \{x_1,x_2,\ldots,x_n\}$ that minimizes a quadratic function of the form

$$Q(\underline{x}) = \tfrac{1}{2}\, \underline{x}^T A \underline{x} - b^T \underline{x}$$

where $A = \{a_{ij}\}$ is symmetric matrix and $\underline{b}$ a vector and where $\underline{x}$ is subject to bounds $l_i \leqslant x_i \leqslant u_i$ i=1,2,...,n.

The method is that of R. Fletcher and M.P. Jackson (1973), Harwell Report TP.528.

The routine can be used to solve linear least squares data fitting problems in cases when the variables are subject to bounds. For this application there is an extra entry point which provides the variance covariance matrix for the fit.

Versions: VE04A; VE04AD.
Language: FORTRAN, Date: April 1973, Size: 6.7K; 282 cards.
Origin: R. Fletcher, Harwell.

VE05A

To find the minimum of a general function $f(x_1, x_2, \ldots, x_n)$, of n variables, subject to bounds and linear constraints of the form

$$l_i \leqslant x_i \leqslant b_i \qquad i=1,2,\ldots,n$$

and

$$\sum_{i=1}^{n} c_{ij} x_i \geqslant d_j \qquad j=1,2,\ldots,m$$

Values of the derivatives $\frac{\partial f}{\partial x_i}$ $i=1,2,\ldots,n$ are required. Variables can be specified as unbounded and constraints as strict equalities, and advantage can be taken of sparsity in the constraints.

The method is a variant of Goldfarb's algorithm and is described in the Harwell report, A. Buckley, TP.544.

The user specifies the accuracy required and must provide a subroutine to calculate values of $f(\underset{\sim}{x})$ and its first derivatives. An initial feasible point is optional.

The routine has been designed so that the user can optionally change certain parameters which are used to control the algorithm. These parameters and additional output parameters are stored in COMMON blocks.

Versions: VE05A; VE05AD.
Language: FORTRAN, Date: July 1973, Size: 25.3K; 1063 cards.
Origin: A. Buckley*, Harwell.

Z. Non-FORTRAN and System Facilities

ZA01AS

Provides the FORTRAN programmer with the facility for measuring the real time elapsed in executing a section of program.

The routine uses the 360/OS TIME facility to obtain a real time clock reading. It uses clock reading to update its own internal clock. Note: It is necessary to do it this way in order to allow timing across day changes. The ZA01AS clock is initialised to zero and does not give the time of day.

Remark: In a multi-programming environment elapsed time is not very useful for measuring the efficiency of programs and ZA02AS which times relative to activity should be used.

Language: 360/BAL, Date: Aug. 1967, Size: .1K; 57 cards.

Origin: R.C.F. McLatchie, Harwell.

ZA02AS

Provides the FORTRAN programmer with the following task timing facilities. (1) ZA02AS, to measure the c.p.u. activity in executing a section of program, i.e. using a clock which only ticks on when the program has control of the c.p.u. (2) ZA02BS request an interrupt to occur after a specified interval of time (relative to activity) when either, an ABEND with a user provided completion code is forced, or, control is given to a user provided subroutine; (3) ZA02CS return the amount of time still to run for the step; (4) ZA02DS to ABEND or pass control to a user subroutine, as for ZA02BS, at a specified interval before the step time chop is due.

The 370/OS STIMER and TTIMER facilities are used. The routine maintains an internal clock which is updated at every call.

The interrupt request passed to ZA02BS or ZA02DS may be changed or cancelled. All four facilities may be used in the same program.

Language: 360/BAL, Date: December 1970, Size: 1.2K; 606 cards.

Origin: R.C.F. McLatchie and M.J. Hopper, Harwell.

ZA03AS

To cause the transfer of data performed by the next READ or WRITE statement to be from storage to storage.

The user provides an array which is presented to the FORTRAN I/O support routines as the buffer to be used on the next I/O operation. On a READ the information put into the array by the user is transferred into the variables in the READ statement list. If the operation is a formatted READ the information is converted under FORMAT before being transferred. On a WRITE the process is reversed so that after the write has been performed the buffer array will contain the output record constructed from the list.

ZA03AS allows the FORTRAN programmer to manipulate character information by making use of the FORTRAN FORMAT facilities. It can also be used to re-read input records.

Language: 360/BAL, Date: Aug. 1967, Size: .2K; 74 cards.

Origin: SHARE program CORE I adapted for Harwell by R.C.F. McLatchie.

ZA05AS

Converts a line of ASCII data to an EBCDIC card image. It provides in conjunction with ZA03AS, a facility for reading lines of ASCII paper tape data and data cards.

The user provides the routine with an array containing the line of ASCII data and another array to receive the card image. The routine provides several checking options.

Language: 360/BAL, Date: 1967, Size: 1.3K; 257 cards.
Origin: Harwell.

ZA06AS

Enables a FORTRAN program to gain access to PARM information given on the EXEC card for the step.

The information must be given in the form Keyword = n, or Keyword = string or just Keyword. The user specifies to the routine the Keyword to be searched for then depending on the option selected either, the integer value n is returned, or, the string of text 'string' is returned or an indication of whether the Keyword was found is returned.

The routine locates the PARM field using the register 1 value found in the first save area for the task.

Language: 360/BAL, Date: Jan. 1970, Size: .4K; 174 cards.
Origin: M.J. Hopper, Harwell.

ZA07AS

To set a return code for a job step, i.e. a code number between 0 and 4095 which can be tested on subsequent steps with the EXEC card COND parameter. This allows steps of a job to be executed conditionally on the success or failure of a previous step.

A call to ZA07AS is equivalent to a CALL EXIT but with an argument specifying the return code.

Remark: The FORTRAN statement STOP n also returns a code but its value n must be given as a constant.
Language: 360/BAL, Date: Jan. 1970, Size: .1K; 25 cards.
Origin: M.J. Hopper, Harwell.

ZA08AS

To obtain the time of day in hours, minutes and seconds. The time is returned as an 8 byte character string in the form hh.mm.ss which can be immediately printed using an A8 format.

The 360/OS TIME facility is used.

Language: 360/BAL, Date: 1967, Size: .1K; 45 cards.
Origin: W.E. Hart*, Harwell.

ZA09AS

To obtain the current date in days, months and years. The date is returned as an 8 byte character string in the form dd/mm/yy which can be printed using an A8 format.

The 360/OS TIME facility is used.

Language: 360/BAL, Date: Aug. 1967, Size: .3K; 85 cards.
Origin: W.E. Hart*, Harwell.

ZA10AS

To obtain the current date in days, months and years. The date is returned in three parts, the day (integer), the month abbreviated (characters) and the year (integer).

The 360/OS TIME facility is used.

Language: 370/BAL, Date: 1967, Size: .3K; 62 cards.
Origin: Harwell.

ZA12AS

To allow a FORTRAN program to send a message to the 360 operator's console.

The 360/OS WTO facility is used.

Language: 360/BAL, Date: Jan. 1968, Size: .2K; 59 cards.
Origin: W.E. Hart*, Harwell.

ZA15AS

To give the FORTRAN user the facility of translating from one internal character code to another. Five options are available, (i) EBCDIC to BCD, (ii) BCD to EBCDIC, (iii) a translation specified by a 256 byte translate table supplied by the user, (iv) mixed EBCDIC/BCD to EBCDIC, and (v) EBCDIC to UNIVAC field data code.

Remark: The routine is useful for translating card decks punched in BCD into EBCDIC for use on the 360.
Language: 360/BAL, Date: March 1971, Size: 1.2K; 217 cards.
Origin: M.J. Hopper, Harwell.

ZA16AS

To allow a FORTRAN program to swop the names of two disk data sets during execution and/or return the number of tracks currently in use in a data set.

Its main purpose is to allow a disk data set to be maintained through updates in a way which is safe and is completed in a single job step. This is done by creating the new version as a temporary data set and swopping names if the update is successfully completed and allowing the system to delete the unwanted version.

Language: 360/BAL, Date: October 1972, Size: 1.5K; 206 cards.
Origin: H.A. Kearsey, Harwell.

ZA17AS

To allow a FORTRAN program to extract information from the TIOT system control block. There are options to, (a) obtain the jobname, (b) check for the presence of a DDname, and (c) return a copy of the TIOT in an array provided by the user.

Language: 360/BAL, Date: November 1972, Size: .3K; 93 cards.
Origin: H.A. Kearsey, Harwell.

ZA18AS

To allow a FORTRAN program to obtain the BLKSIZE, LRECL and RECFM values from the DCB fields of either a JFCB (DD statement), a DSCB (disk data set label) or from an open DCB.

Language: 360/BAL, Date: Dec. 1972, Size: .8K; 202 cards.
Origin: M.J. Hopper, Harwell.

ZE01A

To estimate the rounding errors generated in floating point arithmetic. The method is independent of computer.

A 100 pairs of positive numbers are generated chosen to be not exactly representable either on a decimal computer or one working to base 2, 4, 8, 16, etc. Floating point operations are carried out on the numbers in such a way as to give estimates of the maximum and mean errors.

Versions: ZE01A; ZE01AD.
Language: FORTRAN, Date: June 1966, Size: .8K; 35 cards.
Origin: A.R. Curtis, Harwell.

ZR01AS

To set a return code for a job step, i.e. a code number between 0 and 4095 which can be tested on subsequent steps with the EXEC card COND parameter. The routine must be link-edited into the user's program as the main entry to the program.

Remark: See also ZA07AS and FORTRAN STOP.
Language: 360/BAL, Date: Sept. 1968, Size: .2K; 24 cards.
Origin: D. McVicar, Harwell.

ZR02AS

To provide, (i) the entry point address of the module in which the subroutine is incorporated, (ii) the length and contents of the PARM field.

The PARM field is returned in an array supplied by the user.

Remark: See also ZA06AS for PARM field processing.
Language: 360/BAL, Date: Jan. 1970, Size: .2K; 94 cards.
Origin: G. Hunter*, Harwell.

ZR03AS

To allow a FORTRAN program to obtain copies of certain system control blocks, (CDE, channel prog., CVT, DCB, DEB, DSCB, IOB, JFCB, RB, TCB, TIOT, UCB). Only one control block can be copied at a time and the user must specify how many bytes are to be copied. There is an option to allow the user to specify the address of the area to be copied thereby making it possible to chain through control blocks to obtain areas of interest not directly covered by the routine.

Versions: ZR03AS.
Language: 360/BAL, Date: Jan. 1973, Size: 1.2K; 356 cards.
Origin: M.J. Hopper, Harwell.

ZR10AS

In the event of certain types of program interrupt, to print out information to assist in determining the cause and precise location of the interrupt.

The interrupts intercepted by ZR10AS are those with completion codes OC1 to OC5.

The common ones occurring in FORTRAN programs are, OC5 Addressing (array subscript error?), OC1 Operation (overwriting code or address tables?), OC4 Protection (subscript error?).

The information printed by ZR10AS includes, interrupt address, old PSW, register contents, 96 byte core dump of area around interrupt address.

The 360/OS SPIE facility is used to intercept the interrupts; the addresses etc. are retrieved using 360/OS control blocks and printed output produced using normal FORTRAN I/O routines. Execution is terminated using a CALL EXIT.

Language: 360/BAL, Date: 1968, Size: 1.3K; 263 cards.
Origin: K. Moody, IBM, modified: G. Hunter*, M.J. Hopper, Harwell.

ZV01AD

To allow a FORTRAN program to access information contained on a DD card.

The subroutine obtains the information by reading the Job File Control Block (JFCB) associated with the DD card from the System Job Queue.

The DDname must be of the form FTnnF001 and options are provided for obtaining; the volume serial no.; the data set name, and the JFCB (which contains an internal representation of the DD card information).

Language: 360/BAL, Date: Nov. 1967, Size: .4K; 57 cards.
Origin: D. McVicar, Harwell.

AERE Reports

Some of the more recent library routines written by the Numerical Analysis Group at Harwell have been published as A.E.R.E. reports. Each report gives details beyond those normally found in the library write ups and also contains a listing of the routine. The reports can be obtained from Harwell or from H.M. Stationery Office. The following list gives the routines which have been published in this way.

GA05A	'A FORTRAN subroutine for determining rapidly whether points are inside a plane region', J.K. Reid, R.7298.
LA02A	'The calculation of feasible points for linearly constrained optimization problems', R. Fletcher, R.6354.
MA15C	'Two FORTRAN subroutines for direct solution of linear equations whose matrix is sparse, symmetric and positive-definite', J.K. Reid, R.7119.
MA16A	'A FORTRAN subroutine for the solution of large sparse sets of linear equations by conjugate gradients', J.K. Reid, R.6545.
MA17A	See MA15C.
MA18A	'FORTRAN subroutines for the solution of sparse sets of linear equations', A.R. Curtis and J.K. Reid, R.6844.
MA21A	'FORTRAN subroutines for the solution of linear equations, inversion of matrices and evaluation of determinants', S. Marlow and J.K. Reid, R.6899.
MA22A	See MA21A.
MA23A	See MA21A.
MA24A	See MA21A.
MB11A	'A FORTRAN subroutine to invert a rectangular matrix of full rank', M.J.D. Powell, R.6072.
MC09A	See NS03A.
MC10A	See MA21A.
MC12A	See MA18A.
ME07A	See MA21A.
NB01A	See OB11A.
NS01A	'A FORTRAN subroutine for solving systems of non-linear algebraic equations', M.J.D. Powell, R.5947.
NS03A	'FORTRAN subroutines for the solution of sparse systems of non-linear equations', J.K. Reid, R.7293.
OB11A	'A FORTRAN subroutine for drawing a curve through a given sequence of data points', S. Marlow and M.J.D. Powell, R.7092.
OB12A	'A FORTRAN subroutine for plotting a cubic spline function', S. Marlow and M.J.D. Powell, R.7470.
TD02A	See NS03A.

TS01A	'A FORTRAN subroutine for calculating a cubic spline approximation to a given function', M.J.D. Powell, R.7308.
VA06A	'A FORTRAN subroutine for unconstrained minimization, requiring first derivatives of the objective function', M.J.D. Powell, R.6469.
VA07A	'A modified Marquardt subroutine for non-linear least squares', R. Fletcher, R.6799.
VA08A	'A FORTRAN subroutine for minimization by the method of conjugate gradients', R. Fletcher, R.7073.
VA09A	'FORTRAN subroutines for minimization by quasi-Newton methods', R. Fletcher, R.7125.
VA10A	See VA09A.
VE02A	'A FORTRAN subroutine for general quadratic programming', R. Fletcher, R.6370.

PART II: Topics of General Interest to Users of the Harwell Subroutine Library

A. Using the Harwell Subroutine Library

1. Naming conventions

Library subroutines fall into specific groups, each group being associated with a different type of problem, i.e. Differential equations, matrix calculations, input/output aids, etc. Each main group is further subdivided and the grouping is reflected in the subroutine name.

The name is always of a standard form 'aannbc' where a, b and c are alphabetic and n numeric. This form as well as providing scope for classification was chosen as a form unlikely to conflict with the names of user subroutines.

The first two characters 'aa' identify the main classification group and sub-group to which the subroutine belongs, e.g. in the matrix calculation group which has names beginning with M

MA	solutions of linear equations
MB	inverses of matrices
MC	matrix and vector operations

The third and fourth characters, the two digits 'nn', identify the subroutine within the sub-group, e.g.

MA07 solves a band structured system of linear equations
MA12 solves an upper Hessenberg system of linear equations

The fifth character 'b' is primarily to denote a close relationship. If a modified version of a subroutine is introduced and for some reason the new version cannot completely replace the old the fifth character is used to distinguish between the two versions, e.g.

PA02B is essentially the same subroutine as PA02A except that the B version generates an error estimate.

The fifth character is also used when

(a) the library subroutine as presented to the user is really a package of subroutines,

(b) the subroutine contains additional entry points, and

(c) the library subroutine contains named COMMON areas.

The library subroutine DC01AD is a good example of this for it consists of 7 subroutines DC01AD, DC01CD, DC01DD, DC01ED, DC01FD, DC01GD and DC01ZD, and has 2 secondary entry points DC01BD and DC01XD, and 6 named COMMON areas DC01HD, DC01ID, DC01JD, DC01KD, DC01LD and DC01YD.

The sixth character 'c' can be one of the following

omitted	- the subroutine expects its real arguments to be single length, i.e. REAL*4, and usually works in single length arithmetic.
S	- written in 360/assembler code and will expect its real arguments to be given single length.

D - for both FORTRAN and assembler it expects real arguments to be double length, i.e. REAL*8, and usually works in double length arithmetic.

I - for both FORTRAN and assembler it expects its arguments to be integer, i.e. INTEGER*4.

A few of the subroutines do not conform to the above contentions.

2. Write ups (subroutine specifications)

2.1 What a write up should tell you

Each subroutine in the library has associated with it a set of specification notes. Each set of notes, which we shall refer to as write ups, describes the subroutine's requirements and the facilities it offers. A write up consists usually of the following sections:

(1) Purpose: a brief description of what the subroutine does. The content of this section should provide sufficient information for the potential user to decide whether the subroutine is going to be of any use.

(2) Argument list: gives the details required to use the subroutine, i.e. argument specifications, initial values, variable types, calling sequences if any. In the description it is assumed that the user is familiar with the idea of using a SUBROUTINE or FUNCTION routine and is aware of the differences between FORTRAN types such as REAL*8, REAL*4, LOGICAL*1, etc.

(3) General information: covers such things as use of COMMON, specification of user supplied subroutines, other library routines called, details of printed output, etc.

(4) Method: a brief description of the method used or a reference to a description of the method.

(5) Example: an example of the use of the subroutine.

2.2 Problems in interpreting write ups

The lack of rigid rules in respect to the layout and presentation of the information contained in write ups does sometimes lead to misunderstanding. Usually the trouble arises from the conflict between the variation in precision, length and type of the FORTRAN variables used on the IBM/360 and trying to make one write up describe the requirements of all versions of the routine. The following sections may be found useful in this respect.

2.3 Double precision and other versions

The use of double and single length versions of a library routine are often so similar that it is a practice to produce only one write up which is used to serve both versions. The write up will be written in terms of the single length version and it is assumed that the user can deduce what is required for the double length version.

Recent write ups give at the top right hand corner of the first page the names of the versions covered by the write up in an abbreviated form using the '/' character to combine the names, e.g. write up MX01A/AD/I describes the three routines

MX01A	with REAL*4 arguments
MX01AD	with REAL*8 arguments
MX01I	with INTEGER*4 arguments.

When a write up is fairly old and gives only the single length name it can be assumed that a double length version exists and its name is obtained, in the case of a FORTRAN routine, by adding a D to the end and, in the case of an assembler routine, replacing the final S by a D. Only a few library routines have no double length version, apart that is from those for which double precision is not appropriate.

2.4 Argument types

For write ups that say nothing about the type or length of arguments, usually pre-360 write ups, the FORTRAN implicit convention for types should be used, i.e. real variable names begin with letters A to H or O to Z and integers I to N. In more recent write ups the type and sometimes the length are stated. Where the length is omitted real arguments are assumed to be 4 or 8 byte reals depending on whether the single or double length version is being used. Integers unless otherwise stated are always 4 byte integers.

2.5 Obtaining write ups

Write ups for library routines are contained in five filing cabinets marked 'Harwell Subroutine Library Public Files' which stand in the computer reception area in Building 8.12. Users are invited to take any copies required or if it is difficult to get access to the computer building request computer reception to send them. For requests from outside Harwell see section on external users.

2.6 Write up queries

If a user has difficulty in either obtaining or understanding a write up the queries facility offered by the library should be used. The librarian would particularly welcome any constructive criticisms or comments concerning write ups so that they can be improved and mistakes corrected.

From time to time write ups are modified or corrected or sometimes completely re-written and these changes are publicised through the subroutine library's information service, see section 4.

For a user who does not keep back numbers of Harwell subroutine library bulletins there is no easy way to find out whether a write up obtained some time ago is still the current version. It is hoped sometime in the future to record the date when a write up has been re-issued in a computer generated index. Meanwhile it would be appreciated if before notifying us of any errors in a write up a check is made to see if it is the most up-to-date version.

3. Using library subroutines in programs

3.1 How to get hold of the routines

To use a library routine in a program first obtain a write up, code the call to the routine into your program setting the arguments as specified in the write up and then run the program. Compiled versions of the subroutines being called will automatically be included into your program by the Linkage Editor, or Fast Loader depending on which Job Control Language procedure you are using.

The subroutines are fetched from a library (p.d.s. data set) called HAR.LIB which contains compiled versions of all the routines in the Harwell Subroutine Library. HAR.LIB is normally joined with the IBM FORTRAN library, i.e. the one which contains routines such as EXP, SQRT, under the DD name SYSLIB. If the user's program is to be stored permanently on disk for future use routines from these two libraries can be optionally omitted from the stored program by specifying the Linkage Editor option NCAL, see Harwell Users Manual for details.

3.2 What to do if trouble occurs

It may happen that during the course of developing a program which uses library routines an interrupt occurs due to an abnormal condition and that at the time of the interrupt one of the library routines had control. It is always tempting to suspect the library routine because it is the unknown quantity.

On the whole library routines tend to be well tried and tested, by the mere fact that they are used a lot, and in our experience in over 90 per cent of the queries concerning the malfunctioning of library routines it turns out to be the calling program which was in error. If you have trouble come and see us but we do ask that you make reasonably sure before coming to us that the error is associated with a library routine and you bring along plenty of evidence, a print out of the argument values before entry is always useful.

4. Library information facilities

4.1 General queries

General queries concerning library routines should be directed to Mr. S. Marlow, Building 8.9, Ext. 2930.

4.2 Computer generated index

A list of names of all the routines in the library is kept on disk for access by computer. Each entry in the list consists of a sentence or two describing the purpose of the routine. It is kept fairly up to date and users may for this reason still find it useful.

There is usually a copy of the index on the general notice board in the computer reception area which may be referred to but should not be taken away. You may obtain your own copy by running a job on the computer using the procedure HARSLX. The following

```
//     user's JOB card
//     EXEC    HARSLX
```

will produce a listing of the library index. The job can be run in 20K.

4.3 Day to day information files

The most up to date information on library changes is kept in four HUW files in the library's line file library, identifier HSL. The four files are

AD-A092 033 ATOMIC ENERGY RESEARCH ESTABLISHMENT HARWELL (ENGLAND) F/G 12/1
HARWELL SUBROUTINE LIBRARY. A CATALOGUE OF SUBROUTINES (1973).(U)
JUL 73 M J HOPPER
UNCLASSIFIED AERE-R-7477 NL

2 OF 2
AD A
092033

END
DATE
FILMED
1-81
DTIC

HSLS - lists new routines recently put in the library, giving purpose and date of update into the library.

HSLM - lists the routines that have been modified or corrected and may include reports of uncorrected errors, although this is not often as errors are usually put right immediately. Each entry will give the reason for the modification and date of update into the library.

HSLW - gives a list of new and modified write ups with the date when they were put into the public filing cabinets for general use.

HSLP - a promised list of routines, that is routines currently being written and will be going into the library in a short time. The name of the author and extension number is usually given to allow a potential user of the routine to contact the author to discuss special problems.

The lists cover a period of about 3 months and are ordered so that the most recent changes come first, just like the TODAY file run by the central computer group. To examine the files just to see the latest changes you should copy the relevant file into your own line file library, examine it pressing the attention key when you have seen enough, and finally destroying it, eg. to see what new routines have been put in the library

```
COPY   HSLS   HSLS   HSL
EX   HSLS
```

To make a more permanent copy a macro called MNEWS can be used. This is also kept in HSL and must be copied over before it can be used. MNEWS when used with no parameters specified will type out all four files in a paged form suitable for pinning on a notice board. The pages are designed to be stapled together at the top and are of different length so that each section projects below the other for easy reference. Note that before using MNEWS all four files must be copied over into your file library. MNEWS can be used to list just one of the files at a time by specifying the file name of the file to be listed as a parameter, eg. to list the file HSLM

```
COPY MNEWS MNEWS HSL
COPY HSLM HSLM HSL
MNEWS HSLM
```

4.4 The library news sheet

The macro MNEWS is used by us to produce a news sheet which consists of the four files HSLS, HSLM, HSLW and HSLP. Copies of the news sheet are put on the notice boards sited at the collection points around the Harwell establishment. They are sent out every two or three weeks or when sufficient changes have been made to warrant the effort.

4.5 The library status index

The library has a program, which it uses for maintenance purposes, which will scan through the source and load module libraries and produce a list of certain status information pertinent to each routine. Each entry in the list of routines will give

Date of update into source library.

Date of update into load module library (compiled versions).

Number of cards in source deck.

Size of compiled routine in bytes.

Language and compiler.

Names of private routines and COMMON areas integrated with routine.

Entry point names.

External references, names of routines called by the routine.

Names of all library routines required to be loaded with the routine.

Names of all routines in the library which call this routine.

A copy of this status index can usually be found on the general notice board in the computer reception area. It is not run off very often and the list on the notice board may be several weeks out of date. The program is not generally available but any user sufficiently interested should ask us for the JCL details. It can be used on any load module library.

4.6 The library bulletin

This is used to publicise subroutine library news. It is produced approximately every three or four months and gives details of new routines introduced in the period covered. It gives news of changes to routines and write ups, errors, proposed deletions and promised routines. The bulletin is also used to publicise policy changes and discusses the effect on the library of changes to the computer hardware or the computer operating system. Contributions by nonlibrary staff are always welcome, particularly if they are concerned with comparisons carried out between routines or describe difficulties experienced with routines.

4.7 Summary of information sources

The following list summarizes the various sources of library information and gives the approximate frequency with which they are updated or circulated.

(a) HUW files, see 4.3, (within hours of the change being made).

(b) the subroutine library notice board in computer reception area, displays lists of the four HUW files (within same as (a)).

(c) news sheet on notice boards at collection points, see 4.4 (1 - 4 weeks).

(d) computer generated index, see 4.2 (2 months).

(e) bulletin, see 4.6 (3 - 6 months).

(f) supplements to this library catalogue (4 - 6 months).

(g) this catalogue of subroutines (2 years).

(h) status index, copy on notice board, see 4.5 (2 - 4 months).

5. Obtaining copies of library routines in source form

In exceptional circumstances a user may require a copy of a library routine in source form either as a listing produced on the line printer, a card deck or as a file in HUW the Harwell teletype system. A complete copy of the library in source form is kept on a magnetic tape which is always kept at hand in the computer room for the operators to mount when requested.

5.1 Listings and card decks

To obtain copies of routines from the tape a user must run a job using the catalogued procedure HARSLS. This procedure allows the user to obtain listings and card decks which may be punched in either EBCDIC or BCD code. Other options are available for specifying sequence numbering, changing output stream number= ınd specifying the line count for paging.

The statements required to run the basic form of the procuedure, i.e. with options defaulted (details of defaults further on), are as follows

```
//      user's job card...,REGION=90K
//   EXEC HARSLS
//SYSIN DD *
   control cards as described below
/*
```

The control cards must be of the following form

```
*keyword,name1(n1),name2(n2),....etc.
```

The character '*' must be in column one of the control card and the rest following with no embedded blanks, no continuation of cards is allowed. The keyword can be any one of the following

LIST	for listings on the line printer.
PUNCHE	for EBCDIC card decks (and listings).
PUNCHB	for BCD card decks (and listings).

name1,name2,...,etc. are the names of the library routines required and the numbers n1,n2,.. in parenthesis should specify the number of copies required. If the parenthesis and numbers are omitted only one copy is assumed.

Restriction: Not more than 50 names may be specified all told and the number of copies given in parenthesis is restricted to a maximum of 10.

Example: the following

```
//      job card
//   EXEC HARSLS
//SYSIN DD *
*LIST,FA01AS,KB01A(3),VA05AD
*PUNCHE,MA14A
/*
```

would produce one listing each of FA01AS and VA05AD with 3 listings of KB01A and a card deck punched in EBCDIC code with a listing of MA14A.

HARSLS is a magnetic tape job and must be run as such. The tape can be referred to as the HARWELL SUBROUTINE LIBRARY SOURCE TAPE or just H.S.L. SOURCE TAPE and must be READ only and file protected. Users are referred to the Harwell User's Manual for job card and operator instructions relevant to running a job which uses magnetic tape, produces card output and many lines of printer output.

The procedure offers some additional options which are requested through the PARM field of the EXEC card. The following are currently available.

PRINT=n	specifies the FORTRAN stream number which is to be used for listings, default n=6.
PUNCH=n	specifies the FORTRAN stream number which is to be used for punching card output, default n=7.
LINECNT=n	specifies the number of lines per page for listings, default n=42.
NOSEQL	specifies that no line numbers are to be printed on the listings, if NOSEQL is omitted line numbers are printed.
NOSEQP	specifies that no sequence numbers are to be punched on card output, if NOSEQP is omitted sequence numbers are punched in columns 73-80 of each card.
INC=n	specifies the increment to be used for sequence numbering punched card output, default n=1.
NOLET	specifies that if any control card errors were detected the job is to be terminated (all cards are scanned for errors first). If NOLET is omitted the program will ignore all the requests on any control card found to contain errors but will continue to retrieve routines requested on control cards free from errors.
SPOOL=n	controls whether the program spools the routines from tape to disk before producing the required number of copies. If more than n copies of the routine are requested it is spooled to disk first. Default n=1.

Example of specifying options: the following statements

```
//      job card
//   EXEC HARSLS,PARM='INC=100,LINECNT=60,NOSEQL'
//SYSIN DD *
*PUNCHE,PD02A
/*
```

would produce a card deck of PD02A with sequence numbers punched in columns 73-80 starting 100,200,300,...,etc. and a list of the routine with no line numbers printed and page throws every 60 lines.

5.2 IBM Scientific Subroutine Package: SSP

A copy of the IBM SSP library is held on magnetic tape and the procedure HARSLS can be used to obtain listings and card decks. To use HARSLS for this purpose you must

(i) punch LIB=SSP on the EXEC card, e.g.

```
// EXEC  HARSLS,LIB=SSP,PARM='....'
```

(ii) refer to the tape in operator instructions as the SSP LIBRARY SOURCE TAPE.

To use the SSP routines in a program you must first obtain a source deck and include it in your program. The SSP library is not available on the system in compiled form (as a load module library). It is not possible for Harwell Subroutine Library staff to deal with queries concerning the internals of SSP routines and we take no responsibility for the proper working of the routines.

The current version of the library on tape is equivalent to version III mod 1 except for the sample program packages DASCR,REGRE,STEPR,MCANO,FACTO,POLRG and KOLM which are still version III mod 0. A list of routines available is given in the IBM document 'H20-0166-5 SYSTEM/360 Scientific Subroutine Package Version III; Application description'. A more detailed list including listings of the routines and write ups is given in 'H20-0205-3 SYSTEM/360 Scientific Subroutine Package, Version III; Programmer's Manual'. Copies of these can be seen in the information room Building 8.12 and the Mathematics Library in Building 8.9, room G.16.

5.3 Obtaining library routines as HUW files

A library routine can be put into a user's line files by means of a HUW macro called MHARSLS. The macro will construct and run a JCL file in the user's line file to copy over the routine. The user supplies the name of the subroutine required, only one name is allowed, and a copy of the routine is put into as many files needed to contain it, up to a maximum of 4. The name of the first file created will be that of the subroutine itself and subsequent files will have the same name but with the digits 1,2,3 added to the end. The files will have a file type of U and it is up to the user to change the type using the FILETYPE command.

As the card output facility of HARSLS is used the first line of the first file generated will be a HARSLS header card for punched output. This is copied as a comment card and gives the date the routine was last put onto the tape, this may be useful to you, delete if not required.

In addition to the line file output a normal HARSLS line printer listing is produced which should be consulted in the event of failure, i.e. when the routine requested is not found on the tape.

5.4 To run the macro MHARSLS

The macro is kept on the line file with identifier HUW and must be first copied into the user's line file. The statement

```
COPY MHARSLS MHARSLS HUW
```

will do this. It takes up about 2 blocks and a further 2 blocks will be taken by the JCL file called HUWSLS which it generates. To use the macro type in

```
MHARSLS name,SSP,NOCOMS
```

where SSP and NOCOMS can be omitted.

name	is the name of the library routine required.
SSP	is coded if the routine is in the IBM SSP library. If omitted the Harwell Subroutine Library is assumed.
NOCOMS	is coded if comment cards in the routine are not to be copied across. The option is useful when line file space is short and is particularly needed for SSP routines which tend to contain rather a high proportion of comment

B. Data fitting and approximation

1. Introduction

We consider the problem of calculating a function f(t) to fit some data, or for use in subsequent computer calculations.

In the data fitting problem we are given a number of measurements, (y_i, t_i) $(i=1,2,\dots,m)$ say, where y_i is the measured value of the function at the point t_i, and we wish to calculate a function f(t) such that the differences $\{y_i - f(t_i)\}$ are small for all values of i. For example for physical reasons we may want f(t) to have the form ae^{bt}, where a and b are parameters, and in this case we may calculate the values of a and b to minimize the expression

$$\sum_{i=1}^{m} \{y_i - ae^{bt_i}\}^2 . \tag{1}$$

In the approximation problem we are given a function y(t), specified perhaps by a computer subroutine, and we wish to calculate another function, f(t), that is close to y(t) and that is especially suitable for computer calculation. For example it may take one minute to calculate a value of y(t), but we may be able to find a sufficiently accurate approximation f(t), whose values take only a few milliseconds to compute, and therefore it would save computer time to use f(t) in place of y(t) in any subsequent calculations.

In a data fitting or approximation problem we must decide on the form of f(t), which is discussed in Section 2. Also we must decide on how we will measure the goodness of an approximation f(t) to y(t), and this matter is discussed in Section 3. Then in Section 4 we summarise the numerical methods that are held in the Harwell subroutine library for data fitting and for approximation.

2. The choice of f(t)

Frequently one lets f(t) be a polynomial

$$f(t) = \sum_{j=0}^{n} a_j t^j . \tag{2}$$

where the degree is prescribed, and where the values of the coefficients a_j have to be calculated. However it is often better to let f(t) be the ratio of two polynomials

$$f(t) = \sum_{j=0}^{n} a_j t^j \Big/ \sum_{j=0}^{n'} b_j t^j , \tag{3}$$

where now n and n' are given. This case is called "rational approximation", and it is especially efficient when the function to be approximated, y(t), is analytic. However if y(t) has many peaks, then it is usually worthwhile to divide the range of t into pieces, and this can be achieved automatically by letting f(t) be a "cubic spline function", which is a function that is composed of cubic polynomial pieces, joined so that the function is continuous and has continuous first and second derivatives. The use of cubic splines is recommended, because we have a number of subroutines for calculating them.

For all these forms of f(t) we have to calculate the values of parameters like $a_j(j=0,1,\ldots,n)$ in expression (2), and it is feasible to let f(t) have many forms that depend on parameters. A quite general and very useful form is

$$f(t) = \sum_{j=0}^{n} a_j \phi_j(t) \tag{4}$$

where the functions $\phi_j(t)$ are prescribed. Sometimes these functions are suggested by knowledge of the application, and sometimes they are guessed. For example if t denotes a vector of two variables, (t_x,t_y) say, then it may be convenient to try the approximation

$$f(t) = a_0+a_1t_x+a_2t_y+a_3t_x^2+a_4t_y^2+a_5t_xt_y \; . \tag{5}$$

Note that in expression (4) the parameters a_j occur linearly, and in this case we say we have a "linear approximation problem". The solution of linear problems is usually straightforward.

The general approximation problem occurs when

$$f(t) = f(t,a_1,a_2,\ldots,a_n) \; , \tag{6}$$

and this equation states that f(t) depends in a general way on the parameters $(a_1,a_2,\ldots,a_n)$. For example in equation (1) the approximating function is ae^{bt}, so it is the case

$$f(t,a_1,a_2) = a_1e^{a_2t} \; . \tag{7}$$

We do have subroutines that calculate the required values of the parameters $(a_1,a_2,\ldots,a_n)$ in the general case, but these calculations are less sure than in the linear case.

3. <u>The error of the approximation</u>

Having chosen a function $f(t,a_1,a_2,\ldots,a_n)$, we must calculate values for the parameters $(a_1,a_2,\ldots,a_n)$, so we require a criterion to fix the best values of the parameters. Frequently, especially in data fitting problems, we minimize the sum of squares of the errors, in which case we require the values of $(a_1,a_2,\ldots,a_n)$ that give the least value of the expression

$$\sum_{i=1}^{m} \{y_i - f(t_i,a_1,a_2,\ldots,a_n)\}^2 \; , \tag{8}$$

for example expression (1). However instead of minimising this expression one may prefer to minimize

$$\max_{i=1,2,\ldots,m} |y_i - f(t_i,a_1,a_2,\ldots,a_n)| \qquad (9)$$

or

$$\sum_{i=1}^{m} |y_i - f(t_i,a_1,a_2,\ldots,a_n)| \; . \qquad (10)$$

We prefer expression (9) when we require to make the largest difference $|y_i - f(t_i,a_1,a_2,\ldots,a_n)|$ as small as possible, and it is often worthwhile to use expression (10) if it is possible that just a few of the data errors are very large. But usually expression (8) is preferred, perhaps with some prescribed positive weights $(w_1,w_2,\ldots,w_m)$, in which case the expression that is minimized is

$$\sum_{i=1}^{m} \{w_i[y_i - f(t_i,a_1,a_2,\ldots,a_n)]\}^2 \; . \qquad (11)$$

In a similar way weights can be included with expressions (9) and (10).

Usually the different choices (8), (9) and (10) lead to different methods of calculation for obtaining the parameters $(a_1,a_2,\ldots,a_n)$, and as a general rule expression (8) is the easiest to use. One exception is that if a rational approximation (3) is required, then expression (9) is the easiest. In practice one should often minimize the expression that gives the easiest calculation, because although the calculated approximation will depend on the function that is minimized, the differences in these approximations are frequently unimportant.

So far the remarks of this section concern data fitting, for the situation is a little different when one wishes to approximate a given mathematical function. In this case the function $y(t)$ may be defined over an interval, say $0 \leq t \leq 1$, and then the analogue of expression (9), for instance, is the expression

$$\max_{0 \leq t \leq 1} |y(t) - f(t,a_1,a_2,\ldots,a_n)| \; . \qquad (12)$$

There are some subroutines for minimizing expressions like the one above, and then $y(t)$ must be specified by a computer subroutine.

Another common method of function approximation is interpolation. Here one calculates just enough values of $y(t)$, say $y(t_1)$, $y(t_2),\ldots,y(t_m)$, to determine the parameters of $f(t)$ by satisfying the equations

$$y(t_i) = f(t_i), \; i=1,2,\ldots,m. \qquad (13)$$

Of course if $f(t)$ has the form (6) then $m=n$. This method is unsuitable for data fitting because there is no smoothing of data errors.

Another disadvantage of interpolation is that the choice of points $(t_1,t_2,\ldots,t_m)$ can be critical. It is therefore more satisfactory from the point of view of robustness let m exceed the number of parameters, and to minimize say the sum of squares (8).

4. The subroutines of the Harwell library

For the interpolation problem (13), the following subroutines are available. When f(t) is a polynomial then either its coefficients can be calculated by subroutine PC02, or the value of f(t) for a single value of t can be calculated by subroutine TB02. When f(t) is a cubic spline function then use TB04 or TB05 depending on whether f(t) is to be periodic. Both these routines place the knots of the spline at the interpolation points. When f(t) is a more general function of the form (4), then the equations (13) give a system of simultaneous linear equations for the required parameters a_i, which can be solved by subroutine MA21. When f(t) has the general form (6), then the equations (13) may be non-linear, in which case subroutine NS01 is recommended. However if f(t) is a rational function (3), then a direct method for calculating its coefficients from equation (13) is known, but it is not held at present in the Harwell subroutine library.

The other subroutines that are useful for data fitting and approximation when f(t) is a polynomial are as follows. To minimize expression (9) use PE11, and to minimize expression (8) or (11) use VC01, except that VC04 is available for least squares fitting by a straight line. In fact to maintain numerical accuracy subroutine VC01 defines f(t) in terms of orthogonal polynomials, and to find the actual coefficients of f(t) one may call PE08 after using VC01. Instead it is usually more accurate to use PE07 after VC01, but this subroutine only calculates the value of f(t) for any given value of t.

When f(t) is a rational function (3), then subroutines PE04 and PE05 are available. PE04 minimizes expression (12), and PE05 minimizes expression (12) including a weight function. Note that the range of t is now an interval, and therefore the user must provide a subroutine to calculate y(t).

In addition to the routines for interpolating a cubic spline function, the following routines are also available for the case when f(t) is a cubic spline. VB05 minimizes expression (8), and VB06 minimizes expression (8) plus a smoothing term, the purpose of this smoothing term being to damp the effect of data errors. Also there is a subroutine called VC03 that is more sophisticated than any of the routines mentioned already, for it <u>automatically</u> assigns the number and the joins of the cubic polynomial pieces of f(t). Here the intention is to take account of all the trends of the data, without following data errors. Finally there is a spline routine called TS01 that also assigns the polynomial pieces automatically. This routine is intended for the case when y(t) is a mathematical function, specified by a computer subroutine, and it aims to calculate f(t) so that throughout the range of t the difference $|f(t) - y(t)|$ is less than a prescribed tolerance.

In addition to the special purpose routines mentioned above, some of the general algorithms of the Harwell library can be used when the form of f(t) is more general. In particular we have stated already that the linear form (4) is quite manageable. In this case the function (8) can be minimized by MA14, the function (9) by MA11, and the function (10) by MA20. Also subroutine MA09 is available for the least squares problem, but usually MA14 is more accurate.

Finally there is the case (6) when f(t) depends in a general way on the parameters $(a_1, a_2, \ldots, a_n)$. Sometimes the form of this dependence can be exploited, and specially efficient subroutines can be written, for example for fitting sums of exponentials the subroutine VC05 is available. However usually a special routine is not available, and then it is necessary to calculate the function, like expression (8), (9) or (10), which is to be minimized to define $(a_1,a_2,\ldots,a_n)$, and to use a general routine for unconstrained minimization. Therefore the reader is referred to section C on methods for optimization. In particular note that the minimization of a sum of squares is relatively easy, so, unless there is an excellent reason for doing something different, one should prefer to calculate $(a_1,a_2,\ldots,a_n)$ by minimizing expression (8). The most reliable subroutines for minimizing a sum of squares are VA05, VA07 and VB01. It is awkward to minimize expressions (9) and (10) by general methods, because they have discontinuous derivatives. Therefore we hope to provide some subroutines that are designed for these calculations.

C. Optimization

1. Introduction

In this section the use of subroutines which deal with problems related to optimization will be described. These subroutines appear mostly in group V, but there are also subroutines from groups L, M and N. The ultimate aim of this section is to give a "flow diagram" whereby users can determine whether there is a suitable program in the library for their particular problem, if so which it is, and if not, what action should be taken. Some general points however will first be made.

A flow diagram of the type presented below is necessarily crude and does not take into account any special knowledge which the user might have. When possible, this special knowledge should be used to as large an extent as possible in formulating the problem, and in choosing a subroutine. Users are also expected to pose their problems with care, giving attention to such problems as scaling of variables and avoidance of cancellation, which can be of crucial importance. It is also wise that any formula for evaluating derivatives should first be checked by differences in function values. We are always keen to ensure the best use of library subroutines, so that any Harwell user who is in doubt on any of these points should seek advice from a member of the Numerical Analysis Group.

The problem under consideration will in general terms be assum3d to be that of minimizing a function $F(\underset{\sim}{x})$ of many variables $\underset{\sim}{x} = (x_1,x_2,\ldots,x_n)$. In addition the values of $\underset{\sim}{x}$ which are allowed might be restricted in that they have to satisfy equations like $c_i(\underset{\sim}{x}) = 0$ or $\geqslant 0$. Such expressions are referred to as constraints, and a number of constraint functions are referred to collectively as the vector $\underset{\sim}{c}(\underset{\sim}{x})$. It will be assumed that F and $\underset{\sim}{c}$ are differentiable, if not, seek advice. In what follows, the concept of functions which are linear in $\underset{\sim}{x}$ is important, and in particular $f(\underset{\sim}{x})$ is linear in $\underset{\sim}{x}$ if it can be expressed as $f(\underset{\sim}{x}) = \Sigma a_i x_i + b$, where a_i and b are independent of $\underset{\sim}{x}$. The problem stated in these general terms is not conveniently solved by a single algorithm, so the choice of algorithm is determined by the more detailed structure of the problem, and by what information about the problem is available.

It is often possible to simplify a formulation, and three devices in connection with this will be described. One is that linear equality constraints in the formulation can sometimes be used to eliminate variables. This can also be done with sparse non-linear equality constraints in some circumstances. Another device is that some constraints can be removed by making a non-linear transformation of variables. Typically a variable x subject to a constraint $x > 0$ can be removed by transforming to the variable $y = \log x$. When minimizing with respect to y, the constraint $x > 0$ will automatically be imposed. Further possibilities are outlined in M.J. Box, D. Davies and W.H. Swann, "Non-linear Optimization Techniques" ICI Monograph No. 5, Oliver and Boyd, London (1969). Finally in certain sums of squares problems, where

$$F(\underset{\sim}{x}) = \sum_{i=1}^{m} \left[f_i(\underset{\sim}{x})\right]^2 = \underset{\sim}{f}^T\underset{\sim}{f} ,$$

it often happens that the residuals $\underset{\sim}{f}$ are linear in some of the variables, and this can be taken into account when posing the problem. The variables supplied to the optimization device should be the non-linear variables, because every time that $\underset{\sim}{f}$ must be calculated, the linear variables can first be found by a linear least squares calculation. Care should be taken when evaluating derivatives in the modified formulation.

When considering choosing an algorithm, users are advised to evaluate first derivatives of their objective functions if at all possible. Subroutines using derivatives are much more efficient, and very reliable whereas subroutines not using derivatives fail more often than is desirable, and are also affected much more seriously by bad scaling or undue cancellation. The evaluation of second derivatives is much less desirable, and in fact only one subroutine for minimization of a general function which use such information in the most efficient way is available in the library. In this and a number of other cases, there are a number of well tried methods which do not appear in the library, but which might be included if need arose. Such possibilities are indicated in the flow diagram. When following the flow diagram, users will either be advised to use a particular subroutine, or will be recommended to seek further guidance under one of the headings ?, ??, or ???. These mean roughly as follows

? There is no method in the library, but one could be added readily, if effort were available. Users are advised to contact the subroutine librarian to see what can be done.

?? No direct techniques are available, but by using a penalty function (see R. Fletcher, "Methods for the solution of optimization problems", T.P.432, (1970) for instance) a suitable reformulation of the problem might be possible. It is hoped to introduce such a method into the library in the near future and users should check whether this has been done.

??? No direct techniques are available in the library and there is no obvious way of transforming the problem.

In cases ? and particularly ???, Harwell users are advised to seek advice from a member of the Numerical Analysis Group.

2. The flow diagram

1. Does the problem have only one variable (If No, go to 2)

   Use VD02A or VD01A depending upon whether or not derivatives of F are available.

2. Are both F and $\underset{\sim}{c}$ linear (If No, go to 3)

   Use LA01A, or consider using the IBM mathematical programming system MPS.

3. Is F a sum of squares, $F = \underset{\sim}{f}^T\underset{\sim}{f}$ (If No, go to 10)

   Questions 4-9 concern methods for minimizing sums of squares.

4. Are both $\underset{\sim}{f}$ and $\underset{\sim}{c}$ linear (If No, go to 5)

   If the constraints are all equalities then use of MA14A is appropriate. Otherwise there are special methods for the problems which do not appear in the library; either go to ?, or neglect the fact that F is a sum of squares and use either VE02A or VE04A as appropriate.

5. Are there any constraints (If No, go to 7)

6. Are both the derivatives $\partial f_i/\partial x_j$ available and the constraints linear (If No, go to ??)

A method for solving this problem efficiently could be written. Hence go to ?, or neglect the fact that F is a sum of squares and use VE01A. Feasible points in connection with this problem can be calculated using LA02A.

7. Is the matrix of derivatives $\partial f_i/\partial x_j$ sparse (If No, go to 8)

Use of NS03A is generally appropriate, whether or not the derivatives can be calculated explicitly.

8. Can you evaluate the derivatives $\partial f_i/\partial x_j$ (If No, go to 9)

A general purpose routine for this problem is VA07A. Alternatively if the problem is one of data fitting, then VB01A is the equivalent method but provides specialized input, and outputs statistical information such as standard deviations. If exceptionally the rate of convergence of these methods is poor, use VB03A.

9. Is $m = n$ (If Yes, use NS01A; if No, use VA05A)

10. Are first derivatives $\partial F/\partial x_i$ available (If Yes, go to 14)

Questions 11-13 concern methods without derivatives.

11. Are there any constraints (If Yes, go to ??)

12. Is F subject to gross errors (If No, go to 13)

Methods may exist for solving this problem, but are not available in the library. Hence go to ?.

13. Use routine VA10A. If exceptionally this fails, try VA04A. Alternatively, if it is possible to use special information to program a line search in an efficient way, use VA03A.

14. Are second derivatives $\partial^2 F/(\partial x_i \; \partial x_j)$ available? (If Yes, go to 19)

Questions 15-18 concern methods which use first derivatives.

15. Is the problem large; for instance does the need to store one or two nxn matrices cause embarrassment (If No, got to 17)

16. Are there any constraints (If Yes, go to ???)

Use VA08A.

17. Are there any constraints (If Yes, go to 18)

Use VA09A. If exceptionally this fails, try VA06A which is guaranteed to converge, apart from the effects of round-off error, but may take more evaluations of $F(\underset{\sim}{x})$ and be more susceptible to badly scaled variables.

18. <u>Are the constraints non-linear</u> (If Yes, go to ??)

Use either VE01A, VE03A or VE05A. VE03A is guaranteed to converge, and will probably require fewer objective function evaluations. VE01A uses less housekeeping time per iteration however. VE05A is particularly efficient when it is known that many constraints will be active at the solution, and when the constraints are sparse. Feasible points in connection with this problem can be calculated using LA02A.

19. <u>Is F quadratic and $\underset{\sim}{c}$ linear ?</u> (If No, go to 20)

If the constraints are all either upper or lower bounds (like $a \leqslant x \leqslant b$), then use VE04A. Otherwise use VE02A.

<u>Questions 19-21 concern methods using second derivatives.</u>

20. <u>Are there any constraints</u> (If Yes, go to 21)

Use VA11A.

21. <u>Are the constraints non-linear</u> ? (If Yes, go to ??)

There are no methods in the library which use all the information to its full extent. However such a method could readily be written: hence go to ?, or alternatively neglect the availability of second derivatives and go to 18. Feasible points in connection with this problem can be calculated using LA02A.

D. Linear Algebra

1. Introduction

In this section we consider

1) the solution of a system of n linear equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \qquad i=1,2,\dots,n \tag{1}$$

which may be written in matrix notation as the equation

$$Ax = b; \tag{2}$$

2) the evaluation of the inverse of a matrix A;

3) the evaluation of the determinant of the matrix A;

4) the approximate solution of the overdetermined system of linear equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \qquad i=1,2,\dots,m \tag{3}$$

with $m > n$, and associated problems;

5) the eigenvalue problem of finding x, λ such that

$$Ax = \lambda Bx, \tag{4}$$

where B is usually the identity matrix I, and

6) matrix and vector multiplication.

These problems are handled by sections M and E of the subroutine library.

We will distinguish between the cases where A is full (all or near all elements non-zero) or sparse (where so many elements are zero that it is worthwhile to take special account of them) and the cases where A does or does not have the property of being symmetric (or Hermitian in the complex case) and positive definite, that is $a_{ij} = \bar{a}_{ji}$ for all i,j and

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \bar{x}_i a_{ij} x_j > 0$$

unless $x_i = 0$ for $i=1,2,\dots,n$.

2. Full matrices: solution of linear equations, matrix inversion and determinant evaluation

For the case where A is full the subroutines MA21A and MA22A perform any of the three jobs in question for the general and the symmetric positive-definite cases, respectively. Also there are versions for use when all the elements are complex, MA23A handling the general case and MA24A the case where A is Hermitian positive definite. These subroutines are described in an AERE report (Marlow and Reid, 1971). This report also contains a brief summary of the mathematical background to the method of Gaussian elimination. These subroutines include facilities for the rapid solution of further systems with the same matrix and for providing error estimates based either on the computer word-length or on user-supplied information. They supersede subroutines MA01B, MB01B, MD03A, ME04A; it is intended that these be withdrawn gradually from the library and so they will not be compared or described here.

3. Solution of sparse systems of linear equations of band form

A form of sparse matrix that arises quite frequently and can easily be exploited during Gaussian elimination is a band matrix ($a_{ij} = 0$ if $|i-j| > k$). Very substantial savings in storage and computing time result from this. MA07B handles the general case with real coefficients; MA07A also handles this case and is more economical in storage and computing time because it uses no interchanges but is dangerous to use except where it is known that no inchanges are needed, e.g.

$$\text{if } |a_{ii}| \geqslant \sum_{j \neq i} |a_{ij}|$$

for all i or

$$|a_{ii}| \geqslant \sum_{j \neq i} |a_{ji}|$$

for all i. MA15C handles the symmetric and positive-definite band case and takes advantage of any variability in band-width, see Jennings (1966); it can treat very large systems since it uses backing store if necessary and so is very suitable, for example, for solving equations arising from discritization of two-dimensional elliptic partial differential equations. It is described by Reid (1972).

4. Solution of very sparse systems of linear equations

For matrices that are very sparse (e.g. an average of 4 to 10 non-zeros per row) and do not have a compact band structure, three subroutines, MA16A, MA17A or MA18A are available. MA18A treats the general real case by Gaussian elimination, storing only the non-zeros and choosing pivots in a way designed to limit any increase in the number of non-zeros; it can treat very efficiently further systems with the same matrix or another matrix with the same pattern of zeros and non-zeros. It has been documented by Curtis and Reid (1971). MA16A and MA17A treat the case where the matrix is symmetric and positive definite. MA17A is a version of MA18A that has been designed to exploit the symmetry but apart from this its method and facilities are very similar. It has been described by Reid (1972). MA16A is quite different since it uses the method of conjugate gradients; very large systems can be solved since the user is required only to provide a subroutine which calculates Ax after being given x and the explicit storage of A is unnecessary. This has been documented by Reid (1970).

5. Approximate solution of over-determined systems of equations

Since in general we cannot solve an overdetermined system exactly we must be content with minimizing some measure of the error b-Ax. The most frequently used measure is the sum of squares

$$\sum_{i=1}^{m} \left[b_i - \sum_{j=1}^{n} a_{ij}x_j \right]^2 \; ; \tag{5}$$

this is minimized at the solution of the linear system

$$A^T Ax = A^T b \, , \tag{6}$$

known as the normal equations (see, for example, Fox, 1964, page 81). Subroutine MA08A constructs this system and MA09A both constructs and solves it, assuming that it is non-singular. A disadvantage of this technique is that the system (6) tends to be ill-conditioned, that is small errors in A^TA or A^Tb have a comparatively large effect on x. The problem can be overcome by transforming the system using elementary orthogonal transformations and so avoiding the formation of the system (6) (see Businger and Golub, 1965); subroutine MA14A proceeds in this way and allows the user to specify that some of his equations be satisfied exactly. MA09A is however faster than MA14A. Both subroutines provide, if required, $(A^TA)^{-1}$ which may be used to estimate the variances and covariances of x_i, $i=1,2,\ldots,n$.

In the case where A^TA is singular the least squares problem (5) does not have a unique solution. If we choose from all those vectors which minimize (5.1) the one with least value for

$$\sum_{i=1}^{n} x_i^{\,2}$$

then we obtain a problem with a unique solution and the matrix A^+ (generalised inverse of A) is defined by the condition that A^+b is this unique vector for all possible vectors b. Where A^TA is non-singular $A^+ = (A^TA)^{-1}A^T$ and where A is square and non-singular $A^+ = A^{-1}$. Subroutine MB10A calculates this generalised inverse, if requested, will estimate the rank of A and MB11A finds A^+ in the case where A^TA is non-singular (but note that it is written as if m and n were interchanged and A replaced by A^T).

In addition routines MA11B and MA20A are available for solving the over-determined problem in the sense of minimizing the expressions

$$\max_{i=1,..,m} \left| b_i - \sum_{j=1}^{n} a_{ij}x_j \right| ; \qquad m \geqslant n \tag{7}$$

and

$$\sum_{i=1}^{m} \left| b_i - \sum_{j=1}^{n} a_{ij}x_j \right| ; \qquad m \geqslant n \, , \tag{8}$$

respectively. In general it is more expensive to solve these problems than the least squares one since a sequence of subproblems have to be solved by the simplex algorithm of linear programming or an equivalent procedure. Measure (7) is suitable in the approximation problem where it is important that each error be small. Measure (8) has an advantage in data-fitting in that isolated equations with gross errors tend to be ignored.

6. The eigenvalue problem

All the techniques used in the library are described by Wilkinson (1965).

The eigenvalue problem for a symmetric matrix is far more straightforward than for an unsymmetric matrix for it is better conditioned and all eigenvectors always exist.

The simplest technique is that of Jacobi, which is an iterative method which may be used to find all eigenvalues and eigenvectors. The convergence will be particularly rapid for matrices all of whose off-diagonal elements are small. Subroutine EA03A implements a variant of this algorithm.

In general, however, Jacobi's method is slower than the use of Householder transformations to reduce the matrix to tridiagonal form (MC04B) followed by the QR algorithm to find the eigenvalues (EA09C) or the eigenvalues and eigenvectors (EA08C). Subroutines EA06C and EA07C are available to provide a single call for all eigenvalues and eigenvectors or all eigenvalues of a given symmetric matrix.

Hermitian matrices may be handled in a very similar way, and ME08A, EC06C, EC07C, EC08C and EC09C correspond exactly to MC04B, EA06C, EA07C, EA08C and EA09C.

If a single eigenvalue, nearest to a given number, and corresponding eigenvector is required, then EA02A is available. This uses inverse iteration.

For the unsymmetric eigenvalue problem EB07A is available to find all the eigenvalues and EB06A is available for all eigenvalues and eigenvectors. It should be noted that this problem may be very ill-conditioned. The method used involves first choosing a scaling similarity transformation DAD^{-1} so that the matrix is better balanced (MC15A), then an orthogonal reduction to Hessenberg form (MC14A) and finally the QR algorithm applied to the Hessenberg matrix (EB08A for eigenvalue and eigenvectors and EB09A for eigenvalues only). These routines are Fortran versions of the Algol procedures of Wilkinson and Reinsch (1971).

For the general eigenvalue problem $Ax = \lambda Bx$ where A is symmetric and B is symmetric and positive definite the subroutine EA11A is available. This performs a Cholesky decomposition of B in order to reduce the problem to a standard eigenvalue problem for a symmetric matrix.

7. Matrix and vector multiplication

Subroutines MC01AS, MC02AS, MC02BS and MC03AS, written in IBM 360 assembler language are available for matrix multiplication and vector inner products. They are faster than any comparable FORTRAN loop and are accurate since all intermediate results are held in double precision. MC09A is a Fortran subroutine for computing the product of a sparse matrix and a full vector.

References

P. Businger and G.H. Colub (1965) Linear least squares solutions of Householder transformations. Numer. Math. 7, 269.

A.R. Curtis and J.K. Reid (1971) Fortran subroutines for the solution of sparse sets of linear equations. AERE report R-6844.

L. Fox (1964) An introduction to numerical linear algebra. Clarendon Press, Oxford.

A. Jennings (1966) A compact storage scheme for the solution of symmetric linear simultaneous equations. Comp. J. 9, 281.

S. Marlow and J.K. Reid (1971) Fortran subroutines for the solution of linear equations, inversion of matrices and evaluation of determinants. AERE report R 6899.

J.K. Reid (1970) A Fortran subroutine for the solution of large sparse sets of linear equations by conjugate gradients. AERE report R-6545.

J.K. Reid (1972) Two Fortran subroutines for direct solution of linear equations whose matrix is sparse, symmetric and positive-definite. AERE report R-7119.

J.H. Wilkinson (1965) The algebraic eigenvalues problem. Clarendon Press, Oxford.

J.H. Wilkinson and C. Reinsch (1971) Handbook for automatic computation, Vol. II, linear algebra. Springer-Verlag.

E. Quadrature

1. Introduction

In this section we consider the problem of evaluating the definite integral

$$I(a,b;\ f) = \int_a^b f(x)\,dx \tag{1}$$

or various generalisations of it. The relevant routines fall into four categories, as follows:

QA these are routines to solve the simple problem specified in equation (1), except that QA03A produces the answer for a series of values of b.

QB the one routine in this group evaluates a multi-dimensional integral, for up to six dimensions.

QD the single routine in this group is used for evaluating Fourier integrals where the trigonometric term is highly oscillatory.

QM there is again only one routine in this group, for multi-dimensional integrals up to twenty dimensions, using a Monte Carlo method.

There is also a routine, QC02AD, which does not evaluate any integral of the form (1) or similar, but instead calculates weights and evaluation points (see below) enabling certain integrals to be evaluated by the user. This routine is described in Section 5.

The methods available for numerically evaluating equation (1) all choose a number of values x_i, i=1 to n, of the argument x, and approximate to I by a suitable linear combination of the integrand values $f(x_i)$

$$I \simeq \sum_{i=1}^{n} w_i\, f(x_i) \tag{2}$$

where the numbers w_i (usually positive) are called weights. Methods for numerical integration are called adaptive if the choice of the number n of sampling points and their positions x_i is influenced directly by the values of the integrand found at earlier sampling points; in the opposite case, the number and position of the x_i is determined only by the limits a and b of integration and possibly by comparing some estimate of error with a requested tolerance. Generally speaking, adaptive methods are more efficient with difficult integrands (e.g. those containing an integrable singularity, or which vary much more rapidly in some parts of the interval of integration than in others), especially if fairly high accuracy is required.

Integration is a numerical process capable inherently of quite high accuracy, because of the smoothing effect which results from adding values of the integrand with positive weights, thus giving rounding errors a chance to cancel. Clearly this basic stability can be eroded if there is considerable arithmetic cancellation in evaluating the integral, i.e. if f(x) is somewhere positive and elsewhere negative, and the value of I is small compared with that of

$$J(a,b;\, f) = \int_a^b |f(x)|\, dx \,. \tag{3}$$

For this reason therefore, if one requests a specified relative accuracy in I (e.g. that the error should not exceed $10^{-6}I$), it is likely that the subroutine will in fact attempt to limit the error to the specified accuracy relative to the value of J.

Another instance where the basic numerical stability of quadrature may not be fully reflected in a practical subroutine is where the subroutine tries to assess the local accuracy by carrying out numerical operations on values of the integrand which amount to finite differencing. For example, it would be reasonable for an adaptive routine to take evaluation points x_i at a certain spacing h, using finite differences of the integrand values $f(x_i)$ as estimates of the derivatives of f(x); it might then try to choose h so as to achieve some accuracy requirement, reducing h if it seemed that the accuracy requirement would not be met. However, if the computed values of $f(x_i)$ were subject to considerable rounding error, the finite differences would be dominated by this rounding error, and reducing h would not reduce them; the subroutine would therefore continue to reduce h in a fruitless attempt to achieve the requested accuracy, until some lower bound was reached. This would be very inefficient; instead, it would be desirable to recognise that the accuracy was not attainable, and to give the best practicable answer.

2. Subroutines for one-dimensional quadrature (group QA)

This group tackles directly the problems stated in (1), using approximations of the form (2). There are five routines in the group, one non-adaptive and the others adaptive. The non-adaptive routine, QA01AS, must be used if the integrand f(x) is defined only by a table of equally spaced values, and the information given in the write-up may enable the accuracy obtainable to be assessed. Although the write-up describes a possible iterative technique for achieving a required accuracy in the case where f(x) can be computed for any x by means of a subroutine, this method of use of the routine is not likely to be efficient compared with the alternative of using one of the adaptive subroutines, especially if the integrand varies more rapidly in some parts of the range than in others. Therefore, use of this routine is recommended only when the adaptive routines cannot be used because f(x) cannot readily be computed between tabulated values.

It is convenient to discuss the three routines QA02A, QA04A and QA05A, together. They all have similar objectives, namely to achieve a given requested accuracy in evaluating equation (1) All will fail if $b \leq a$. There are minor differences, especially in the way the error request is interpreted by the routine. QA02A interprets it as an absolute error on the value of I, QA04A interprets it as an error relative to J, while QA05A can be given two accuracy parameters, one to be interpreted as absolute and the other as relative, and will attempt to meet whichever turns out to be less demanding.

There is also a considerable difference in the basic approach of the routines. QA02A starts at one end of the range, and works towards the other, trying to achieve an error which is uniformly distributed over the range. It therefore needs little work space, because it carries only a few function values at a time. The others first cover the range fairly coarsely, obtaining first approximations to I and J, and also an estimate of the error on I. If, as normally happens, this error estimate is too large, they then sub-divide the range, stacking function values for part of it in temporary work space, and refining the sub-division of the other part. This process can be nested quite deeply (e.g. to 30 levels in QA04A) with the result that quite a lot of work space is needed, and the routines occupy more memory than QA02A. As soon as they consider they have achieved adequate accuracy on one sub-interval of the range, they then unstack another sub-interval and choose further evaluation points x_i within this one, again nesting down until sufficient accuracy has been achieved. They also use higher order integration formulae on each sub-interval than QA02A (which uses Simpson's rule), with the result that they are considerably more economical in function evaluations except when only modest accuracy is required. Moreover, as it happens QA04A and QA05A have been made considerably more robust than QA02A in the face of singularities or discontinuities in the integrand function f(x).

All three of the routines have their own individual (one might almost say eccentric) ways of indicating that they may not have been able to achieve the requested accuracy, and of nevertheless returning the best approximation they can. So far as efficiency is concerned (measured by the number of function evaluations taken to achieve the result), it is convenient first to discuss the difference between QA02A and the others, and then to distinguish between QA04A and QA05A separately. Many tests have been carried out, and although it is difficult to generalise with absolute confidence, nevertheless a very rough guiding principal for integrands which do not present a special difficulty one may expect QA02A to be competitive for error tolerances larger than about 10^{-4} (relative accuracy), and to be no longer competitive when the tolerance is 10^{-6} or less. It has tended to perform quite well on smooth oscillatory integrands with perhaps 10-100 cycles of oscillation in the range. However, it should be definitely avoided if the integrand has any kind of discontinuity or singularity (e.g. like $x^{-\frac{1}{2}}$ at one end of the range) or if it is subject to rounding error. On smooth integrands it may be worth using even for high accuracies (of course, double precision versions should be used when relative errors below about 10^{-6} are wanted), if it is important to save storage space. However, in such cases use of QA04AD should be considered if at all possible.

We turn now to the comparison between QA04A and QA05A. First we note that QA05A is a very large subroutine, occupying over four times as much memory as QA04A. As against this, QA05A is noticeably more efficient at accuracies better than about 10^{-4} on most problems, and the efficiency ratio may become large at really high accuracies. QA05A is also better on integrands with various types of singularity which it has been programmed to recognise, but it may give disappointing results compared with QA04A on singularities of other types. Both routines are quite rugged in the presence of rounding error, although since this more commonly makes itself felt in situations where high accuracy has been requested QA05A is frequently more economical in such cases.

On the whole, the choice between QA04A and QA05A could well be based on the following considerations: if memory requirements are likely to be a problem choose QA04A; or else if high accuracy is required choose QA05A, but otherwise choose QA04A. On further consideration, not referred to previously, which might be of importance in some cases is the minimum number of function evaluations which the subroutine can possibly use, even on a very simple integrand. The point here is that if this number is too small the subroutine might erroneously conclude that the integrand was very simple, while with a larger minimum number of evaluation points the probability of this happenining must be considerably reduced. The minimum possible numbers are 17 for QA02A, 9 for QA04A, and 5 for QA05A; however, QA05A does have some additional protection built in for the cases in which the values of the integrand at five equally spaced points lie on a straight line.

We turn now to QA03A, which carries out a slightly different function from the others. It tabulates the value of the integral at equally spaced values of the upper limit b. It is closely similar to QA02A in internal design, and so on integrands which offer no particular difficulty its performance is likely to be comparable. However, some modifications which were recently made to QA02A to make it more rugged in the face of rounding errors or singularities have not been made to QA03A, and while no instance is known of difficulty being caused by this, it is clearly conceivable that on some problem it might arise. If this happened, the remedy would be to use one of the other adaptive routines over each sub-interval for which the interval was required, accumulating the sum of the results to obtain the answers.

3. Multi-dimensional quadrature (groups QB and QM)

Subroutines QB01A and QM01A fulfil fairly closely related functions, but the methods they use are very different. Thus it is easy to understand why they are in different groups, but nevertheless it is useful to treat them in a single discussion. QB01A treats a multi-dimensional integral as a repeated integral, using as its integration method the "product" of separate conventional one-dimensional methods in each of the dimensions. The one-dimensional methods available are discussed below. An advantage of the repeated integral approach is that it is easy to allow the limits of an inner integral to depend on the variables of integration of outer integrals, and this facility is provided. QM01A, on the other hand, treats the repeated integral as one over a volume of n-dimensional space, generating points at random in this space for estimation of the integral by a Monte Carlo sampling technique. The shape of the region must therefore be rectangular. An advantage of this approach is fairly easily explained by means of a simple example. Suppose that a six-dimensional integral has to be evaluated, and that to achieve the required accuracy one would need approximately n points for each one dimensional quadrature involved. Then the total number of evaluation points required will be $N=n^6$ and this can rapidly become large with n. With the Monte Carlo approach, one chooses in advance how many evaluation points to use, and because there is no particular relationship between the individual co-ordinates of one evaluation point and those of another, one hopes that N need not be so large. In fact, this will depend very much on the accuracy required, but certainly for moderate accuracy the Monte Carlo approach is likely to be more efficient when the number of dimensions is high.

Let us consider now the two routines separately, starting with QB01A. The user is provided with a facility by which he can choose one of three types of integration method for each of the nested one-dimensional integrals. One choice available is an adaptive Simpson's rule (this is of the nested type used in QA04A and QA05A, not the progressive type used in QA02A, so that storage is needed for stacking function values which will be needed later). An alternative method is Chebyshev integration based on interpolating the integrand with a polynomial of degree 2^r, where r is varied up to 7 if necessary in an attempt to achieve the required accuracy. Because the evaluation points are determined in advance for each r, the spacing cannot be varied to be closer where the integrand varies more rapiely, so this is not a truly adaptive method. However, it will often be the case that the outer integrations of a repeated integral will have fairly smooth integrands, and in this case very good results may be obtained for quite modest values of r; for the inner integrals, one would expect Simpson's rule to give better results. It is also possible for the user to specify an n point Gauss quadrature formula for values of n up to 15; no error estimate is possible in this case, but nevertheless this may be a useful facility where earlier experiments have shown what values of n to use for integrals of the type concerned. There would then be two advantages: (i) the total number of evaluation points to be used would be known in advance, so that the computational cost could be controlled; (ii) because the integration points used are definitely fixed, the approximation to the integral which is computed will be a smooth function of any parameters on which the integrand (but not the limits) depends in a smooth manner, and this may be useful if it is intended to think of t' integral as a function of these parameters.

Turning now to QM01A, this provides the facility of using the "weighted uniform sampling" method. If the integrand is a function $f(\underline{x})$, where $\underline{x}$ is the vector of co-ordinate values, and if a function $g(\underline{x})$ is known having the property that the integral of $g(\underline{x})$ is unity, and g varies in a rather similar manner to f, then the variance of the error in evaluating the integral is related to the variance of the ratio f/g, instead of merely to the variance of f. As an example, suppose that integrand was a smooth function multiplied by 1/r, where r was the distance from a point in the multi-dimensional region (we can assume that the point where r=0 would not be used as an evaluation point). It may be that the integral of 1/r over the region could be evaluated exactly, so that we could take $g(\underline{x}) = c/r$, where c was a constant chosen to make the integral of g exactly equal to unity. Then we would expect to get the integral of f quite accurately, since the ratio f/g would not vary nearly so much as f itself. One might even go further, and use this technique in the case where the region of integration was not rectangular in shape. One would define both f and g to be zero (or to have some very small value) outside the true region of integration, and would enclose the latter in a rectangular region over which the integral was requested. There would be an inefficiency due to the fact that some evaluation points would fall outside the true region of integration, and so would be wasted; however, the method ought to work, and the inefficiency would be small if the volume of the enclosing rectangular region was not much greater than that of the desired region of integration.

Because the relative efficiency of Monte Carlo methods tends to increase with increasing number of dimensions, it is probably wise to prefer QB01A for those problems of dimension up to six for which it can be used, restricting QM01A to higher dimensional problems. However, such advice can only be tentative, and if many similar integrals are likely to be needed some experiment may well be worthwhile.

4. Fourier integrals (group QD)

There is only one subroutine in this group, namely QD01A, and it uses a well known method to evaluate Fourier type integrals in cases where the "frequency" is large, so that the integrand as a whole is highly oscillatory. The method attains an accuracy comparable to that which would be obtained applying Simpson's rule to the non-oscillatory factor only, and this is usually highly satisfactory. Because the method needs equally spaced function values, it cannot be truly adaptive, but instead operates in a manner rather similar to that described for the Chebyshev option in QB01A. The subroutine is definitely to be preferred to an ordinary adaptive method applied to the whole integrand in cases where the trigometric factor oscillates through a considerable number of complete cycles in the range of integration.

5. The subroutine QC02AD

This subroutine can be used to calculate weights w_i and evaluation points x_i in order to use (2) as an approximation to

$$I = \int_0^\infty e^{-x} f(x)\, dx \tag{4}$$

The well known Gauss-Laguerre quadrature formulas are commonly used to evaluate integrals of this type, but they are designed to be efficient in the case where f(x) is a polynomial whose degree does not exceed 2K-1 for a K-point formula. Experience has shown that the Gauss-Laguerre formulae are inefficient when used with functions f(x) which do not tend to infinity with x, as a polynomial must do, but instead tend to a finite limit. Subroutine QC02AD may be used to obtain weights w_i and evaluation points x_i, for a range of values of K; these can be used in exactly the same way as Gauss-Laguerre weights and evaluation points, but can often give much more rapid convergence to the true value of I as the order K is increased.

Unfortunately, it is necessary for the user to supply some qualitative information about the range of x in which most of the variation of f(x) takes place. This is supplied in the form of a parameter a; while the choice of a is not too critical, nevertheless better results will be obtained if it is chosen wisely, and the following considerations may be helpful. If, for example, f(x) varies monotonically from its value at x=0 to its limit as x tends to infinity, then a good value of a to choose would be such that f(a) is half way through this range of variation of f. If f(x) oscillates several times before tending to a limit, then the value of a should be chosen so that there are roughly as many major oscillations of f to the left of x=a as to the right of it. If nearly all the variation of f is concentrated in an interval $x_1 < x < x_2$, then one could choose $a=(x_1,x_2)^{\frac{1}{2}}$. Generally speaking, the object is to choose a so that the graph of f(x) against the variable z=x/(a+x) would be unduly cramped towards z=0 or towards z=1.

For the sake of compatibility with an earlier version of QC02AD the arrays which the user must supply for the subroutine to return its results in have fixed dimensions. This may be inconvenient to a user who wants only the results for fairly low values of K. The user has also to supply an error tolerance which the subroutine will use in evaluating the weights and values, and it is suggested that this tolerance should be made fairly small, since the routine, which has recently been considerably modified, is not unduly expensive in computer time, and in any case is not likely to be used very frequently. The routine is capable of very high accuracy if required, up to the almost limit of the double precision word.

It is perhaps worth repeating that this subroutine does not evaluate the integral (4); it merely returns numbers which the user may substitute in equation (2) in order to evaluate the integral.

AERE - R 7477
Supplement No. 1

HARWELL SUBROUTINE LIBRARY

A CATALOGUE OF SUBROUTINES (1973) SUPPLEMENT NO. 1

Compiled by M. J. Hopper

This supplement gives a list of new subroutines introduced into the Library in the period August 1973 to July 1974.

Computer Science and Systems Division,
AERE, HARWELL.

August, 1974

HL74/3159(C13)
/KE

ISBN-0-70-580254-X

DD03AD

Solves systems of ordinary differential equations constituting two-point boundary-value or eigenvalue problems having the form

$$y_i' = g_i(t,y_1,\ldots,y_n,\mu_1,\ldots,\mu_p) \quad i = 1,2,\ldots,n \qquad a \leqslant t \leqslant b$$

with boundary conditions of the form

$$h_i(y_1(a),\ldots,y_n(a),y_1(b),\ldots,y_n(b),\mu_1,\ldots,\mu_p) = 0 \quad i = 1,2,\ldots,n+p$$

The method is an extension of the multiple shooting method by M.R. Osborne (J. Math. Anal. Appl. 27, 1969) and H.B. Keller (Numerical Methods for Two-point Boundary Value Problems, Blaisdell, 1968) and using a 4th order Runge-Kutta method by R. England (Computer Journal, 12, 1969) for the associated initial value problems.

The provision of shooting and matching points is optional. The user must specify a required accuracy and can set a limit on the number of integrations over $a \leqslant x \leqslant b$. Starting values for y_i $i = 1,2,\ldots,n$ must be given and subroutines provided to evaluate g_i and h_i.

Versions: DD03AD; no single precision version.
Calls: MC02AD, NS03AD.
Language: FORTRAN, Date: Oct. 1973, Size: 21.5K; 737 cards.
Origin: J.K. Reid, Harwell.

FP01A

Given an interval (α,β), $\alpha < \beta$, this subroutine automatically determines axis limits, label points and label formats suitable for drawing a labelled axis on the graph plotter.

A new interval (a,b) and a label interval h is calculated such that a and b are multiples of h and round figure numbers which satisfy $\alpha \leqslant a < \beta \leqslant b$. No graph plotting is performed.

Versions: FP01A
Language: FORTRAN, Date: May 1974, Size: 2.5K; 255 cards.
Origin: M.J. Hopper, Harwell.

MA19A

Calculates the minimax solution of a system of m linear algebraic equations in n unknowns, $m \geqslant n$, where the maximum element of the solution is subject to a simple bound. Given equations

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

find the solution x_j $j = 1,2,\ldots,n$ such that

$$\max_i \left\{ \left| \sum_{j=1}^{n} a_{ij} x_j - b_i \right| \right\}$$

is minimized subject to the bounds $|x_j| \leqslant g \quad j = 1,2,..,n.$

A variation of the 'exchange algorithm' is used that incorporates a technique for reducing the number of iterations, and which will also provide a defined solution even when the matrix $\{a_{ij}\}$ is rank deficient.

Versions: MA19A; MA19AD
Language: FORTRAN, Date: March 1974, Size: 8.3K; 336 cards.
Origin: K. Madsen, Copenhagen.

OB01A

Prints a string of EBCDIC characters on the graph plotter using the GHOST graphics facilities. The association of symbol with EBCDIC code is compatible with a line printer fitted with the Harwell special text chain. Thus, the routine can be used to simulate line printer output.

Versions: OB01A.
Calls: IC01AS.
Language: FORTRAN, Date: May 1974, Size: 2.6K; 128 cards.
Origin: M.J. Hopper, Harwell.

PA06AD

To find all the real and complex roots of a polynomial with complex coefficients, i.e. calculate the zeros of

$$a_o + a_1x + a_2x^2 + \dots + a_nx^n = 0$$

The user can supply error bounds on the coefficients of the polynomial and the routine returns bounds on the moduli of the errors in the roots.

The roots are found by the method of Madsen (BIT, 13, 71-75, 1973) and error bounds by the application of Rouche's theorem as recommended by Wilkinson (J. Inst. Maths. Applics., 8, 16-35, 1971).

Versions: PA06AD; there is no single precision version.
Language: FORTRAN, Date: November 1973, Size: 8.1K; 271 cards.
Origin: K. Madsen, Copenhagen, and J.K. Reid, Harwell.

PA07AD

To find all the real and complex roots of a polynomial with real coefficients, i.e. calculate the zeros of

$$a_o + a_1x + a_2x^2 + \dots + a_nx^n = 0$$

The user can supply error bounds on the coefficients of the polynomial and the routine returns bounds on the moduli of the errors in the roots.

The roots are found by the method of Madsen (BIT, 13, 71-75, 1973) and error bounds by the application of Rouche's theorem as recommended by Wilkinson (J. Inst. Maths. Applics., 8, 16-35, 1971).

Remark: This routine supersedes PA01A/AD
Versions: PA07AD; there is no single precision version.

Language: FORTRAN, Date: November 1973, Size: 8.2K; 373 cards.
Origin: K. Madsen, Copenhagen, and J.K. Reid, Harwell.

QG01A

To integrate a cubic spline S(x) between limits which are knot points, i.e. evaluate the integral

$$\int_{\xi_j}^{\xi_k} S(x)\, dx$$

where ξ_k and ξ_j are two knot points of S(x).

Versions: QG01A; QG01AD.
Language: FORTRAN, Date: March 1974, Size: .8K; 46 cards.
Origin: M.J. Hopper, Harwell.

QG02A

To integrate a cubic spline S(x) between limits which need not be knot points, i.e. evaluate the integral

$$\int_{a}^{b} S(x)\, dx$$

where S(x) is defined as zero outside the range of its knots.

Versions: QG02A; QG02AD.
Calls: QG01A.
Language: FORTRAN, Date: March 1974, Size: 1.2K; 74 cards.
Origin: M.J. Hopper, Harwell.

TC01A

Given a cubic spline S(x), an estimate x_0 and a spline value y, the routine finds the point ξ nearest to x_0 such that $S(\xi) = y$.

The method is to locate the knot interval containing x_0, then examine successive nearest knot intervals for the presence of a real root of S(x) - y.

Versions: TC01A; TC01AD.
Language: FORTRAN, Date: April 1974, Size: 3.1K; 283 cards.
Origin: S. Marlow, Harwell.

TG02A

To compute function values and the 1st, 2nd and 3rd derivative values of a cubic spline S(x) at any given point x.

The spline and its derivatives are defined to be zero outside the range of its knot points. An option is provided for reducing the search time for the knot interval containing x making tabulation of the spline and its derivatives economical.

Symmetric formulae are used for the spline and its derivatives.

Versions: TG02A; TG02AD.
Language: FORTRAN, Date: March 1974, Size: 1.4K; 109 cards.
Origin: M.J. Hopper, Harwell.

VE06A

To find $\underline{x} = \{x_1, x_2, .., x_n\}$ that minimizes a quadratic function of the form

$$Q(\underline{x}) = q_0 + \frac{1}{2}\underline{x}^T H\underline{x} + \underline{g}^T\underline{x}$$

where H is symmetric positive definite and $\underline{x}$ is subject to linear constraints $C\underline{x} \leqslant \underline{b}$ and/or simple bounds $l_j \leqslant x_j \leqslant u_j$ $j = 1,2,..,n$. The matrix H and constraint matrix C are assumed to be sparse and the nonzero elements of these two matrices must be provided in a compact form.

The routine could be used for linear least squares data fitting problems which are subject to constraints and are sparse.

Versions: VE06A; VE06AD.
Calls: LA03A.
Language: FORTRAN, Date: June 1974, Size: 20.1K; 1114 cards.
Origin: D. Divall, Southampton.

VF01A

To minimize a general function $f(\underline{x})$ of several variables $\underline{x} = \{x_1,x_2,..,x_n\}$ subject to general nonlinear constraints $c_i(\underline{x}) = 0$ $i = 1,2,..k$ and $c_i(\underline{x}) \geqslant 0$ $i = k + 1,..,m$, $k \leqslant n$. Derivatives of $f(\underline{x})$ and all the functions $c_i(x)$, $i = 1,2,..,m$, must be provided.

The method is described by Fletcher in the A.E.R.E. internal report CSS 2, Dec., 1973. A penalty function of the form

$$\phi(\underline{x},\underline{\theta},\underline{\sigma}) = f(\underline{x}) + \frac{1}{2}\sum_{i=1}^{m} \sigma_i(c_i(\underline{x}) - \theta_i)^2$$

is used.

The user must provide subroutines to calculate derivatives and function values, and must give an initial estimate of the solution (which need not be feasible) and accuracy requirements.

Versions: VF01A; VF01AD.
Calls: MC11A, VA09A, VE04A.
Language: FORTRAN, Date: September 1973, Size: 25.7K; 245 cards.
Origin: R. Fletcher*, Harwell.

VG01A

To find the minimax solution to a system of nonlinear equations $f_i(x_1,x_2,..,x_n) = 0$ $i = 1,2,..,m$. Derivatives of the f_i with respect to the x_j are required.

* No longer at Harwell.

The method is described in a Harwell internal report TP.559 by K. Madsen.

The user must provide a subroutine to compute function values and derivatives and provide an initial approximation to the solution. The number of iterations can be limited and accuracy requirements specified.

The routine can be used for data fitting, $m > n$, when minimax is preferred to least squares.

Versions: VG01A, VG01AD.

Calls: MA19A.

Language: FORTRAN, Date: June 1974, Size: 2.8K; 92 cards.

Origin: K. Madsen, Copenhagen.

VG02A

To find the minimax solution to a system of nonlinear equations $f_i(x_1,x_2,..,x_n) = 0 \quad i = 1,2,..,m$. Derivatives are not required.

The method is described in a Harwell internal report TP.559 by K. Madsen.

The user must provide a subroutine to evaluate f and provide an initial approximation to the solution. The total number of iterations can be limited and accuracy requirements specified.

The routine is applicable to the minimax data fitting problem.

Versions: VG02A, VG02AD.

Calls: MA19A.

Language: FORTRAN, Date: June 1974, Size: 5.9K; 205 cards.

Origin: K. Madsen, Copenhagen.

GENERAL INDEX

This is a revised index which covers both the original 1973 catalogue and this supplement, it supersedes all previous indexes. New references are indicated by the mark ⁽¹⁾.

References to Harwell and Theorectical Physics Division reports take the form R-nnnn or TP-nnn. The Computer User's Manual references, TP-389, are identified by the prefix CUM and refer to the chapter or appendix number.